INTEREX presents the HP Computer Users Conference

NEW ORLEANS 1992

August 23-27, 1992

Proceedings

Volume 1

Sponsored by **INTEREX**, the International Association of Hewlett-Packard Computer Users

INTEREX

The International Association of
Hewlett-Packard Computer Users

Proceedings

Volume 1

of the

1992 INTEREX
HP Computer Users Conference

in

New Orleans, Louisiana
August 23-27, 1992

Index by Paper Number

1001 Performance Comparison of the A990
Robert Combs - Combs International

1002 Lilly Fermentation Process Control System
Lincoln Brill - Eli Lilly and Co.

1003 An IMAGE-II Data Extraction Accelerator
Don Wright - Interactive Computer Technology

1004 Using the Mouse for Screen Data Entry On the HP 1000
Don Wright - Interactive Computer Technology

1005 What to Expect From RTE 6.0
Esther Heller, Rebecca Carroll - Hewlett Packard Co.

1006 Interrupts and RTE
David Medlicott - Hewlett Packard Co.

1007 SoftBench Link/1000 Encapsulation: A State-of-the-Art CASE Tool
Gene Marshall - Hewlett-Packard Co.

1009 What's in a Namesaver?
Todd Poyner - Hewlett-Packard Co

1010 Symbolic Links on RTE-A /VC PLUS
Gary Gan - Hewlett-Packard Co.

1013 Advanced HP 1000 Assembly Language: Writing CDS Code
Alan Tibbetts - Consultant

1014 Using BSD IPC to Synchronize RTE & HP-UX System Times
Wendy King - US Naval Observatory

2001 Evolution of Software Configuration Management
Jim Sterne - Softool Corporation

2002 Computer Access and Resource Monitoring
Yagya Dutt Arya - Indian Institute of Technology

2003 Automatic Backup, Archival And Retrieval In A Heterogeneous Network
Richard Preece - Hi-Comp GmbH

2004 Changing The Rules for Data/Voice Integration
Ken Guy - MICOM Communications Corp.

2005 LAN Manager - A Case Study
Jimmy W. Frazier - Hoechst Celanese Corporation

2006 Portability Needs Under HP-UX Application
P. Viallefont - C.N.E.S.

2007 Migrating to "OPEN"
John McCabe - Lever Brothers Company

2008 Defining HP 3000 Printers in HP-UX Spooler
Soren R. Olsen - Boeing Computer Services

2009 How Do You Know That Program Will Work?
Frederick W. Ochs, Jr. - ORBIT Software

2010 An Electronic Data Interchange Primer
Frederick W. Ochs, Jr. - ORBIT Software

2011 A Case Study: Network Design and Implementation
Ken Bare - Collier-Jackson, Inc.

2012 Writing UNIX-like Commands in the Bourne Shell
Craig P. Albrecht - Cray Research, Inc.

2013 An Inside Look at the UNIX Command NAWK An Interactive Investigation
Craig P. Albrecht - Cray Research, Inc.

2015 HP Client/Server Configurations
Greg Proulx - Oracle Corp.
2017 Understanding IP Addressing
J. W. Swearingen - The Apex Group, Inc.
2018 Convert That Old Application to X
James Langan - J.B. Langan & Associates, Inc.
2019 Connect Those Computers
James Langan - J.B. Langan & Associates, Inc.
2020 How to Migrate Forms to X11/Motif
Robert Combs - Combs International
2021 Is There a Workstation in Your Future?
Brian Nogar - Hewlett-Packard Co.
2022 Using Structured Analysis for Application Migration
Phil A. Beetley - Hewlett-Packard Co.
2023 Comprehensive Automated Storage Management for Networked Environments
Gottfried B. Bertram - Hewlett-Packard Co.
2024 Integrating Information with Object- Oriented Databases
Douglas Dedo - Hewlett-Packard Co.
2025 HP-UX System Management with SAM
Doug Drees - Hewlett-Packard Co.
2026 Benefiting from Object Technology in a Distributed Environment
Jeff Eastman - Hewlett-Packard Co.
2027 Security Aspects In A Backup Environment
Wolfgang Friedrich - Hewlett-Packard Co.
2028 Cutting the "Super" Hype--Evaluating a RISC Architecture
Linley Gwennap - Hewlett-Packard Co.
2029 Using Agent Concepts To Migrate To New Technologies
Edward Hansen - Hewlett Packard Co.
2030 An Introduction to HP-UX Security Issues and Controls
Chris Hauck - Hewlett-Packard Co.
2031 The Real Cost Savings in Managing Open Systems
Martin A. Kellner - Hewlett-Packard Co.
2032 Harnessing CASE for Your Move to Open Systems
Casey Lemus - Hewlett-Packard Co.
2033 System Management in Distributed Environments
Wolfgang A. Maier - Hewlett-Packard Co.
2034 How to Maximize Up Time in a UNIX Environment
Cecilia Margevicius - Hewlett-Packard Co.
2035 Comparing Multiprocessing Architectures
Dave Martinez - Hewlett-Packard Co.
2036 Where Are Your HP-UX System Resources Going?
Marty Poniatowski - Hewlett-Packard Co.
2038 Cooperative Computing Through Open Systems
Mikael Edholm - Hewlett-Packard Co.
2039 Hewlett-Packard Open View Vision
Steve Bunten - Hewlett-Packard Co.
2040 Creating Telephone Switch to Computer Applications
Bob Alley - Hewlett-Packard Co.

Index by Paper Number

2041 Managing Networked PCs
Michael Rafeld - Hewlett-Packard Co.

2042 ISDN Applications
Eric Grall - Hewlett-Packard Co.

2043 SCSI--A View From Both Ends of the Cable
Michael Rusnack - Hewlett-Packard Co.

2044 Performance Management Coexistence Living with MPE/iX and HP-UX: What Do I Need to Know
Rex Backman - Hewlett-Packard Co.

2046 Using Standard Tools To Build An Open, Client/Server Prototype
Bernard S. Hirsch - Hewlett-Packard Co.

2047 Using 4GLs for Application Development
Graham Masters - Hewlett Packard Co.

2048 Measuring and Modeling UNIX Systems
Dan Sternadel - Hewlett-Packard Co.

2049 Performance Is Easy
Joe Thomas - Hewlett Packard Co.

2050 Facing the Fears of Moving from Host Based to Client/Server Computing
Tony Jones - Hewlett-Packard Co.

2051 HP-UX and OSF/1 in the Commercial Market
Lisa Burns - Hewlett-Packard Co.

2052 Innovative Use of Client/Server Architecture in a System Support
Richard Lasslo - Hewlett-Packard Co.

2053 Enterprise Networking: Technology and its Impact on Business Applications
Sam Sudarsanam - Hewlett-Packard Co.

2056 From Downsizing to Rightsizing: A Guideline for Successful Information Technology Restructuring in the 90's
Feyzi Fatehi - Hewlett-Packard Co.

2057 Help! My Filesystem is Full and I Can't Get Up!
Bill Hassell - Hewlett-Packard Co.

3001 An Overview of Queue Network Modeling Capacity Planning
Robert Lund - Lund Performance Solutions

3002 How to Win at Your Next EDP Audit
Betsy Leight - Operations Control Systems

3003 Mainframe Software Management Techniques What Every HP 3000 User Should Know
Betsy Leight - Operations Control Systems

3004 Successful Project Methodology
Robert S. Dobis - Crowe, Chizek and Company

3005 Managing the HP Mainframe
Guy Smith - GS Communicatioins

3006 HP Quotes & Contracts 101
Michael Hornsby - Beechglen Development, Inc.

3007 Designing an EDI Audit and Control System
Kurt Nuehring - Outdoor Technologies Group

3008 Developing Quick Response EDI Software
Rod E. Keiser - Computer Management Solutions

3009 Are You My Priority
Jeff Odom - Bahlsen, Inc.

3010 How To Telecommunicate And Retain Your Sanity
Paul Edwards - Paul Edwards & Associates

3011 Managing a PowerHouse Shop
David G. Robinson - PowerSpec International

3012 Automating Computer Operations
Kirk E. Bedwell - Marsh Company

3013 Obtaining the Competitive Edge Through Automated Data Collection
Ray Agrusti - Eagle Consulting & Development Corp.

3014 Applying Mainframe Security Standards in the Minicomputer Environment
Lee Courtney - Monterey Software Group

3015 Negotiating Win-Win Agreements
Robert Andrews - Dynamic Information Systems Corp.

3016 Mainframe Downsizing As "Rightsizing"
David J. Rubinstein - Innovative Information Systems, Inc.

3017 Leadership is a Matter of Performance
J.B. Watterson - CDSI

3020 The Process Of Capacity Planning--A Case Study
Barney E. Green - CompuChem Laboratories

3021 The Experience of Relaxation
Margaret Brunner - Northern California Cancer Center

3022 In House Computing: Survival of the Fittest
Doug Samuels - Space Coast Credit Union

3023 Coping With The New I.S. Revolution-- What Every I.S. Executive Needs To Know
Roger Lawson - Proactive Systems Ltd.

3024 $pending To $ave On $oftware
Shelby Robert - Proactive Systems

3026 Managing Development Projects for New Technologies
Suzanne Harmon - Information Systems Consultant

3027 Transitioning Programmers from 3GL to 4GL
Suzanne Harmon - Information Systems Consultant

3028 Managing the Software Process & Outsourcing
Michael Mazza - Brown & Williamson Tabacco Corp.

3029 You, Yes You, Can Write Documentation! A DP Professional's Guide to Writing
Pamela Dickerson - ECI Computer, Inc.

3030 DP - Life Success Principles
Robert Lund - Lund Performance Solutions

3031 Maintaining a Quality Staff Starts During the Interview
Jeff Odom - Bahlsen, Inc.

3032 Software Aquisition--Getting It Off The Ground
Robert G. Boynton - Warn Industries

3033 Capacity Planning Part-Time
Javier Diaz - Goodyear Mexico

3034 Project Management, Begin By Sticking To The Basics
Jeff O'Donnell - DesignAlyze Corporate Consulting

3035 Salespeak--How to Understand It
Jeffry J. Franz - 21st Century Systems Group

3036 Help Desk Management Relieves the Monday Morning Blues
Kathy Spanel - GBS Consultants, Inc.

3037 Attributes of a First-Rate Software Support Application
Mike Butler - Dynamic Information Systems Corp.

3038 Telecommuting--The Future Has Arrived
Pamela Herbert - A.H. Custom Software, Inc.

3039 RF/DC and the HP 3000
Joe Howell - Professional Products

3040 Risk Taking... In Your Career and Personal Life
Kathy McKittrick - McKittrick Associates

3051 MORE Secrets of MPE V Tables: A Closer Look at Code Management Structure
Craig Nickerson - United Electric Controls Company

3052 Reload (Install) Cookbook on MPE XL Systems
Donald E. DeFreese - McDonnell Aircraft Company

3053 Quick Access to Archived Data for the HP3000 with Optical Technology
Husni S. Sayed - IEM, Inc.

3054 Electronic Draft Capture Interfacing For The HP3000
Gary L. Biggs - Micro Consulting, Inc.

3055 Popular Mass Storage: Optical Disk and Helical Scan Tape
Husni S. Sayed - IEM, Inc.

3056 Online Backup: The Method Behind the Magic
David Merit - ORBIT Software

3058 A Plan For Input and Output Queues
Terry Warns - WL Software

3059 Client/Server Architecture for MPE/XL- HP-UX Macintosh Windows
James F. Dowling - Bose Corporation

3061 Glance/XL Advanced Analysis
Michael Hornsby - Beechglen Development, Inc.

3062 How Developers Can Adopt Self-Describing Files
David Greer - Robelle Consulting Ltd.

3063 Integrating Voice Response Into An HP Environment
Will Strother - Computer Communications Specialists

3064 Using Terminal Emulation to Perform OLTP Benchmarks
Gary A. Showers - State Farm Insurance Companies

3101 The Warning Signs, a Pop Quiz on Quality
Bob Green - Robelle Consulting Ltd.

3102 An Update On Software Inspections Methods
Steven M. Cooper - Allegro Consultants, Inc.

3103 STREAMX: It's Not Just for Passwords Anymore!
Stephen S. Hammond - Association of American Medical Colleges

3104 User Interphase Prototyping For Free
Anthony C. Furnivall - SDL/Software, Inc.

3106 How To Write Quality Pascal Programs
Stan Sieler - Allegro Consultants, Inc.

3107 The Case For OOP
Linda A. Haake - Speedware Corp.

3108 The Object Is The Subject
Douglas Colter - Speedware Corp.

3110 Why Is "EDI Standard" An Oxymoron?
Trevor Richards - M.B. Foster Associates, Inc.

3112 Integrated EDI: Is It Possible?
Rod E. Keiser - Computer Management Solutions

3113 How To Laser Away Pre-Printed Forms
Jason Kent - Proactive Systems
3114 Incorporating Dynamic Electronic Forms Into HP 3000 Applications
Ross G. Hopmans - Brant Technologies, Inc.
3115 Application Installation
Frannie Casella - Northern Calif. Cancer Center
3116 Software Re-engineering: Teaching an Old Dog, New Tricks
Christopher Nelson - S/Wizardry, Ltd.
3117 Object-Oriented Methodology: The Adager Instance
Alfredo Rego - Adager
3122 Generic Routines in COBOL Using $DEFINE and.LEN.
Rick Hoover - CIV Software
3151 Database Freedom
Alfredo Rego - Adager
3152 Migration from TurboImage to ALLBASE
George B. Scott - Great Business Solutions, Inc.
3153 ALLBASE/SQL vs Image for Application Development
Nicholas J. Walsh, Jr. - Speedware Corp.
3154 Early Returns on Critical Item Update: You Be the Judge...
Jerry Fochtman - Exxon Chemical Company
3155 Using ORACLE to Access a TurboIMAGE Database
Kimble F. Ross - Database Consultants
3156 Validating Your Database--What You Don't Know Does Hurt You!
Shelby Robert - Proactive Systems
3157 SQL for the Curious
Dirk Huizenga - Dynamic Information Systems Corp.
3158 TurboIMAGE/Standard Interface to Third Party Indexing
Eric Savage - Dynamic Information Systems Corp.
3159 The Expanded IMAGE Retrieval Toolkit
Donna Bloechl - Dynamic Information Systems Corp.
3160 Moving From TurboIMAGE To Relational
Greg Proulx - Oracle Corp.
3162 Data Integrity From TurboIMAGE To Relational
Jo-Ning Ta - Oracle Corp.
3163 Understanding Relational DBMS
Jim Eduljee - Infocentre Corporation
3166 TurboIMAGE 4.0: Relational Access
Tim Joseph, Larry Boyd - Bradmark Technologies, Inc.
3169 Normalizing IMAGE in the 90's
Rick Hoover - CIV Software
3201 Systems Integration vs. Open Systems
Eric B. Threatt - Wesson, Taylor, Wells & Associates
3202 Open Systems
David Haberman - Speedware Corporation
3251 Using Powerhouse For Data Migration
Marilyn Petty Glick - Time Systems, Inc.
3252 IBM S/36 to HP 3000/900 Migration
Jamie Harwood - Great Western Carpet Cushion Co.

3253 Transporting an MPE Application to UNIX
Todd Saylor - Bi-Tech Software, Inc.
3255 IBM to HP Migration--Overview & Case Study
Mark L. Symonds - IISI
3256 Looking at UNIX Through MPE Eyes
Betsy Leight - Operations Control Systems
3302 Meeting the Network Challenge: Running Multiple Network Protocols on a PC
Randy Robinson - Walker Richer & Quinn
3303 Client Server Computing Taking the First Step
Doug Walker - Walker Richer & Quinn
3304 Client/Server Technologies in the 90's
David Haberman - Speedware Corporation
3306 What Does A Distributed System Give You?
Niket Patwardhan - Oracle Corporation
3308 ASCII or EBCDIC: Only Your Printer Knows for Sure
Erik J. Vistica - ROLM Systems
3309 Data Transfer With NRJE: Making Your Data Turn "Blue"
Erik J. Vistica - ROLM Systems
3310 Networking HP 3000's
Robert Carnagey - Spectra Physics Scanning Systems
3351 Object Oriented Applications
David Johnson - Johnson Computer Software Team
3352 Client/Server Computing Under MPE
Joseph Geiser - Insurance Data Processing, Inc.
3354 A Client/Server Prototype for IMAGE
Ross G. Hopmans - Brant Technologies, Inc.
3355 Networking Personal Computers
Kirk E. Bedwell - Marsh Company
3356 Winning Strategies for PC Managers
Brad Feazell - Dril-Quip, Inc.
3357 The HP 3000 and PC Application Software Together at Last!
Patricia I. Loo - EG&G Idaho
3358 Work Flow Automation via NewWave
Sam Fine - The Apex Group Inc.
3359 Using Windows and Networks to Increase Productivity and Customer Service
Kevin Wilhelmsen - Space Coast Credit Union
3360 Utilizing Graphical User Interfaces in NewWave
Leonard Block - The Apex Group
3361 Personal Computer Management, Audit and Control
Dirk Huizenga - Dynamic Information Systems Corp.
3363 Using Dynamic Data Exchange to Integrate Host and Windows Applications
Brian Hargus - Walker Richer & Quinn
3364 Client/Server Technology Made Easy Using PPL
Tim Klooster - Dynamic Information Systems Corp.
3801 HP 3000 Open Distributed OLTP
Rick Bartlett - Hewlett-Packard Co.
3802 Configuring Your Disks for High Availability
Dave Beasley - Hewlett-Packard Co.

3804 MPE/iX High End Kernel Algorithms for OLTP Performance
Hung Nguyen, Korak Mitra - Hewlett-Packard Co.
3805 TurboIMAGE Applications In The 90's and Beyond
Randolph Diepeveen - Hewlett-Packard Co.
3806 Key Buzzwords for the 1990's: Open, Inter-operability, POSIX and DCE
Joseph Feiner - Hewlett-Packard Co.
3807 What Does Migrating to ALLBASE/SQL Mean For You?
Janice Hodor - Hewlett-Packard Co.
3808 Basic Dump Analysis for System Managers
David Kohl - Hewlett-Packard Co.
3809 POSIX Interface for MPE/iX
Rajesh Lalwani - Hewlett-Packard Co.
3810 AllBASE/Turbo Connect: The Write Stuff
Randolph Diepeveen - Hewlett-Packard Co.
3811 Introduction to MPE/iX Architecture
Tom Shem - Hewlett-Packard Co.
3812 Rapid Development of Client/Server Applications Using RPC
Charles Knouse - Hewlett-Packard Co.
3851 Caveat Emptor: What Buyers and Developers of Client/Server Applications Need to Know
Mike Bailey - Hewlett-Packard Co.
3852 Stored Procedures and Rules in ALLBASE/SQL
Amelia Carlson, et al - Hewlett-Packard Co.
3853 Improving the Bottom Line With Object-Oriented Databases
Douglas Dedo - Hewlett-Packard Co.
3854 HP 3000 Open Systems and Client/Server Computing
Christine Fronczak - Hewlett-Packard Co.
3855 Object-Oriented Technologies: A Management Overview
Orly Larson - Hewlett-Packard Co.
3856 Managing IT/MIS in the 90's
John Podkomorski - Hewlett-Packard Co.
3857 Implementing a Client/Server Application
Stayton Chock - Hewlett-Packard Co.
3858 HP 3000 Multiprocessing: Synchronization and its Performance Implications
Kimberly Rogers, et al - Hewlett-Packard Co.
3859 Software Resiliency
Kendall Sutton, Jessy Hsu - Hewlett-Packard Co.
3860 HP: The Best Alternative to Mainframe Computing (.... and getting there)
Robert Winter - Hewlett-Packard Co.
3862 Don't Put Your Career on Hold
John Picket - Hewlett-Packard Co.
3863 HP X.25 Access Products: Providing Open Systems Solutions
Jean-Claude Viollier - Hewlett-Packard Co.
3864 Object Oriented COBOL: The Next Generation
Megan Adams, Dmitry Lenkov - Hewlett-Packard Co.
3865 New Directions for COBOL --Where will COBOL 2000 Take Us?
Julia Rodriguez - Hewlett-Packard Co.
3867 The SLA Cookbook: A Methodology for Managing Computer System Performance
Doug McBride - Hewlett-Packard Co.

3868 Enterprise Performance Management - by exception only
Wayne Morris - Hewlett-Packard Co.

3869 Quantifying Investments in Technology
Mike Overly - Hewlett-Packard Co.

3870 What's New and Improved with MPE/V
Robert Ross - Hewlett-Packard Co.

3871 O.S.E., a framework for an Information Technology Strategy
Andre Van Aken - Hewlett-Packard Co.

3872 Managing Your Help Desk in the 1990s
Tamara Gordon - Hewlett-Packard Co.

3873 A User's Experience of HP 3000 and DTC In a Multivendor Environment
Juha Hakala - Automation Unit of Finnish Libraries

3874 Disk Arrays
Ed Pavlinik - Hewlett-Packard Co.

Index by Author

Adams, Megan, Lenkov, Dmitry Object Oriented COBOL: The Next Generation
3864, Hewlett-Packard Co.

Agrusti, Ray Obtaining the Competitive Edge Through Automated Data Collection
3013, Eagle Consulting & Development Corp.

Albrecht, Craig P. Writing UNIX-like Commands in the Bourne Shell
2012, Cray Research, Inc.

Albrecht, Craig P. An Inside Look at the UNIX Command NAWK: An Interactive Investigation
2013, Cray Research, Inc.

Alley, Bob Creating Telephone Switch to Computer Applications
2040, Hewlett-Packard Co.

Andrews, Robert Negotiating Win-Win Agreements
3015, Dynamic Information Systems Corp.

Arya, Yagya Dutt Computer Access and Resource Monitoring
2002, Indian Institute of Technology

Backman, Rex Performance Management Coexistence Living with MPE/iX and HP-UX: What Do I Need to Know
2044, Hewlett-Packard Co.

Bailey, Mike Caveat Emptor: What Buyers and Developers of Client/Server Applications Need to Know
3851, Hewlett-Packard Co.

Bare, Ken A Case Study: Network Design and Implementation
2011, Collier-Jackson, Inc.

Bartlett, Rick HP 3000 Open Distributed OLTP
3801, Hewlett-Packard Co.

Beasley, Dave Configuring Your Disks for High Availability
3802, Hewlett-Packard Co.

Bedwell, Kirk E. Networking Personal Computers
3355, Marsh Company

Bedwell, Kirk E. Automating Computer Operations
3012, Marsh Company

Beetley, Phil A. Using Structured Analysis for Application Migration
2022, Hewlett-Packard Co.

Bertram, Gottfried B. Comprehensive Automated Storage Management for Networked Environments
2023, Hewlett-Packard Co.

Biggs, Gary L. Electronic Draft Capture Interfacing For The HP3000
3054, Micro Consulting, Inc.

Block, Leonard Utilizing Graphical User Interfaces in NewWave
3360, The Apex Group

Bloechl, Donna The Expanded IMAGE Retrieval Toolkit
3159, Dynamic Information Systems Corp.

Boynton, Robert G. Software Aquisition--Getting It Off The Ground
3032, Warn Industries

Brill, Lincoln Lilly Fermentation Process Control System
1002, Eli Lilly and Co.

Brunner, Margaret The Experience of Relaxation
3021, Northern California Cancer Center

Bunten, Steve Hewlett-Packard Open View Vision
2039, Hewlett-Packard Co.

Burns, Lisa HP-UX and OSF/1 in the Commercial Market
2051, Hewlett-Packard Co.

Butler, Mike — Attributes of a First-Rate Software Support Application
3037, Dynamic Information Systems Corp.

Carlson, Amelia, et al — Stored Procedures and Rules in ALLBASE/SQL
3852, Hewlett-Packard Co.

Carnagey, Robert — Networking HP 3000's
3310, Spectra Physics Scanning Systems

Casella, Frannie — Application Installation
3115, Northern Calif. Cancer Center

Chock, Stayton — Implementing a Client/Server Application
3857, Hewlett-Packard Co.

Colter, Douglas — The Object Is The Subject
3108, Speedware Corp.

Combs, Robert — Performance Comparison of the A990
1001, Combs International

Combs, Robert — How to Migrate Forms to X11/Motif
2020, Combs International

Cooper, Steven M. — An Update On Software Inspections Methods
3102, Allegro Consultants, Inc.

Courtney, Lee — Applying Mainframe Security Standards in the Minicomputer Environment
3014, Monterey Software Group

DeFreese, Donald E. — Reload (Install) Cookbook on MPE XL Systems
3052, McDonnell Aircraft Company

Dedo, Douglas — Improving the Bottom Line With Object-Oriented Databases
3853, Hewlett-Packard Co.

Dedo, Douglas — Integrating Information with Object- Oriented Databases
2024, Hewlett-Packard Co.

Diaz, Javier — Capacity Planning Part-Time
3033, Goodyear Mexico

Dickerson, Pamela — You, Yes You, Can Write Documentation! A DP Professional's Guide to Writing
3029, ECI Computer, Inc.

Diepeveen, Randolph — TurboIMAGE Applications In The 90's and Beyond
3805, Hewlett-Packard Co.

Diepeveen, Randolph — AllBASE/Turbo Connect: The Write Stuff
3810, Hewlett-Packard Co.

Dobis, Robert S. — Successful Project Methodology
3004, Crowe, Chizek and Company

Dowling, James F. — Client/Server Architecture for MPE/XL- HP-UX Macintosh Windows
3059, Bose Corporation

Drees, Doug — HP-UX System Management with SAM
2025, Hewlett-Packard Co.

Eastman, Jeff — Benefiting from Object Technology in a Distributed Environment
2026, Hewlett-Packard Co.

Edholm, Mikael — Cooperative Computing Through Open Systems
2038, Hewlett-Packard Co.

Eduljee, Jim — Understanding Relational DBMS
3163, Infocentre Corporation

Edwards, Paul — How To Telecommunicate And Retain Your Sanity
3010, Paul Edwards & Associates

Fatehi, Feyzi 2056, Hewlett-Packard Co.	From Downsizing to Rightsizing: A Guideline for Successful Information Technology Restructuring in the 90's
Feazell, Brad 3356, Dril-Quip, Inc.	Winning Strategies for PC Managers
Feiner, Joseph 3806, Hewlett-Packard Co.	Key Buzzwords for the 1990's: Open, Inter-operability, POSIX and DCE
Fine, Sam 3358, The Apex Group Inc.	Work Flow Automation via NewWave
Fochtman, Jerry 3154, Exxon Chemical Company	Early Returns on Critical Item Update: You Be the Judge...
Franz, Jeffry J. 3035, 21st Century Systems Group	Salespeak--How to Understand It
Frazier, Jimmy W. 2005, Hoechst Celanese Corporation -	LAN Manager - A Case Study
Friedrich, Wolfgang 2027, Hewlett-Packard Co.	Security Aspects In A Backup Environment
Fronczak, Christine 3854, Hewlett-Packard Co.	HP 3000 Open Systems and Client/Server Computing
Furnivall, Anthony C. 3104, SDL/Software, Inc.	User Interphase Prototyping For Free
Gan, Gary 1010, Hewlett-Packard Co.	Symbolic Links on RTE-A/VC PLUS
Geiser, Joseph 3352, Insurance Data Processing, Inc.	Client/Server Computing Under MPE
Glick, Marilyn Petty 3251, Time Systems, Inc.	Using Powerhouse For Data Migration
Gordon, Tamara 3872, Hewlett-Packard Co.	Managing Your Help Desk in the 1990s
Grall, Eric 2042, Hewlett-Packard Co.	ISDN Applications
Green, Barney E. 3020, CompuChem Laboratories	The Process Of Capacity Planning--A Case Study
Green, Bob 3101, Robelle Consulting Ltd.	The Warning Signs, a Pop Quiz on Quality
Greer, David 3062, Robelle Consulting Ltd.	How Developers Can Adopt Self-Describing Files
Guy, Ken 2004, MICOM Communications Corp.	Changing The Rules for Data/Voice Integration
Gwennap, Linley 2028, Hewlett-Packard Co.	Cutting the "Super" Hype--Evaluating a RISC Architecture
Haake, Linda A. 3107, Speedware Corp.	The Case For OOP
Haberman, David 3304, Speedware Corporation	Client/Server Technologies in the 90's
Haberman, David 3202, Speedware Corporation	Open Systems
Hammond, Stephen S. 3103, Association of American Medical Colleges	STREAMX: It's Not Just for Passwords Anymore!

Hansen, Edward Using Agent Concepts To Migrate To New Technologies
2029, Hewlett Packard Co.
Hargus, Brian Using Dynamic Data Exchange to Integrate Host and Windows Applications
3363, Walker Richer & Quinn, Inc.
Harmon, Suzanne Managing Development Projects for New Technologies
3026,
Harmon, Suzanne Transitioning Programmers from 3GL to 4GL
3027,
Harwood, Jamie IBM S/36 to HP 3000/900 Migration
3252, Great Western Carpet Cushion Co.
Hassell, Bill Help! My Filesystem is Full and I Can't Get Up!
2057, Hewlett-Packard Co.
Hauck, Chris An Introduction to HP-UX Security Issues and Controls
2030, Hewlett-Packard Co.
Heller, Esther, Carroll, Rebecca What to Expect From RTE 6.0
1005, Hewlett Packard Co.
Herbert, Pamela Telecommuting--The Future Has Arrived
3038, A.H. Custom Software, Inc.
Hirsch, Bernard S. Using Standard Tools To Build An Open, Client/Server Prototype
2046, Hewlett-Packard Co.
Hodor, Janice What Does Migrating to ALLBASE/SQL Mean For You?
3807, Hewlett-Packard Co.
Hoover, Rick Normalizing IMAGE in the 90's
3169, CIV Software
Hoover, Rick Generic Routines in COBOL Using $DEFINE and.LEN.
3122, CIV Software
Hopmans, Ross G. A Client/Server Prototype for IMAGE
3354, Brant Technologies, Inc.
Hopmans, Ross G. Incorporating Dynamic Electronic Forms Into HP 3000 Applications
3114, Brant Technologies, Inc.
Hornsby, Michael HP Quotes & Contracts 101
3006, Beechglen Development, Inc.
Hornsby, Michael Glance/XL Advanced Analysis
3061, Beechglen Development, Inc.
Howell, Joe RF/DC and the HP 3000
3039, Professional Products
Huizenga, Dirk SQL for the Curious
3157, Dynamic Information Systems Corp.
Huizenga, Dirk Personal Computer Management, Audit and Control
3361, Dynamic Information Systems Corp.
Hukala, Juha A User's Experience of HP 3000 and DTC in a Multivendor Environment
3873, Automation Unit of Finnish Libraries
Jones, Tony Facing the Fears of Moving from Host Based to Client/Server Computing
2050, Hewlett-Packard Co.
Johnson, David Object Oriented Applications
3351, Johnson Computer Software Team
Joseph, Tim, Boyd, Larry TurboIMAGE 4.0: Relational Access
3166, Bradmark Technologies, Inc.

Keiser, Rod E. Developing Quick Response EDI Software
3008, Computer Management Solutions
Keiser, Rod E. Integrated EDI: Is It Possible?
3112, Computer Management Solutions
Kellner, Martin A. The Real Cost Savings in Managing Open Systems
2031, Hewlett-Packard Co.
Kent, Jason How To Laser Away Pre-Printed Forms
3113, Proactive Systems
King, Wendy Using BSD IPC to Synchronize RTE & HP-UX System Times
1014, US Naval Observatory
Klooster, Tim Client/Server Technology Made Easy Using PPL
3364, Dynamic Information Systems Corp.
Knouse, Charles Rapid Development of Client/Server Applications Using RPC
3812, Hewlett-Packard Co.
Kohl, David Basic Dump Analysis for System Managers
3808, Hewlett-Packard Co.
Lalwani, Rajesh POSIX Interface for MPE/iX
3809, Hewlett-Packard Co.
Langan, James Convert That Old Application to X
2018, J.B. Langan & Associates, Inc.
Langan, James Connect Those Computers
2019, J.B. Langan & Associates, Inc.
Larson, Orly Object-Oriented Technologies: A Management Overview
3855, Hewlett-Packard Co.
Lasslo, Richard Innovative Use of Client/Server Architecture in a System Support
2052, Hewlett-Packard Co.
Lawson, Roger Coping With The New I.S. Revolution-- What Every I.S. Executive Needs To Know
3023, Proactive Systems Ltd.
Leight, Betsy Mainframe Software Management Techniques What Every HP 3000 User Should Know
3003, Operations Control Systems
Leight, Betsy How to Win at Your Next EDP Audit
3002, Operations Control Systems
Leight, Betsy Looking at UNIX Through MPE Eyes
3256, Operations Control Systems
Lemus, Casey Harnessing CASE for Your Move to Open Systems
2032,
Loo, Patricia I. The HP 3000 and PC Application Software Together at Last!
3357, EG&G Idaho
Lund, Robert An Overview of Queue Network Modeling Capacity Planning
3001, Lund Performance Solutions
Lund, Robert DP - Life Success Principles
3030, Lund Performance Solutions
Maier, Wolfgang A. System Management in Distributed Environments
2033, Hewlett-Packard Co.
Margevicius, Cecilia How to Maximize Up Time in a UNIX Environment
2034, Hewlett-Packard Co.
Marshall, Gene SoftBench Link/1000 Encapsulation: A State-of-the-Art CASE Tool
1007, Hewlett-Packard Co.

Martinez, Dave — Comparing Multiprocessing Architectures
2035, Hewlett-Packard Co.

Masters, Graham — Using 4GLs for Application Development
2047, Hewlett Packard Co.

Mazza, Michael — Managing the Software Process & Outsourcing
3028, Brown & Williamson Tabacco Corp.

McBride, Doug — The SLA Cookbook: A Methodology for Managing Computer System Performance
3867, Hewlett-Packard Co.

McCabe, John — Migrating to "OPEN"
2007, Lever Brothers Company

McKittrick, Kathy — Risk Taking... In Your Career and Personal Life
3040, McKittrick Associates

Medlicott, David — Interrupts and RTE
1006, Hewlett Packard

Merit, David — Online Backup: The Method Behind the Magic
3056, ORBIT Software

Morris, Wayne — Enterprise Performance Management - by exception only
3868, Hewlett-Packard Co.

Nelson, Christopher — Software Re-engineering: Teaching an Old Dog, New Tricks
3116, S/Wizardry, Ltd.

Nguyen, Hung, Mitra, Korak — MPE/iX High End Kernel Algorithms for OLTP Performance
3804, Hewlett-Packard Co.

Nickerson, Craig — MORE Secrets of MPE V Tables: A Closer Look at Code Management Structure
3051, United Electric Controls Company

Nogar, Brian — Is There a Workstation in Your Future?
2021, Hewlett-Packard Co.

Nuehring, Kurt — Designing an EDI Audit and Control System
3007, Outdoor Technologies Group

O'Donnell, Jeff — Project Management, Begin By Sticking To The Basics
3034, DesignAlyze Corporate Consulting

Ochs, Jr., Frederick W. — How Do You Know That Program Will Work?
2009, ORBIT Software

Ochs, Jr., Frederick W. — An Electronic Data Interchange Primer
2010, ORBIT Software

Odom, Jeff — Are You My Priority
3009, Bahlsen, Inc.

Odom, Jeff — Maintaining a Quality Staff Starts During the Interview
3031, Bahlsen, Inc.

Olsen, Soren R. — Defining HP 3000 Printers in HP-UX Spooler
2008, Boeing Computer Services

Overly, Mike — Quantifying Investments in Technology
3869, Hewlett-Packard Co.

Patwardhan, Niket — What Does A Distributed System Give You?
3306, Oracle Corporation

Pavlinik, Ed — Disk Arrays
3874, Hewlett-Packard Co.

Picket, John — Don't Put Your Career on Hold
3862, Hewlett-Packard Co.

Podkomorski, John — Managing IT/MIS in the 90's
3856, Hewlett-Packard Co.

Poniatowski, Marty — Where Are Your HP-UX System Resources Going?
2036, Hewlett-Packard Co.

Poyner, Todd — What's in a Namesaver?
1009, Hewlett-Packard Co

Preece, Richard — Automatic Backup, Archival And Retrieval In A Heterogeneous Network
2003, Hi-Comp GmbH

Proulx, Greg — Moving From TurboIMAGE To Relational
3160, Oracle Corp.

Proulx, Greg — HP Client/Server Configurations
2015, Oracle Corp.

Rafeld, Michael — Managing Networked PCs
2041, Hewlett-Packard Co.

Rego, Alfredo — Database Freedom
3151, Adager

Rego, Alfredo — Object-Oriented Methodology: The Adager Instance
3117, Adager

Richards, Trevor — Why Is "EDI Standard" An Oxymoron?
3110, M.B. Foster Associates, Inc.

Robert, Shelby — Validating Your Database--What You Don't Know Does Hurt You!
3156, Proactive Systems

Robert, Shelby — $pending To $ave On $oftware
3024, Proactive Systems

Robinson, David G. — Managing a PowerHouse Shop
3011, PowerSpec International

Robinson, Randy — Meeting the Network Challenge: Running Multiple Network Protocols on a PC
3302, Walker Richer & Quinn, Inc.

Rodriguez, Julia — New Directions for COBOL --Where will COBOL 2000 Take Us?
3865, Hewlett-Packard Co.

Rogers, Kimberly, et al — HP 3000 Multiprocessing: Synchronization and its Performance Implications
3858, Hewlett-Packard Co.

Ross, Kimble F. — Using ORACLE to Access a TurboIMAGE Database
3155, Database Consultants

Ross, Robert — What's New and Improved with MPE/V
3870, Hewlett-Packard Co.

Rubinstein, David J. — Mainframe Downsizing As "Rightsizing"
3016, Innovative Information Systems, Inc.

Rusnack, Michael — SCSI--A View From Both Ends of the Cable
2043, Hewlett-Packard Co.

Samuels, Doug — In House Computing: Survival of the Fittest
3022, Space Coast Credit Union

Savage, Eric — TurboIMAGE/Standard Interface to Third Party Indexing
3158, Dynamic Information Systems Corp.

Sayed, Husni S. — Quick Access to Archived Data for the HP3000 with Optical Technology
3053, IEM, Inc.

Sayed, Husni S. — Popular Mass Storage: Optical Disk and Helical Scan Tape
3055, IEM, Inc.

Saylor, Todd — Transporting an MPE Application to UNIX
3253, Bi-Tech Software, Inc.

Scott, George B. — Migration from TurboImage to ALLBASE
3152, Great Business Solutions, Inc.

Shem, Tom — Introduction to MPE/iX Architecture
3811, Hewlett-Packard Co.

Showers, Gary A. — Using Terminal Emulation to Perform OLTP Benchmarks
3064, State Farm Insurance Companies

Sieler, Stan — How To Write Quality Pascal Programs
3106, Allegro Consultants, Inc.

Smith, Guy — Managing the HP Mainframe
3005, GS Communicatioins

Spanel, Kathy — Help Desk Management Relieves the Monday Morning Blues
3036, GBS Consultants, Inc.

Sternadel, Dan — Measuring and Modeling UNIX Systems
2048, Hewlett-Packard Co.

Sterne, Jim — Evolution of Software Configuration Management
2001, Softool Corporation

Strother, Will — Integrating Voice Response Into An HP Environment
3063, Computer Communications Specialists

Sudarsanam, Sam — Enterprise Networking: Technology and its Impact on Business Applications
2053, Hewlett-Packard Co.

Sutton, Kendall, Hsu, Jessy — Software Resiliency
3859, Hewlett-Packard Co.

Swearingen, J. W. — Understanding IP Addressing
2017, The Apex Group, Inc.

Symonds, Mark L. — IBM to HP Migration--Overview & Case Study
3255, IISI

Ta, Jo-Ning — Data Integrity From TurboIMAGE To Relational
3162, Oracle Corp.

Thomas, Joe — Performance Is Easy
2049, Hewlett Packard

Threatt, Eric B. — Systems Integration vs. Open Systems
3201, Wesson, Taylor, Wells & Associates

Tibbetts, Alan — Advanced HP 1000 Assembly Language: Writing CDS Code
1013, Consultant

Van Aken, Andre — O.S.E., a framework for an Information Technology Strategy
3871, Hewlett-Packard Co.

Viallefont, P. — Portability Needs Under HP-UX Application
2006, C.N.E.S.

Viollier, Jean-Claude — HP X.25 Access Products: Providing Open Systems Solutions
3863, Hewlett-Packard Co.

Vistica, Erik J. — ASCII or EBCDIC: Only Your Printer Knows for Sure
3308, ROLM Systems

Vistica, Erik J. — Data Transfer With NRJE: Making Your Data Turn "Blue"
3309, ROLM Systems

Walker, Doug — Client Server Computing Taking the First Step
3303, Walker Richer & Quinn

Walsh, Jr., Nicholas J. ALLBASE/SQL vs Image for Application Development
3153, Speedware Corp.
Warns, Terry A Plan For Input and Output Queues
3058, WL Software
Watterson, J.B. Leadership is a Matter of Performance
3017, CDSI
Wilhelmsen, Kevin Using Windows and Networks to Increase Productivity and Customer Service
3359, Space Coast Credit Union
FaWinter, Robert HP: The Best Alternative to Mainframe Computing (.... and getting there)
3860, Hewlett-Packard Co.
Wright, Don Using the Mouse for Screen Data Entry On the HP 1000
1004, Interactive Computer Technology
Wright, Don An IMAGE-II Data Extraction Accelerator
1003, Interactive Computer Technology

RTE

1001 Performance Comparison of the A990
Robert Combs - Combs International
1002 Lilly Fermentation Process Control System
Lincoln Brill - Eli Lilly and Co.
1003 An IMAGE-II Data Extraction Accelerator
Don Wright - Interactive Computer Technology
1004 Using the Mouse for Screen Data Entry On the HP 1000
Don Wright - Interactive Computer Technology
1005 What to Expect From RTE 6.0
Esther Heller, Rebecca Carroll - Hewlett Packard Co.
1006 Interrupts and RTE
David Medlicott - Hewlett Packard Co.
1007 SoftBench Link/1000 Encapsulation: A State-of-the-Art CASE Tool
Gene Marshall - Hewlett-Packard Co.
1009 What's in a Namesaver?
Todd Poyner - Hewlett-Packard Co
1010 Symbolic Links on RTE-A /VC PLUS
Gary Gan - Hewlett-Packard Co.
1013 Advanced HP 1000 Assembly Language: Writing CDS Code
Alan Tibbetts - Consultant
1014 Using BSD IPC to Synchronize RTE & HP-UX System Times
Wendy King - US Naval Observatory
2041 Managing Networked PCs
Michael Rafeld - Hewlett-Packard Co.
2057 Help! My Filesystem is Full and I Can't Get Up!
Bill Hassell - Hewlett-Packard Co.

HP-UX

2001 Evolution of Software Configuration Management
Jim Sterne - Softool Corporation
2002 Computer Access and Resource Monitoring
Yagya Dutt Arya - Indian Institute of Technology
2003 Automatic Backup, Archival And Retrieval In A Heterogeneous Network
Richard Preece - Hi-Comp GmbH
2004 Changing The Rules for Data/Voice Integration
Ken Guy - MICOM Communications Corp.
2005 LAN Manager - A Case Study
Jimmy W. Frazier - Hoechst Celanese Corporation
2006 Portability Needs Under HP-UX Application
P. Viallefont - C.N.E.S.
2007 Migrating to "OPEN"
John McCabe - Lever Brothers Company
2008 Defining HP 3000 Printers in HP-UX Spooler
Soren R. Olsen - Boeing Computer Services

2009 How Do You Know That Program Will Work?
Frederick W. Ochs, Jr. - ORBIT Software

2010 An Electronic Data Interchange Primer
Frederick W. Ochs, Jr. - ORBIT Software

2011 A Case Study: Network Design and Implementation
Ken Bare - Collier-Jackson, Inc.

2012 Writing UNIX-like Commands in the Bourne Shell
Craig P. Albrecht - Cray Research, Inc.

2013 An Inside Look at the UNIX Command NAWK An Interactive Investigation
Craig P. Albrecht - Cray Research, Inc.

2015 HP Client/Server Configurations
Greg Proulx - Oracle Corp.

2017 Understanding IP Addressing
J. W. Swearingen - The Apex Group, Inc.

2018 Convert That Old Application to X
James Langan - J.B. Langan & Associates, Inc.

2019 Connect Those Computers
James Langan - J.B. Langan & Associates, Inc.

2020 How to Migrate Forms to X11/Motif
Robert Combs - Combs International

2021 Is There a Workstation in Your Future?
Brian Nogar - Hewlett-Packard Co.

2022 Using Structured Analysis for Application Migration
Phil A. Beetley - Hewlett-Packard Co.

2023 Comprehensive Automated Storage Management for Networked Environments
Gottfried B. Bertram - Hewlett-Packard Co.

2024 Integrating Information with Object- Oriented Databases
Douglas Dedo - Hewlett-Packard Co.

2025 HP-UX System Management with SAM
Doug Drees - Hewlett-Packard Co.

2026 Benefiting from Object Technology in a Distributed Environment
Jeff Eastman - Hewlett-Packard Co.

2027 Security Aspects In A Backup Environment
Wolfgang Friedrich - Hewlett-Packard Co.

2028 Cutting the "Super" Hype--Evaluating a RISC Architecture
Linley Gwennap - Hewlett-Packard Co.

2029 Using Agent Concepts To Migrate To New Technologies
Edward Hansen - Hewlett Packard Co.

2030 An Introduction to HP-UX Security Issues and Controls
Chris Hauck - Hewlett-Packard Co.

2031 The Real Cost Savings in Managing Open Systems
Martin A. Kellner - Hewlett-Packard Co.

2032 Harnessing CASE for Your Move to Open Systems
Casey Lemus -

2033 System Management in Distributed Environments
Wolfgang A. Maier - Hewlett-Packard Co.

2034 How to Maximize Up Time in a UNIX Environment
Cecilia Margevicius - Hewlett-Packard Co.

2035 Comparing Multiprocessing Architectures
Dave Martinez - Hewlett-Packard Co.

2036 Where Are Your HP-UX System Resources Going?
Marty Poniatowski - Hewlett-Packard Co.

2038 Cooperative Computing Through Open Systems
Mikael Edholm - Hewlett-Packard Co.

2039 Hewlett-Packard Open View Vision
Steve Bunten - Hewlett-Packard Co.

2040 Creating Telephone Switch to Computer Applications
Bob Alley - Hewlett-Packard Co.

2042 ISDN Applications
Eric Grall - Hewlett-Packard Co.

2043 SCSI--A View From Both Ends of the Cable
Michael Rusnack - Hewlett-Packard Co.

2044 Performance Management Coexistence Living with MPE/iX and HP-UX: What Do I Need to Know
Rex Backman - Hewlett-Packard Co.

2046 Using Standard Tools To Build An Open, Client/Server Prototype
Bernard S. Hirsch - Hewlett-Packard Co.

2047 Using 4GLs for Application Development
Graham Masters - Hewlett Packard

2048 Measuring and Modeling UNIX Systems
Dan Sternadel - Hewlett-Packard Co.

2049 Performance Is Easy
Joe Thomas - Hewlett Packard Co.

2050 Facing the Fears of Moving from Host Based to Client/Server Computing
Tony Jones -

2051 HP-UX and OSF/1 in the Commercial Market
Lisa Burns - Hewlett-Packard Co.

2052 Innovative Use of Client/Server Architecture in a System Support
Richard Lasslo - Hewlett-Packard Co.

2053 Enterprise Networking: Technology and its Impact on Business Applications
Sam Sudarsanam - Hewlett-Packard Co.

2056 From Downsizing to Rightsizing: A Guideline for Successful Information Technology Restructuring in the 90's
Feyzi Fatehi - Hewlett-Packard Co.

MPE

3001 An Overview of Queue Network Modeling Capacity Planning
Robert Lund - Lund Performance Solutions
3002 How to Win at Your Next EDP Audit
Betsy Leight - Operations Control Systems
3003 Mainframe Software Management Techniques What Every HP 3000 User Should Know
Betsy Leight - Operations Control Systems
3004 Successful Project Methodology
Robert S. Dobis - Crowe, Chizek and Company
3005 Managing the HP Mainframe
Guy Smith - GS Communicatioins
3006 HP Quotes & Contracts 101
Michael Hornsby - Beechglen Development, Inc.
3007 Designing an EDI Audit and Control System
Kurt Nuehring - Outdoor Technologies Group
3008 Developing Quick Response EDI Software
Rod E. Keiser - Computer Management Solutions
3009 Are You My Priority
Jeff Odom - Bahlsen, Inc.
3010 How To Telecommunicate And Retain Your Sanity
Paul Edwards - Paul Edwards & Associates
3011 Managing a PowerHouse Shop
David G. Robinson - PowerSpec International
3012 Automating Computer Operations
Kirk E. Bedwell - Marsh Company
3013 Obtaining the Competitive Edge Through Automated Data Collection
Ray Agrusti - Eagle Consulting & Development Corp.
3014 Applying Mainframe Security Standards in the Minicomputer Environment
Lee Courtney - Monterey Software Group
3015 Negotiating Win-Win Agreements
Robert Andrews - Dynamic Information Systems Corp.
3016 Mainframe Downsizing As "Rightsizing"
David J. Rubinstein - Innovative Information Systems, Inc.
3017 Leadership is a Matter of Performance
J.B. Watterson -CDSI
3020 The Process Of Capacity Planning--A Case Study
Barney E. Green - CompuChem Laboratories
3021 The Experience of Relaxation
Margaret Brunner - Northern California Cancer Center
3022 In House Computing: Survival of the Fittest
Doug Samuels - Space Coast Credit Union
3023 Coping With The New I.S. Revolution-- What Every I.S. Executive Needs To Know
Roger Lawson - Proactive Systems Ltd.
3024 $pending To $ave On $oftware
Shelby Robert - Proactive Systems
3026 Managing Development Projects for New Technologies
Suzanne Harmon - Information Systems Consultant

3027 Transitioning Programmers from 3GL to 4GL
Suzanne Harmon - Information Systems Consultant

3028 Managing the Software Process & Outsourcing
Michael Mazza - Brown & Williamson Tabacco Corp.

3029 You, Yes You, Can Write Documentation! A DP Professional's Guide to Writing
Pamela Dickerson - ECI Computer, Inc.

3030 DP - Life Success Principles
Robert Lund - Lund Performance Solutions

3031 Maintaining a Quality Staff Starts During the Interview
Jeff Odom - Bahlsen, Inc.

3032 Software Aquisition--Getting It Off The Ground
Robert G. Boynton - Warn Industries

3033 Capacity Planning Part-Time
Javier Diaz - Goodyear Mexico

3034 Project Management, Begin By Sticking To The Basics
Jeff O'Donnell - DesignAlyze Corporate Consulting

3035 Salespeak--How to Understand It
Jeffry J. Franz - 21st Century Systems Group

3036 Help Desk Management Relieves the Monday Morning Blues
Kathy Spanel - GBS Consultants, Inc.

3037 Attributes of a First-Rate Software Support Application
Mike Butler - Dynamic Information Systems Corp.

3038 Telecommuting--The Future Has Arrived
Pamela H. Bristow - A.H. Custom Software, Inc.

3039 RF/DC and the HP 3000
Joe Howell - Professional Products

3040 Risk Taking... In Your Career and Personal Life
Kathy McKittrick - McKittrick Associates

3051 MORE Secrets of MPE V Tables: A Closer Look at Code Management Structure
Craig Nickerson - United Electric Controls Company

3052 Reload (Install) Cookbook on MPE XL Systems
Donald E. DeFreese - McDonnell Aircraft Company

3053 Quick Access to Archived Data for the HP3000 with Optical Technology
Husni S. Sayed - IEM, Inc.

3054 Electronic Draft Capture Interfacing For The HP3000
Gary L. Biggs - Micro Consulting, Inc.

3055 Popular Mass Storage: Optical Disk and Helical Scan Tape
Husni S. Sayed - IEM, Inc.

3056 Online Backup: The Method Behind the Magic
David Merit - ORBIT Software

3058 A Plan For Input and Output Queues
Terry Warns - WL Software

3059 Client/Server Architecture for MPE/XL- HP-UX Macintosh Windows
James F. Dowling - Bose Corporation

3061 Glance/XL Advanced Analysis
Michael Hornsby - Beechglen Development, Inc.

3062 How Developers Can Adopt Self-Describing Files
David Greer - Robelle Consulting Ltd.

3063 Integrating Voice Response Into An HP Environment
Will Strother - Computer Communications Specialists
3064 Using Terminal Emulation to Perform OLTP Benchmarks
Gary A. Showers - State Farm Insurance Companies
3101 The Warning Signs, a Pop Quiz on Quality
Bob Green - Robelle Consulting Ltd.
3102 An Update On Software Inspections Methods
Steven M. Cooper - Allegro Consultants, Inc.
3103 STREAMX: It's Not Just for Passwords Anymore!
Stephen S. Hammond - Association of American Medical Colleges
3104 User Interphase Prototyping For Free
Anthony C. Furnivall - SDL/Software, Inc.
3106 How To Write Quality Pascal Programs
Stan Sieler - Allegro Consultants, Inc.
3107 The Case For OOP
Linda A. Haake - Speedware Corp.
3108 The Object Is The Subject
Douglas Colter - Speedware Corp.
3110 Why Is "EDI Standard" An Oxymoron?
Trevor Richards - M.B. Foster Associates, Inc.
3112 Integrated EDI: Is It Possible?
Rod E. Keiser - Computer Management Solutions
3113 How To Laser Away Pre-Printed Forms
Jason Kent - Proactive Systems
3114 Incorporating Dynamic Electronic Forms Into HP 3000 Applications
Ross G. Hopmans - Brant Technologies, Inc.
3115 Application Installation
Frannie Casella - Northern Calif. Cancer Center
3116 Software Re-engineering: Teaching an Old Dog, New Tricks
Christopher Nelson - S/Wizardry, Ltd.
3117 Object-Oriented Methodology: The Adager Instance
Alfredo Rego - Adager
3122 Generic Routines in COBOL Using $DEFINE and.LEN.
Rick Hoover - CIV Software
3151 Database Freedom
Alfredo Rego - Adager
3152 Migration from TurboImage to ALLBASE
George B. Scott - Great Business Solutions, Inc.
3153 ALLBASE/SQL vs Image for Application Development
Nicholas J. Walsh, Jr. - Speedware Corp.
3154 Early Returns on Critical Item Update: You Be the Judge...
Jerry Fochtman - Exxon Chemical Company
3155 Using ORACLE to Access a TurboIMAGE Database
Kimble F. Ross - Database Consultants
3156 Validating Your Database--What You Don't Know Does Hurt You!
Shelby Robert - Proactive Systems
3157 SQL for the Curious
Dirk Huizenga - Dynamic Information Systems Corp.

3158 TurboIMAGE/Standard Interface to Third Party Indexing
Eric Savage - Dynamic Information Systems Corp.

3159 The Expanded IMAGE Retrieval Toolkit
Donna Bloechl - Dynamic Information Systems Corp.

3160 Moving From TurboIMAGE To Relational
Greg Proulx - Oracle Corp.

3162 Data Integrity From TurboIMAGE To Relational
Jo-Ning Ta - Oracle Corp.

3163 Understanding Relational DBMS
Jim Eduljee - Infocentre Corporation

3166 TurboIMAGE 4.0: Relational Access
Tim Joseph, Larry Boyd - Bradmark Technologies, Inc.

3169 Normalizing IMAGE in the 90's
Rick Hoover - CIV Software

3201 Systems Integration vs. Open Systems
Eric B. Threatt - Wesson, Taylor, Wells & Associates

3202 Open Systems
David Haberman - Speedware Corporation

3251 Using Powerhouse For Data Migration
Marilyn Petty Glick - Time Systems, Inc.

3252 IBM S/36 to HP 3000/900 Migration
Jamie Harwood - Great Western Carpet Cushion Co.

3253 Transporting an MPE Application to UNIX
Todd Saylor - Bi-Tech Software, Inc.

3255 IBM to HP Migration--Overview & Case Study
Mark L. Symonds - IISI

3256 Looking at UNIX Through MPE Eyes
Betsy Leight - Operations Control Systems

3302 Meeting the Network Challenge: Running Multiple Network Protocols on a PC
Randy Robinson - Walker Richer & Quinn

3303 Client Server Computing Taking the First Step
Doug Walker - Walker Richer & Quinn

3304 Client/Server Technologies in the 90's
David Haberman - Speedware Corporation

3306 What Does A Distributed System Give You?
Niket Patwardhan - Oracle Corporation

3308 ASCII or EBCDIC: Only Your Printer Knows for Sure
Erik J. Vistica - ROLM Systems

3309 Data Transfer With NRJE: Making Your Data Turn "Blue"
Erik J. Vistica - ROLM Systems

3310 Networking HP 3000's
Robert Carnagey - Spectra Physics Scanning Systems

3351 Object Oriented Applications
David Johnson - Johnson Computer Software Team

3352 Client/Server Computing Under MPE
Joseph Geiser - Insurance Data Processing, Inc.

3354 A Client/Server Prototype for IMAGE
Ross G. Hopmans - Brant Technologies, Inc.

3355 Networking Personal Computers
Kirk E. Bedwell - Marsh Company

3356 Winning Strategies for PC Managers
Brad Feazell - Dril-Quip, Inc.

3357 The HP 3000 and PC Application Software Together at Last!
Patricia I. Loo - EG&G Idaho

3358 Work Flow Automation via NewWave
Sam Fine - The Apex Group Inc.

3359 Using Windows and Networks to Increase Productivity and Customer Service
Kevin Wilhelmsen - Space Coast Credit Union

3360 Utilizing Graphical User Interfaces in NewWave
Leonard Block - The Apex Group

3361 Personal Computer Management, Audit and Control
Dirk Huizenga - Dynamic Information Systems Corp.

3363 Using Dynamic Data Exchange to Integrate Host and Windows Applications
Brian Hargus - Walker Richer & Quinn

3364 Client/Server Technology Made Easy Using PPL
Tim Klooster - Dynamic Information Systems Corp.

3801 HP 3000 Open Distributed OLTP
Rick Bartlett - Hewlett-Packard Co.

3802 Configuring Your Disks for High Availability
Dave Beasley - Hewlett-Packard Co.

3804 MPE/iX High End Kernel Algorithms for OLTP Performance
Hung Nguyen, Korak Mitra - Hewlett-Packard Co.

3805 TurboIMAGE Applications In The 90's and Beyond
Randolph Diepeveen - Hewlett-Packard Co.

3806 Key Buzzwords for the 1990's: Open, Inter-operability, POSIX and DCE
Joseph Feiner - Hewlett-Packard Co.

3807 What Does Migrating to ALLBASE/SQL Mean For You?
Janice Hodor - Hewlett-Packard Co.

3808 Basic Dump Analysis for System Managers
David Kohl - Hewlett-Packard Co.

3809 POSIX Interface for MPE/iX
Rajesh Lalwani - Hewlett-Packard Co.

3810 AllBASE/Turbo Connect: The Write Stuff
Randolph Diepeveen - Hewlett-Packard Co.

3811 Introduction to MPE/iX Architecture
Tom Shem - Hewlett-Packard Co.

3812 Rapid Development of Client/Server Applications Using RPC
Charles Knouse - Hewlett-Packard Co.

3851 Caveat Emptor: What Buyers and Developers of Client/Server Applications Need to Know
Mike Bailey - Hewlett-Packard Co.

3852 Stored Procedures and Rules in ALLBASE/SQL
Amelia Carlson, et al - Hewlett-Packard Co.

3853 Improving the Bottom Line With Object-Oriented Databases
Douglas Dedo - Hewlett-Packard Co.

3854 HP 3000 Open Systems and Client/Server Computing
Christine Fronczak - Hewlett-Packard Co.

3855 Object-Oriented Technologies: A Management Overview
Orly Larson - Hewlett-Packard Co.

3856 Managing IT/MIS in the 90's
John Podkomorski - Hewlett-Packard Co.

3857 Implementing a Client/Server Application
Stayton Chock - Hewlett-Packard Co.

3858 HP 3000 Multiprocessing: Synchronization and its Performance Implications
Kimberly Rogers, et al - Hewlett-Packard Co.

3859 Software Resiliency
Kendall Sutton, Jessy Hsu - Hewlett-Packard Co.

3860 HP: The Best Alternative to Mainframe Computing (.... and getting there)
Robert Winter - Hewlett-Packard Co.

3862 Don't Put Your Career on Hold
John Picket - Hewlett-Packard Co.

3863 HP X.25 Access Products: Providing Open Systems Solutions
Jean-Claude Viollier - Hewlett-Packard Co.

3864 Object Oriented COBOL: The Next Generation
Megan Adams, Dmitry Lenkov - Hewlett-Packard Co.

3865 New Directions for COBOL --Where will COBOL 2000 Take Us?
Julia Rodriguez - Hewlett-Packard Co.

3867 The SLA Cookbook: A Methodology for Managing Computer System Performance
Doug McBride - Hewlett-Packard Co.

3868 Enterprise Performance Management - by exception only
Wayne Morris - Hewlett-Packard Co.

3869 Quantifying Investments in Technology
Mike Overly - Hewlett-Packard Co.

3870 What's New and Improved with MPE/V
Robert Ross - Hewlett-Packard Co.

3871 O.S.E., a framework for an Information Technology Strategy
Andre Van Aken - Hewlett-Packard Co.

3872 Managing Your Help Desk in the 1990s
Tamara Gordon - Hewlett-Packard Co.

3873 A User's Experience of HP 3000 and DTC In a Multivendor Environment
Juha Hukala - Automation Unit of Finnish Libraries

3874 Disk Arrays
Ed Pavlinik - Hewlett-Packard Co.

PAPER # 1001
Performance Comparison of the A990

Bob Combs
Combs International, Inc.
886 Belmont Avenue, Suite 3
North Haledon, NJ 07508
(201) 427-9292

Abstract

Remember when MIPS were MIPS and we didn't have to worry about who measured them? Performance comparison has become too difficult for most of us to pin down. With the advent of the new A990 HP1000 computer, the only way to see what it can really do is to compare it against other known A-series processors' performance. We will measure some simple programs, software development tasks, and a full real-time data acquisition and control system. We will compare processing loads on the various A-Series processors to that of the A990 to see how it stacks up against its predecessors.

1. Introduction

Ever since HP redefined MIPS to be some measurement other than what it stood for (Million Instructions Per Second), I haven't trusted any numbers with units which sounded like rocks out of the streambed or some form of Polish stew. Therefore I took a very pragmatic view of testing: "Let's run tests on each of the A-Series processors and compare their results." Since we're all familiar with the current line's performance, this testing yields a simple conversion factor to show what kind of performance increase one will get when moving to the A990.

When HP first announced an A-Series processor that would be faster than the A900, the A990 was promised to be twice as fast as the A900. I expected my testing to reveal how close HP had come to that promise. I also wanted proof that the A990, a new design, was compatible with its predecessors.

The A990 comes in two flavors; one for the A900 style chassis and one for the A400/A600+/A700 style chassis. The main difference between the two is that the A900 chassis has some additional memory address/enable signals. Specifically, the A990 has 28 memory address lines instead of the normal 24 in the rest of the A-Series line. Since the A400/A600+/A700 chassis doesn't have these signals, HP has produced a version of the A990 which runs them to the outside end of the cpu card; opposite the backplane. On these cards, a jumper connector is used to couple the memory cards to the A990. These additional signals are used to address up to 128kb of memory for the A990, while still using the current A900 memory cards.

There are a few other interesting points about the A990. It has a two level cache, uses low power CMOS ASIC's, a low 40 Watts of power consumption, and comes with an on board Real-Time Clock with battery backup. The A990 has an MTBF of 200,000 hours. See the end of section 4 for processor comparisons.

2. Chronology

I was afforded the opportunity to beta test three versions of the A990. The first two were the A900 chassis compatible cards, and the third was the A400 chassis version with the external memory jumper.

2.1. Beta-1

The initial card was received and tested in November of 1991. At the time HP was still testing the A990. Patches were being installed in firmware until testing was complete to minimize the number of production versions of the A990 chip sets made. Each production version is an expensive proposition. This initial beta version (we'll call it beta-1) had some firmware patches to slow it down, but ran reasonably well.

Strangely, one of my simple test programs (COPY), caused the beta-1 A990 to crash halt. What was strange was that RTE and all other tests worked fine. Even my data acquisition system and NS-ARPA worked fine.

2.2. Beta-2

The second version (call it beta-2) was really the same card, which had been returned to HP, they installed the newest set of patches, and then returned it to me. This was in December of 1991. The only problem I had seen, was eliminated, but at a speed penalty. HP had pre-warned me that the patches would slow the A990 down. The beta-2 test numbers reflect this.

The strange bug seen with the beta-1 card was fixed and no other problems were encountered throughout testing. The A990 proved to be a very solid processor. (This is a form of pun since the A990 has the longest MTBF of any A-Series processor.)

2.3. A990 (final)

The final version was, in fact, the production version of the A990. It arrived the end of January of 1992. This time, the card type was the one compatible with A400 chassis. The production A990 not only ran faster than the beta-2 patched version, but also ran faster than the original beta-1 card tested. The speed increase was so noticeable that when first seen, it makes one smile.

An interesting side story: the A990 was code named *millennium*, but all of its proprietary design components were named after famous rabbits. The cache ASIC that had the original errors (thus requiring the intermediate firmware patches) was code named "Bugs". Who says fate doesn't have a sense of humor?

3. Test Suite

There are three test suites that were developed. The first set are simple minded programs that test speeds of specific cpu features. The second set tests development speed by timing the speed of compiling and linking. And the final test was the running of a full blown real-time application.

All tests were run several times and the values averaged to insure fair numbers for each processor. Note that most tests used the RTE routines **ResetTimer** and **ElapsedTime.**

Remember, the purpose of these tests are not to give specific timing values for a processor, but to compare the various A-Series processors to one another in different areas.

3.1. Simple Minded Tests

The simple minded tests are run with nothing else running on the computer. They are designed to 1) check memory speed, 2) calculation speed, 3) file open speed, 4) file copying speed, 5) terminal output speed, and 6) graphics output speed.

3.1.1. MEM

The memory test swaps two memory locations back and forth 30,000 times, timing the process. This time test is then performed 100 times, averaging the times.

```
FTN77
      Program Mem(3,40),simple speed check program
*
*     Check the timing of some number of memory loop accesses.
*
***********************************************************************
      Implicit none

      Integer*4       ElapsedTime
      Integer*2       I, IA, IB
      Integer*2       J
      Integer*4       LoopCount
      Integer*2       Lu(5)
      Integer*4       TotalCount
      Real*4          XCount

      Call RMPAR(Lu)
      If (Lu .LE. 0)  Lu = 1

      Write(Lu,'(10X,"Mem timing:")')

      TotalCount = 0j
      Do I = 1, 100

          Call ResetTimer

          Do J = 1, 30000
              IA = IB
              IB = I
          EndDo

          TotalCount = TotalCount + ElapsedTime()

      EndDo

      XCount = Real(TotalCount)/100.

      Write(Lu,'(10X,"time of 30000 count = ",F12.3," msec",/)') XCount

      End
```

3.1.2. CALC

CALC uses a similar loop mechanism as MEM, 100 sets of 10000 computations. There are five loops to measure integer*2, integer*4, real*4, and real*8 computations. The fifth loop is a blank loop to measure the DO loops benchmark overhead. In each of the comparisons in the next section, the null loop offsets were subtracted from the math loops to present a *more true* comparison.

```
FTN77
      Program Calc(3,40),simple calculation speed check program
*
*     This program will check the speed of a number of iterations.
*     Note that the item of significance is the comparison between
*     the A990 and the other A-Series processors, not the actual values.
*
***********************************************************************
      Implicit none

      Integer*4       ElapsedTime
      Integer*2       I1, I2, I3, I4, I5
      Integer*2       I, IA, IB
      Integer*4       J1, J2, J3, J4, J5
      Integer*2       J
      Integer*4       LoopCount
      Integer*2       Lu(5)
      Integer*4       TotalCount
      Real*4          XCount
      Real*4          X1, X2, X3, X4, X5
      Real*8          Z1, Z2, Z3, Z4, Z5

      Data I1/1/, I2/2/, I3/3/, I4/4/
      Data J1/1j/, J2/2j/, J3/3j/, J4/4j/
      Data X1/1./, X2/2./, X3/3./, X4/4./
      Data Z1/1.D0/, Z2/2.D0/, Z3/3.D0/, Z4/4.D0/


      Call RMPAR(Lu)
      If (Lu .LE. 0)  Lu = 1

      Write(Lu,'(10X,"Calc timing(10000 counts):")')

* -- integer arithmetic
      TotalCount = 0j
      Do I = 1, 100
          Call ResetTimer
          Do J = 1, 10000
              I5 = (I1+I2*I3)/I4
          EndDo
          TotalCount = TotalCount + ElapsedTime()
      EndDo
      XCount = Real(TotalCount)/100.
      Write(Lu,'(10X,"Integer*2 =        ",F12.3," msec",/)') XCount

* -- double integer arithmetic
      TotalCount = 0j
      Do I = 1, 100
          Call ResetTimer
          Do J = 1, 10000
              J5 = (J1+J2*J3)/J4
          EndDo
          TotalCount = TotalCount + ElapsedTime()
      EndDo
      XCount = Real(TotalCount)/100.
      Write(Lu,'(10X,"Integer*4 =        ",F12.3," msec",/)') XCount

* -- real arithmetic
      TotalCount = 0j
      Do I = 1, 100
          Call ResetTimer
          Do J = 1, 10000
              X5 = (X1+X2*X3)/X4
          EndDo
          TotalCount = TotalCount + ElapsedTime()
      EndDo
      XCount = Real(TotalCount)/100.
      Write(Lu,'(10X,"Real*4 =           ",F12.3," msec",/)') XCount

* -- double real arithmetic
      TotalCount = 0j
      Do I = 1, 100
          Call ResetTimer
          Do J = 1, 10000
              Z5 = (Z1+Z2*Z3)/Z4
          EndDo
          TotalCount = TotalCount + ElapsedTime()
      EndDo
      XCount = Real(TotalCount)/100.
      Write(Lu,'(10X,"Real*8 =         ",F12.3," msec",/)') XCount

* -- baseline timing overhead
      TotalCount = 0j
      Do I = 1, 100
          Call ResetTimer
          Do J = 1, 10000
          EndDo
          TotalCount = TotalCount + ElapsedTime()
      EndDo
      XCount = Real(TotalCount)/100.
      Write(Lu,'(10X,"overhead timing =  ",F12.3," msec",/)') XCount

      End
```

3.1.3. FILE

This test opens a file reads a line and then closes the file. The file is accessed 100 times, timing each and then averaging the time.

```
FTN77
      Program File(3,40),simple file speed check program
*
*     This program will check the time file access takes.
*
***********************************************************************
      Implicit none

      Character*64      CName
      Integer*4         ElapsedTime
      Integer*2         FmpClose, FmpOpen, FmpRead
      Integer*2         I, IA, IB
      Integer*2         IBuf(128)
      Integer*2         IDCB(144)
      Integer*2         IErr
      Integer*2         IType
      Integer*2         J
      Integer*4         LoopCount
      Integer*2         Lu(5)
      Integer*2         RLen
      Integer*4         TotslCount
      Real*4            XCount

      Call RMPAR(Lu)
      If (Lu .LE. 0)  Lu = 1

      Write(Lu,'(10X,"File timing:")')

      TotalCount = 0j
      CName = '/A990/JUNK_FILE'
      Do I = 1, 100
          Call ResetTimer

          IType=FmpOpen(IDCB,IErr,CName,'RO',1)
          If (IErr .LT. 0) GoTo 800
          RLen=FmpRead(IDCB,IErr,IBuf,256)
          If (IErr .LT. 0) GoTo 800
          IErr=FmpClose(IDCB,IErr)
          If (IErr .LT. 0) GoTo 800

          TotalCount = TotalCount + ElapsedTime()
      EndDo

      XCount = Real(TotalCount)/100.

      Write(Lu,'(10X,"file time = ",F12.3," msec",/)') XCount
      GoTo 900

  800 Write(Lu,'(1x,"file I/O error ",I6)') IErr

  900 End
```

3.1.3.1 Junk_File

This single line file was used by both FILE and COPY.

```
This is a simple junk file for reading by the file access test.
```

3.1.4. COPY

COPY does just that. It copies the Junk_File to a new temporary file and purges it. Again, the time is averaged from 100 iterations.

```
FTN77
      Program Copy(3,40),simple file copy speed check program
*
*     This program will check the time file copy I/O takes.
*
***********************************************************************
      Implicit none

      Character*64    CName, CNameT
      Integer*4       ElapsedTime
      Integer*2       FmpCopy, FmpPurge
      Integer*2       I, IA, IB
      Integer*2       IBuf(288)
      Integer*2       IErr, IErr1, IErr2
      Integer*2       Lu(5)
      Integer*4       TotalCount
      Real*4          XCount

      Call RMPAR(Lu)
      If (Lu .LE. 0)  Lu = 1

      Write(Lu,'(10X,"Copy timing:")')

      TotalCount = 0j
      CName = '/A990/JUNK_FILE'
      CNameT = '/A990/TEMP_FILE'
      Do I = 1, 100
D        Write(1,'("beginning pass ",I3)') I
          Call ResetTimer

          IErr=FmpCopy(CName, IErr1,CNameT,IErr2,IBuf,288,'A')
          If (IErr .LT. 0) GoTo 800

          TotalCount = TotalCount + ElapsedTime()
          IErr=FmpPurge(CNameT)
      EndDo

      XCount = Real(TotalCount)/100.

      Write(Lu,'(10X,"file copy I/O time = ",F12.3," msec",/)') XCount
      GoTo 900

  800 Write(Lu,'(1x,"file I/O error ",I6)') IErr

  900 End
```

3.1.5. OUTPUT

We didn't expect any real output difference between the processors, but decided to test it anyway. Here a common text line is output for 100 (or 500) times and the overall process timed. Note that we used a mux card and the same terminal for each processor tested to insure comparability.

```
FTN77
      Program Output(3,40),simple I/O speed check program
*
*     This program will check the time I/O takes.
*
***********************************************************************
      Implicit none

      Character*64    CBuf
      Integer*4       ElapsedTime
      Integer*2       I
      Integer*2       IBuf(32)
      Integer*2       Lu(5)
      Integer*2       LUO(2)
      Integer*2       NOut
      Real*4          XCount

      Equivalence (IBuf,CBuf)

      Call RMPAR(Lu)
      If (Lu .LE. 0)  Lu = 1
      LUO(1)=Lu
      LUO(2)=0
```

```
If (LU(2) .GT. 0) LUO=Lu(2)
NOut = 100
If (Lu(3) .GT. 100) NOut=Lu(3)

Write(Lu,'(10X,"Output ",I6," I/O timing:")') NOut

CBuf='The quick brown fox jumped over the lazy dogs back. '
Call ResetTimer
Do I = 1, NOut
    Call XLUEX(2,LUO,IBuf,-64)
EndDo
XCount = Real(ElapsedTime())/NOut

Write(Lu,'(10X,"I/O time = ",F12.3," msec",/)') XCount

End
```

3.1.6. TGRAF

This program draws a color graphic picture (process diagram) on the graphic terminal. Since the graphics routines (our PGRAF product) are layer on top of DGL, we suspected there might be some processor overhead. So we tested the time it took to output a picture.

```
FTN77
$CDS OFF
$FILES 1,1,1
      PROGRAM TGRAF()
     |,CII PGRAF <911104.1213>
*
*     This module will demo the use of the PGRAF library routines.
*
*     Note:
*         Predefined symbol objects with the following names have been
*         assumed for the following calls:
*
*         call COLOR  "REACTOR         "
*         call TEXT   "TEMP            " "RATE"
*         call FILL   "TANK            " "BAR1" "BAR2"
*                     "HOPPER          " "FURNACE"  "REACTOR"
*         call SCALE  "GRAPH           "
*         call GRAPH  "GRAPH           "
*         call ROTATE "PRESSURE        "
*
*     runstring:
*         RU TGRAF [filename]
*     inputs:
*         filename    optional picture filename to display
*                     default is PLANT.PIC
*
**********************************************************************
      IMPLICIT NONE

      INTEGER*2       ALPHA_OFF(2)    ! escapes to turn off alpha screen
      INTEGER*2       ALPHA_ON(2)     ! escapes to turn on alpha screen
      INTEGER*2       COLOR           ! PGRAF library routine
      INTEGER*2       DGRAF           ! PGRAF library routine
      INTEGER*2       EGRAF           ! PGRAF library routine
      INTEGER*2       ERROR_TYPE      ! error
      INTEGER*2       FILE_ID         ! window #
      INTEGER*2       FILL            ! PGRAF library routine
      INTEGER*2       GRAPH           ! PGRAF library routine
      INTEGER*2       IERR            ! error
      INTEGER*2       ROTATE          ! PGRAF library routine
      INTEGER*2       SCALE           ! PGRAF library routine
      INTEGER*2       TEXT            ! PGRAF library routine
      CHARACTER*64    USER_FILE       ! picture filename
      Integer*4       JTime,ElapsedTime
      Real*4          XTime


*                    esc *  d  F
      DATA ALPHA_OFF/15452B,62106B/
*                    esc *  d  E
      DATA ALPHA_ON /15452B,62105B/


*     fetch input file
      USER_FILE=' '
      CALL FPARM(USER_FILE)
      IF(USER_FILE.EQ.' ') USER_FILE='/PGRAF/PLANT.PIC '

*     display PGRAF user picture file
```

```
      Call ResetTimer()
      IERR=DGRAF(1,0.0,1.0,0.0,1.0,USER_FILE,FILE_ID,ERROR_TYPE)
      JTime=ElapsedTime()
      XTime=JTime*.001
*     display time it took to draw static picture
      Write(1,*) 'Time = ',XTime

*     ignore rest of program
        . . .

      END
```

3.2. Development Tests

The development speed tests consisted of timing a typical compile and link of reasonably good size module. The program HPMDM.FTN was chosen since it is a standard HP module which most are familiar. Timing programs COMP and LOAD were written to perform the tests and automatically average the result of 10 timings.

3.2.1. COMP

```
FTN77
      Program Comp(3,90), <911126.1059>
*
*     Time a fair sized compile.
*
**********************************************************************
      Implicit none

      Integer*4       ElapsedTime
      Integer*2       FmpRunProgram
      Integer*2       I
      Integer*2       IErr
      Integer*2       IfBrk
      Integer*4       JTime
      Integer*2       Parms(5)
      Character*6     RunName
      Character*64    String
      Integer*4       Total
      Real*4          XSecs

      String='RU,FTN7X,/GFOX/RTE_A5241/HPMDM,,-'

      Do I=1,10
          If (IfBrk().NE.0) Stop
          RunName=' '
          Call ResetTimer
          IErr=FmpRunProgram(String,Parms,RunName)
          JTime=ElapsedTime()
          Total=Total+JTime
          Write(1,'("pass ",I2,"  time ",I10)')  I, JTime
      EndDo

      XSecs=Real(Total)/10000.0

      Write(1,'("Compile time = ",F7.3," sec")') XSecs

      End
```

3.2.1. LOAD

```
FTN77
      Program Load(3,90), <911126.1100>
*
*     Time a fair sized link.
*
**********************************************************************
      Implicit none

      Integer*4       ElapsedTime
      Integer*2       FmpRunProgram
      Integer*2       I
      Integer*2       IErr
      Integer*2       IfBrk
      Integer*4       JTime
      Integer*2       Parms(5)
      Character*6     RunName
      Character*64    String
      Integer*4       Total
```

```
Real*4          XSecs

String='RU,LINK,HPMDM.LOD'

Do I=1,10
    If (IfBrk().NE.0) Stop
    RunName=' '
    Call ResetTimer
    IErr=FmpRunProgram(String,Parms,RunName)
    JTime=ElapsedTime()
    Total=Total+JTime
    Write(1,'("pass ",I2,"  time ",I10)')  I, JTime
EndDo

XSecs=Real(Total)/10000.0

Write(1,'("Link time = ",F7.3," sec")') XSecs

End
```

3.3. Real-Time System Tests

We used our MAXS+ system which is a data acquisition and control system. It scans analog/digital I/O converts the values to engineering units, processes a set of control and monitoring computations, checks for alarm values, trends signal values, and writes history to archive files. The front end signal gathering was replaced with some dummy signal generation modules to insure repeatability of the data. This seemed to work quite well as the data was very repeatable and therefore a fair comparison. MAXS+ is a typical HP1000 application, making this set of tests a good overall real-world test.

The MAXS+ has a performance monitor built into it which measures how many variables per second are being processed, as well as some additional performance data on the various stages of processing. The bulk of the processing was in the high priority stage labeled as 'A scan'. Another timing program was written which could time any program (TIMETEST). This program was then used to time a couple of the commands. Namesort sorts all of the variable names into link-list sorted alphabetically order. The SV command lists the variables from memory. The SI command lists all commands in the MAXS+ command table, which requires an in memory sort.

3.3.1. TIMETEST

```
FTN77
      Program TimeTest(3,90), <911127.1241>
*
*     Time the command.
*
**********************************************************************
      Implicit none

      Integer*4       ElapsedTime
      Integer*2       FmpRunProgram
      Integer*2       I
      Integer*2       IErr
      Integer*2       IfBrk
      Integer*2       IString(40)
      Integer*4       JTime
      Integer*2       Len
      Integer*2       Parms(5)
      Character*6     RunName
      Character*80    String
      Integer*4       Total
      Real*4          XSecs

      Equivalence (String,IString)


      String=' '
      Call GetSt(IString,-80,Len)

      Do I=1,10
          If (IfBrk().NE.0) Stop
          RunName=' '
```

```
        Call ResetTimer
        IErr=FmpRunProgram(String,Parms,RunName)
        JTime=ElapsedTime()
        Total=Total+JTime
        Write(1,'("pass ",I2,"  time ",I10)')  I, JTime
EndDo

XSecs=Real(Total)/10000.0

Write(1,'("Command time = ",F7.3," sec")') XSecs

End
```

4. Test Results

The tables presented here summarize the results of the tests. It should be taken into account that the A900 used a faster disc that the other machines. Therefore, the FILE and COPY comparisons against the A900 are misleading. Most of the other tests are either compleetly memory based or mostly memory based and are fairly accurate comparisons.

27-Jan-92	A400	A600+	A900	A990	Comparison of A990: vs A400	vs A600+	vs A900
Mem:	244.03	286.84	111.20	24.30	10.04	11.80	4.58
Calc:							
Int*2	229.33	231.42	126.53	30.40	7.54	7.61	4.16
Int*4	276.40	332.84	123.03	38.27	7.22	8.70	3.22
Real*4	448.17	643.18	111.93	39.30	11.40	16.37	2.85
Real*8	1388.03	1483.24	191.40	80.47	17.25	18.43	2.38
File:	132.47	132.26	62.33	83.23	1.59	1.59	0.75
Copy:	846.93	830.58	895.63	618.40	1.37	1.34	1.45
Output:	3.83	3.50	1.57	0.74	5.18	4.74	2.13
TGraf:	10.07	12.03	11.80	8.76	1.15	1.37	1.35
Compile	47.18	48.94	34.80	15.99	2.95	3.06	2.18
Link	21.38	22.09	14.40	9.07	2.36	2.44	1.59

Note: all times in msec.

MAXS+scan 133 variables of mixed type

	A400	A600+	A900	A990	A990 vs A400	vs A600+	vs A900
A (msec)	0.97	1.05	0.38	0.18	5.39	5.83	2.11
var/sec	137.11	126.67	350.00	738.89	5.39	5.83	2.11
namesort	3.51	4.05	2.59	0.99	3.54	4.08	2.61
svrpt *.* @ll junk	6.25	6.68	6.04	2.40	2.61	2.78	2.52
sirpt * @ll junk	7.58	7.74	4.86	4.42	1.71	1.75	1.10
Scan total	1.01	1.10	0.45	0.19	5.32	5.79	2.37

The A990 tests shown below are for the final A990 card.

A990 (Beta C version)		HP7958 disc		Average
mem	24.3	24.3	24.3	24.30
int*2	34.7	34.5	34.5	30.40
int*4	42.5	42.4	42.4	38.27
real*4	43.4	43.4	43.6	39.30
real*8	84.7	84.6	84.6	80.47
null loop	4.3	4.3	3.9	4.17
file	83.2	83.3	83.2	83.23
copy	610.2	622.7	622.3	618.40
output 100	0.4	0.4	0.4	0.40
output 500	0.74	0.74	0.74	0.74
tgraf R7	8.76	8.76	8.77	8.76
tgraf 2397	1			1.00
Compile	15.969	15.982	16.008	15.99
Load	9.07	9.07	9.067	9.07

A900 times:		HP2203 disc					Average
mem	101.3	139.3	104.7	99.5			111.20
int*2	141.7	144.6	142.8				126.53
int*4	151	131.6	136				123.03
real*4	139.5	124.3	121.5				111.93
real*8	217.1	203.7	202.9				191.40
null loop	16.5	17.4	15.6				16.50
file	62.6	61.2	63.2				62.33
copy	889	897.8	900.1				895.63
output 500	1.56	1.56	1.6				1.57
tgraf 2397	11.76	11.77	11.88				11.80
Compile	19.107	19.14	19.109	19.117	19.14	19.125	19.12
Load	10.538	10.566	10.568	10.566	10.555	10.58	10.56

The A900 tests were taken from a different hardware configuration that used a faster disc, HP2203 (24 msec) than the HP7958 (28 msec) used with the other processors. This accounts for some of the disc I/O related differences noted.

The A600+ and A400 tests were performed on the same identical hardware as the A990.

A600+ times:		HP7958 disc				Average
mem	286.9	286.9	286.7	286.8	286.9	286.84
int*2	279.2	280	279.2	279.3	279.7	231.42
int*4	380.9	380.9	381	380.9	380.8	332.84
real*4	691.2	691.5	691.3	690.7	691.5	643.18
real*8	1531.3	1531.4	1531.3	1531.3	1531.2	1483.24
null loop	48.8	48.3	47.4	48.2	47.6	48.06
file	132	133.2	132.1	132	132	132.26
copy	829.5	829.9	830.5	831	832	830.58
output 100	1.7	1.7	1.7	1.7	1.6	1.68
output 500	3.5	3.5	3.52	3.5	3.5	3.50
tgraf R7	9.47	9.45	9.49	9.47		9.47
tgraf 2397	12.27	12.03	11.79	12.03		12.03
Compile						
Load						

A400 times:		HP7958 disc				Average
mem	244.3	244.3	243.5			244.03
int*2	272.7	272.9	273.3			229.33
int*4	320.1	320	320			276.40
real*4	491.6	491.9	491.9			448.17
real*8	1431.9	1431.4	1431.7			1388.03
null loop	43.6	43.6	43.7			43.63
file	132.2	132.4	132.8			132.47
copy	847.1	847.2	846.5			846.93
output 100	1.8	1.8	1.8			1.80
output 500	3.9	3.8	3.8			3.83
tgraf 2397	10.06	10.07	10.08			10.07
Compile	47.167	47.174	47.18	47.197		47.18
Load	21.38	21.368	21.377	21.383		21.38

The numbers seem also to show that the A400 is about 10% faster than the A600+.

This table shows the comparison of the three versions of A990s tested. The comparison is not very relevant to this paper, but is of some interest to note the dramatic speed difference between firmware patches and implementing the logic in the A990's custom ASIC chips.

Comparison of Beta-1, Beta-2, & final A990 versions

	Beta-1	Beta-2	A990 final
mem	30.45	51.68	24.30
int*2	59.38	66.93	30.40
int*4	66.52	77.60	38.27
real*4	61.50	69.48	39.30
real*8	114.97	114.80	80.47
file	82.97	83.03	83.23
copy	*638.35	645.38	618.40
output 100	0.50	0.53	0.40
output 500	0.98	1.13	0.74
tgraf	9.10	8.95	8.76
comp	-	19.12	15.99
load	-	10.56	9.07

* Beta-1 COPY had to run differently due to crash "bug".

Here are some other comparisons of the various A-Series processors. This information was obtained from HP data sheets and specifications.

	A400	A600+	A900	A990
Logic family	CMOS/TTL	TTL	TTL	CMOS
Power (Watts)	32	53	157	40
System reliability comparison (HP243x)	2.8	1.9	1.0	3.0 *

* Estimated MTBF system comparison with A900.

The power consumptions and logic family used are per cpu board(s). The system reliability is a comparison of MTBF ratings for the Micro 2x system packages, using the A900 as the normal benchmark. The numbers indicate, for example, that an A990 system should be down only one third as often as an A900.

5. Conclusions

The two prime questions I was looking for in my testing were:

Is the A990 fully compatible with the other RTE machines?

and

Is the A990 twice as fast as the A900?

Both answers are yes.

The compatability of the A990 with its predecessors was to be expected from HP. Still, it's nice to prove it. It is 100% compatible with the other A-Series processors.

The speed of the A990 is over twice that of the A900. How much over is somewhat dependent upon the type of program you run, but my results indicate that HP exceeded their promise by a speed factor of anywhere from 2.11 to 2.37 on average. And some application programs may run as much as 4.5 times faster.

Lilly Fermentation Process Control System

Lincoln Brill; Biochemical Engineering
Lilly Research Laboratories,
Division of Eli Lilly and Company

Several years ago, it was decided to modernize the Lilly Fermentation Process Control System (LFPCS). As a result of this decision, it was upgraded to an HP A990 RTE-A computer system from an HP E-series system. This upgrade consisted of redesigning the LFPCS, developing the necessary software and installing three computer development systems.

The LFPCS requires a wide variety of functionality, such as: recipe processing, record logging and front-end and operator interface communications. To provide this functionality for the LFPCS redesign, several software applications were developed. These applications were then prioritized based on their I/O and CPU needs and built into a memory-based system. To design recipes (i.e. process control strategies), a compiler named "Recipe Builder" was created, and, to configure the LFPCS specific to a plantsite's needs, the Level 1 Table Generator was developed. The last component of the software included the system management programs, namely MAKE and command files. These programs make it easier to control the software development and system management responsibilities.

To complete the LFPCS project, three computer development systems were installed. These systems included three HP A-series computers, three front-ends (Foxboro UISCM, Opto22, and Allen-Bradley PLCs), an NS connection, operator interface links, host links and miscellaneous peripherals.

Paper # 1003

An IMAGE-II Data Extraction Accelerator

Donald A. Wright
Interactive Computer Technology
2069 Lake Elmo Avenue North
Lake Elmo, MN 55052 USA
612/770-3728

Abstract:

HP's IMAGE/1000-II Data Base Management System is a comprehensive structured data base with functionality similar to that of the HP3000 Turbo-Image system. It supports most data types and has very few practical limitations with respect to data set sizes, record lengths, numbers of key items, and the like. It even includes a comprehensive built-in backup (logging) mechanism with both roll-back and roll-forward capabilities.

What IMAGE-II does NOT have is high performance. Because of its single-threaded design, the performance issue becomes especially noticeable if concurrent data base access is needed by two or more programs, even when they are accessing different data bases. However, even when only one program is accessing the IMAGE system at a time, IMAGE's performance does not compare well with normal programmatic access to flat files.

This paper describes the results of a successful attempt to accelerate IMAGE searches, with serial searches improved by factors of 5 to 30 or more. The methods used are practical and are implementable at the application level; there is no attempt to modify any of the IMAGE structures, callable subroutines, or utility programs including DBMON. Preferably some of these methods would eventually be used by HP within the IMAGE subsystem itself to transparently improve IMAGE's overall performance, but until then these methods are within the reach of any HP1000 programmer who needs to implement them.

IMAGE/1000-II Overview:

A working IMAGE-II Data Base Management System may be described as having five different components:

1) Data Base files. These are Type-1 RTE files containing data base structure information, and Type-2 RTE files containing application information and IMAGE pointers.

2) IMAGE Monitor Program, DBMON. This HP program does all normal reads and writes to the data base.
3) HP IMAGE utility programs for performing startup, backup, restore, cleanup, and other maintenance functions.
4) An application-callable IMAGE library of HP subroutines which allow the user's application to open, close, read, write, and perform other data base functions through DBMON.
5) Application programs written by the user.

The accelerator to be described below is a sixth component: A user-written application-callable subroutine which makes direct accesses to the IMAGE RTE files, bypassing DBMON.

Any normal IMAGE-II application program calls several of the subroutines in the IMAGE library. Those subroutines in turn schedule the DBMON program, asking it to do any necessary reads and writes and some much more complex operations. See Figure 1:

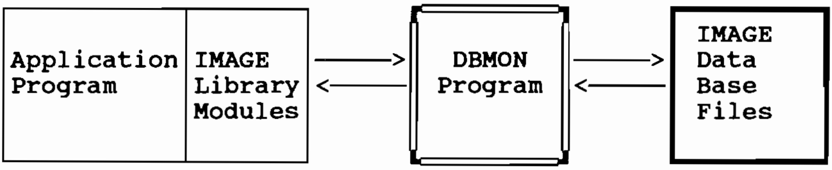


Figure 1, Application Program Accessing Data Through the DBMON Program

The IMAGE subsystem allows several different kinds of reads, through the callable subroutine DBGET:

1) Serial reads. These can be forward or backward; data set entries are read in the order of their physical appearance within the data set file.
2) Chained reads. IMAGE follows a chain of linked entries in a detail data set.
3) Directed reads. The application asks for a specific entry by record number.
4) Keyed reads. IMAGE finds an entry in a master data set according to the value of a key item.

The longest searches usually involve a serial read, because a complete search of a data set calls for a physical read of every single occupied record in the data set. In fact, under some circumstances even the unoccupied records must be examined.

Accelerating Serial Searches:

Most of the effort in this project was aimed at speeding up serial searches, although the methodology does apply to a lesser extent to the other types of reads as well. There was no effort to improve the speed of writing; the special subroutine developed here is read-only. All of the following methods were used to accelerate reads:

1) Disk reads are done via EXEC calls directly from the application program, bypassing DBMON.
2) Large blocks of disk data are read at a time.
3) Reads employ read-ahead for forward serial searches, read-behind for backward serial, and read-centered for all other modes.
4) A disk access is never done if the data already exists in the memory buffer.
5) Disk and DBINF call results are remembered, so those calls are not repeated on successive uses unless a change demands it.
6) Serial reads stop when all occupied records have been examined. Backward serial reads begin at the highest-used record location.
7) Type-2 records are examined directly in the large buffer rather than using FMP calls to move records to a local record buffer.
8) Masking of data (selection of only those entries having an item which matches a mask) can be specified in the search call and performed in the search subroutine itself.

Setup/Initialization:

Because this accelerator lies on top of IMAGE and does not have access to the memory data structures used by the IMAGE-access library routines, it is necessary to locate some important information about the data set prior to beginning any search. Some of those items are:

1) Data set record size, including media portion of record.
2) Data set starting track and sector, for EXEC calls.
3) Number of records in the data set.
4) Highest-used record in the data set.
5) Size, type, and organization of items in the data set.

Some of this information is available from DBMON by way of DBINF calls, but the physical location of the data set, its record size, and the highest-used record are not returned by DBINF. The data base root file must be accessed to obtain this information. The author has no access to IMAGE sources or any other information regarding IMAGE data structures, so the information below has been determined empirically. It may change if IMAGE is enhanced by HP.

The root file is a Type-1 file which appears to have this organization (partial description):

- Block 1: Pointers and important values:
  - Word 63: Number of sets in data base.
  - Word 65: Number of items in data base.
  - Word 69: Root file block# containing usage pointers.

- Block 2:
  - Beginning in Word 1: A 2-word specification for each item. These may actually extend into block 3 or higher, depending on the number of items.
  - Immediately following the item specifications: 40-word descriptions of each data set, in which the first 32 words contain a complete file descriptor of the data set.

- Usages block - pointed to by Word 69 of block 1. 8 words per data set, in which words 5 and 6 are the doubleword number of the highest-used entry. May extend into additional blocks, depending on the number of data sets.

Access to these areas of the root file, along with several DBINF calls, provides all of the information necessary to read the data set file directly. In our implementation the search subroutine is called DBSEARCH, and the application program calls a setup entry point to DBSEARCH with these values:

- IBASE value from the DBOPN call.
- Data Base root file descriptor, as provided to DBOPN.
- Data set name, 6 characters, or data set number, integer.
- ISTAT return array, same as for IMAGE.
- A large scratch array.
- The size of the scratch array in words.

The setup entry point takes the following steps to initialize the search subroutine:

1) The data base root file is opened non-exclusively and its type (type 1) is confirmed. Its LU number, starting track, and starting sector are determined from DCB words 1, 4, and 5 respectively, and it is closed.

2) An EXEC read is done on the first block of the root file, and three needed values are determined: Number of items, number of data sets, and the block number where data set usage information begins.

3) If the set name is an integer, we index into the root file to the data

set description corresponding to that number. If it is a name, we search the descriptions for the one with the corresponding file name.

4) The data set file is opened nonexclusively and its LU, starting track, starting sector, size, and record length are determined from DCB words 1, 4, 5, 6, and 7 respectively. Then it is closed.

5) DBINF is called once in mode 104 to obtain the item list, and then repeatedly in mode 102 to assemble a list of items in the specified data set, complete with data types and item lengths.

6) DBINF is called once in mode 202 to determine the set type (detail or master), entry length (data only), capacity, current number of records used, and number of paths.

7) The root file is accessed one last time to determine the highest-used record in the data set. Note: This is NOT necessarily the same as the current number of records used, because some records may have been deleted since the highestused record "highwater mark" was established.

This information is all saved in the subroutine's data structure for later use in normal search calls. A SAVE statement was used after the subroutine's variable declarations so that they are preserved even when CDS is used.

Actual Search Calls:

Once the accelerator subroutine is set up for the required data set, the search is performed by making repeated calls to DBSEARCH. This is very similar to the manner in which DBGET is normally used. The following parameters are passed:

- IBASE value from the DBOPN call.
- Scratch array specified in the setup call.
- Mode: Same as DBGET mode, except that keyed reads are not supported.
- ISTAT array, used the same as in IMAGE calls, except that words 3 & 4 are used in modes 1-4 to specify the "current record" at subroutine entry, and the forward and backward chain pointers are used similarly in modes 5 and 6.
- IBUF data-return array, long enough for a complete data set entry.
- Optional item name or number, for masking.
- Optional character mask.
- Optional characters specifying masking options such as casefolding.

The subroutine takes the following steps in executing each call:

1) Check the number of passed parameters for legality and to determine whether masking is to be done.

2) Initialize the search path, if necessary, according to the mode.

3) If not the same item as previously referenced, call DBINF to get its full name and number. Then find it in the set's item list and determine its length and type.

4) Determine the pending record number, depending on the mode, making sure that we have not already examined the total number of filled records in the data set.

5) Compute the record's disk block address and determine if we already have the required blocks in the memory buffer. If not, calculate the bounds of the read, depending on mode, and do a read.

6) Determine whether or not the accessed record is empty, by examining word 6 of the record. This is an error unless the mode is a serial read, in which case we go back to step 4 and try the next record.

7) If we are doing a masked search, compare the specified item with the supplied mask. If no match, loop up to step 4 and try the next record.

8) Return to the caller with the data entry.

Performance:

In an effort to measure the accelerator's value, timed forward serial searches of several different data bases were done, each with three different programs:

1) QUERY, doing a simple masked serial search and building a select file. Results are shown under QUERY in Table 1 below.

2) A program called DBGETTEST, written especially for this occasion. It uses repetitive DBGET calls, as any normal application would do, and performs the same masked search as does QUERY, although it does not build the select file. Results are listed under DBGET in Table 1 below.

3) A program called MASKTOSELECT, written for use at ICT. It uses the new DBSEARCH accelerator subroutine to perform the same masked search as QUERY and builds the same select file. The DBSEARCH subroutine is given a disk buffer area of 8K words

in the program's data space (not EMA). Results are listed under QUICK in Table 1 below.

Case 1: SALES data base, at ICT. A400 CPU with 7958 disk. Search the PEOPLE data set, sized at 16896 blocks, 92.3% full (4.0 Mbytes). Mask the 16-character LNAME (last name) item for @ANDERSON@. In 9235 records, 33 qualify. The same 33 are found by all three programs (a good sign).

Case 2: BOOKS data base, at ICT. A400 CPU with 7958 disk. Search the JOURNL data set, sized at 3052 blocks, 43.1% full (336K bytes). Mask the 28-character item DESCR for all occurrences of @HEWLETT@. In 2761 records, 46 qualify.

Case 3: PEOPLE (personal mailing list) data base, at ICT. A400 CPU with 7958A disk. Search the DATA data set, sized at 1352 blocks, 63.1% full (218K bytes). Mask the 2-character ADRST item for all occurrences of "IA". In 631 records, 48 qualify.

Case 4: LUG (local user group) data base, at ICT. A400 CPU with 7945 disk. Search the MEMBRS data set, sized at 736 blocks, 54.2% full (102K bytes). Mask the 32-character ORG item for @HEWLETT@. In 271 records, 15 qualify.

Case 5: DATA data base, different computer. A900 CPU with 7914 disk. Search the RECORD data set, sized at 80,894 blocks, 2.4% full (497K bytes). Mask the 60-character item KYITEM for all occurrences of @LMO@. In 1179 records, 19 qualify. See also the discussion below regarding the sparse data set.

Other test conditions: On both systems DBMON is loaded to maximum size (DBMON.LOD defaults) for maximum performance. Both systems were otherwise idle during the tests. Most runs were repeated several times to demonstrate repeatability. Small differences usually did appear, in which case the lowest values were always used.

The ratios shown below are the number of seconds for the QUERY or the DBGET run divided by the number of seconds for the QUICK run. They show the performance improvement factor available from the QUICK method:

Case	DB	Size			Seconds required for search				
#	Name	Kbyt	Disk	CPU	QUICK	QUERY	Ratio	DBGET	Ratio
1	SALES	4000	7958A	A400	17.6	387.0	22.0	552.5	31.4
2	BOOKS	336	7958A	A400	4.4	102.0	23.2	93.3	21.2
3	PEOPLE	218	7958A	A400	3.0	25.0	8.3	37.5	12.5
4	LUG	102	7945	A400	2.7	15.0	5.6	18.9	7.0
5	DATA	497	7914	A900	11.6	139.0	12.0	152.6	13.2

Table 1, Results of 5 cases, each done 3 different ways.

Discussion:

The smallest ratios of performance improvement (5.6, 7.0, 8.3, etc.) appear in tests with relatively small amounts of data. It appears that the QUICK method has a lower limit of perhaps one to two seconds on the A400, due to setup time. The more impressive performance improvement ratios all occur with larger searches.

The 8K-word buffer used with the QUICK (MASKTOSELECT) program is entirely practical in most applications. It is always possible to divide an application in such a way that the search program can be relatively small and thus can have a large buffer for disk reads. In fact, when that same MASKTOSELECT program was changed to CDS mode it allowed a 20-page buffer, resulting in an A400 search time of 13.7 seconds in Case 1, 40.3 times faster than the DBGETTEST program.

The A900 test (Case# 5) appears to show performance lower than some of the A400 tests, but this is not actually so. The 497K bytes of data represent actual filled records, but the population of filled records in that particular data set is sparse due to a large number of deletions which created empty records. An examination of the data set file indicates that the A900 test actually had to scan 4,364K bytes in the 11.6 seconds, which is more data than the A400 scanned in 17.6 seconds.

In terms of scan speed, the A900 (Case 5) scanned at 376,207 bytes per second in the QUICK test and the A400 (Case 1) scanned at 227,273 bytes per second, suggesting that the A900 performs this task some 1.65 times faster than the A400. Most of this difference is probably due to the faster CPU, but some of it may relate to disk performance (7914 versus 7958) and some to the fact that empty records can be scanned slightly faster than filled ones because no masking is required.

The performance improvement provided by the A900 over the A400 for the QUERY and DBGET tests is even greater, amounting to a factor of 3.04 for QUERY and 3.95 for DBGET. Because searches using DBMON are more compute-intensive than the QUICK method, they can take better advantage of the A900's faster CPU. All of this suggests that the very largest performance improvements for the QUICK method over the others will be on small CPU's with large data bases. The smallest improvements will be on fast CPU's (A900's and especially A990's) with data bases less than a megabyte in size. Even then, the available improvement factor of 2 to 10 may be quite valuable to those users whose jobs call for frequent analysis of the application data.

When the application program reads the data base directly without going through IMAGE, it is still necessary to enforce IMAGE's ability to refuse access to a second application program if a first program has opened the data base exclusively. The accelerator handles this problem straightforwardly by requiring the data base to be opened in a valid read mode. It will return IMAGE errors if called with the data base closed.

Note that there is never a problem with failure to honor data base locks imposed by another program, because locks only affect writes and updates. Reading is always allowed regardless of locks.

The test results do show interesting discrepancies between the speed of QUERY and that of the DBGETTEST program. In most tests QUERY is actually significantly faster than DBGETTEST. One might assume that QUERY uses DBGET just like any application program would, so the speeds ought to be about the same. Does QUERY actually perform direct calls to DBMON, bypassing DBGET to achieve an improvement through overlapped requests or some other enhancement technique? If so, why does that not work in Case 2? One can only speculate.

<u>SUMMARY:</u>

When frequent data extraction is desirable, the type of accelerator software described is likely to be of substantial value in almost all cases where the size of the data base exceeds 250K bytes or so. The value will be at least proportional to the size of the data base. The improvement will be even greater on low power CPU's than on high power ones.

Paper # 1004

Using the Mouse for Screen Data Entry on the HP1000

Donald A. Wright
Interactive Computer Technology
2069 Lake Elmo Avenue North
Lake Elmo, MN 55052 USA
612/770-3728

Abstract:

Most of us have become accustomed to using the mouse for many of our PC applications. The mouse provides an "analog" mechanism for moving the cursor and locating functions or items, more intuitive for many people than the keyboard with its cursor keys.

In the past a mouse has been offered as an attachment for some HP terminals and was even supported on the HP1000 as a graphic input device. However, it was never supported as a non-graphic screen data entry device. Today most mice are found directly attached to PC's and not to multi-terminal minicomputers like the HP1000. They are supported by programs which run on the PC but not by programs running on the HP1000.

This paper describes procedures and software which allow HP1000 programs to use mouse input to control the cursor and make choices on forms-mode data-entry screens on the HP1000. They allow HP1000 users to modernize their screens in a very impressive fashion.

Overview:

The mouse used in the development of this paper is the 3-button Logitech "Mouseman" device, model number M-CJ13-9F, Serial and Mouseport Version, commonly available in virtually all computer stores. A mouse application requires two MUX or OBIO ports; one for the mouse and the other for the terminal, which may be any model of HP terminal. With most MUXes it is necessary to make a special supply connection to provide the power that the Mouseman needs to operate.

The software used with the mouse allows either the mouse or the cursor keys to move the cursor and also allows the mouse buttons to duplicate the function of any three of the keyboard softkeys or to provide new functions. Thus an existing application can simply be enhanced with the mouse to provide intuitive motion and function selection, and new applications can be developed which make use of other intuitive mouse functions such as pick-drag-and-put.

Protocols:

The Mouseman is capable of using several different protocols and operating at either 1200 or 9600 baud. A very serious attempt was made to use the default protocol and baud rate (two or three-button Microsoft Compatible protocol, 1200 baud), because this would have allowed use of virtually any brand of mouse. However, neither the Mouseman nor a standard Radio Shack mouse could be made to work with the HP1000 MUXes. No matter how the MUX was configured there seemed to be a problem with in the number of bits in the character, causing framing errors at the HP1000.

Switching to the "Five-Byte Packed Binary" protocol solved this problem. 9600 baud was also tried, as this would significantly reduce the problems with the shortage of baud rate generators on 'C' and 'D' muxes, but the Mouseman can transmit up to 500 characters per second at that baud rate and this overflowed the OBIO input buffer unless an inordinate amount of attention was paid to keeping it clear.

The configuration chosen was Five-byte Packed Binary at 1200 baud. The MUX is configured for 8 data bits, one stop bit, and no parity. The five 8-bit bytes received from the mouse at every dot of movement and every button press are defined as follows:

	Bit #	7	6	5	4	3	2	1	0
Byte #	1	1	0	0	0	0	L	M	R
	2	X7	X6	X5	X4	X3	X2	X1	X0
	3	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
	4	X7	X6	X5	X4	X3	X2	X1	X0
	5	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0

Where:

L, M, and R represent the Left, Middle, and Right keys, 0 = TRUE. X0-X7 and Y0-Y7 are the 2's complement delta position in mouse dots. The second set of values represent the delta occurring <u>during</u> transmission of the first report and is NOT a duplicate.

Configuration:

The Mouseman interprets the 1200-baud two-character command '*U' at its input port as an instruction to select the Five-Byte Packed Binary (MSC) protocol. This command is normally issued by the MOUSEINIT subroutine, described later, when the application is started.

Hardware Connections:

The Mouseman, like other serial mice, gets its power from the RS-232 serial line. It has a 9-pin (DB9S) connector which you will want to convert to 25-pin DB25 using a standard adapter. Logitech documentation and some experimentation suggests that these four pins are absolutely required:

DB25	DB9S	Function
2	3	Negative voltage and serial data host to mouse.
3	2	Serial data mouse to host.
4	7	Positive voltage host to mouse (RTS).
7	5	Signal ground.

Four-Port OBIO Connections for A400:

Connect the mouse to port B or C (one of the two modem ports) and switch it to the DTE position (the mouse is wired as a DCE).

B-, C-, and D-MUX Connections:

A reversal of pins 2 and 3 is required for connection to the 8-port MUX, and it is also necessary to supply a positive voltage of 5 volts or so at pin 4 (DB25) from some source other than the MUX, because there is no positive voltage available at the DB25 connectors of the newer MUXes. The Mouseman appears to draw about 3 mA from the negative supply (DB25 pin 2) and about 15 mA from the positive (DB25 pin 4). It is also possible to provide the positive voltage at pin 20 instead of pin 4. The exact voltage seems to be quite uncritical.

As you can see, the 4-port OBIO is MUCH easier to use if your CPU happens to be an A400. The OBIO also has none of the the baud-rate generator problems of the C- and D-MUXes.

MOUSE Library:

There are just three subroutines in the little library of contributed routines which control the mouse itself:

MOUSEINIT
MOUSEREAD
MOUSECALC

MOUSEINIT determines the type of MUX port in use for the mouse, issues all necessary initialization requests including the 1200-baud 30B request, switches the mouse to the specified non-default protocol, and makes the first call to MOUSEREAD. Note: At this writing the C-MUX portion of the MUX library code is in place and may work correctly but it has not been tested.

MOUSEREAD issues a class read on the mouse LU and returns a flag to the caller indicating whether or not there is actually data available for a class GET. This subroutine is called from both MOUSEINIT and MOUSECALC and probably will not be called directly from the application.

MOUSECALC does the biggest job. It examines the characters received from the last Class Get on the mouse LU, moves the cursor if necessary, and returns the edges of the button positions. It applies hysteresis to the mouse movement information to prevent jittering at a boundary location and then determines whether cursor movement is required. If so, it flushes the current read on LU 1 and sends the cursor movement command, but does not re-issue the read. The calling application must be prepared to deal with the empty read completions caused by this flushing and to re-issue new reads.

All cursor movement commands issued to LU 1 are relative rather than absolute movements. The mouse software does not know the absolute position of the cursor, because it is possible for the operator to move it with the cursor keys as well as with the mouse.

MOUSECALC contains four compile-time parameters which allow selection of the vertical resolution in mouse dots per line, horizontal resolution in mouse dots per column, and hysteresis percentage in both dimensions.

Some mouse software contains a ballistic mode which allows faster cursor movement for the same mouse distance when the mouse is moved rapidly. MOUSECALC does not contain a ballistic mode, but it would certainly not be hard to add on the real-time HP1000.

Screen/Forms Management:

The mouse software can probably be used in combination with any screen package, including commercial packages such as Forms/1000 or your own homebuilt package. There is one unusual requirement: The terminal reads must be done by class I/O and the class number must be available for use also for mouse LU reads. This allows the application to wait on a single class for input from either device, and to process whatever comes in. With either commercial or homebuilt forms packages it will be necessary to write new terminal I/O routines to deal with this issue.

In addition to the class I/O requirements, a few of the other functions which should be in the screen package are:

> **Named Fields**: The data input fields should be addressable by name, for programming convenience.

Read/Write: Normal reads and writes to named fields must of course be available.

Changeable Enhancements: It should be possible to change a field's enhancements in response to an operator input; to cause it to blink, or change its color, etc.

Cursor Position Sensing: Since the cursor may be moved either by the cursor keys or by the mouse, it is a must to be able to sense its position. In addition, it is helpful to have an algorithm which will tell the software the name of the field nearest the cursor.

Mouse Functions and Programming Hints:

A few of the functions which can be performed on a mouse at least as cleanly and intuitively as the keyboard are:

Cursor Movement. Clearly this is one of the biggest advantages of the mouse, as movement can be accomplished smoothly in any direction.

Circular Attribute or Value Selection. When a field has a fixed range of possible attributes or numeric values, the mouse can be used to rotate through those at the push of a mouse button until the desired value appears in the field. This may be much more convenient for an operator than keyboard entry, because the application will present only valid choices as the operator rotates through them. In some cases it may be desirable to provide attribute rotation or value changes in both directions, similar to the linear increment/decrement described below.

Linear Value Increment/Decrement: When a field's possible values are a large set of integer numbers, one mouse button may be used to increment the field value and another to decrement it. In such a case it will also be desirable to permit direct keyboard input of the field value for those who would rather do it that way.

Pick-Drag-and-Put: Every PC user is accustomed to the kind of user interface function which follows this sequence: a) The mouse is moved so that the cursor is positioned on or near a desired attribute or value; b) A mouse button is pressed to select that attribute, and the attribute itself is then highlighted or enhanced in some way to provide operator feedback; c) The mouse is moved to the field where the attribute or value is to be placed; d) A mouse button is pressed to cause the software to insert that attribute or value in the field.

The value actually appears in the destination field and the enhancement of the source field is ended. This works just as well on the HP1000 as it does on PC's, except that the cursor cannot easily take on a graphic form so the operator sees the cursor itself dragging across the screen to the destination rather than a graphic icon. This is not a drawback if the attribute has been enhanced.

Two of the basic principles of mouse-input programming are:

Minimize confusion: The functions of both the mouse buttons and the terminal softkeys should be standardized as much as possible throughout all programs in the application. For example, the same button and/or softkey should always be used to select a common action such as SAVE SCREEN or PREVIOUS SCREEN. It may also be helpful to give the three mouse buttons the same action as three of the function keys, always the same three, and to highlight those function keys in some special way so that their association with the mouse buttons is intuitive. In our demo, described later, we made the three softkey labels three rows high instead of two, and also applied a different color to them.

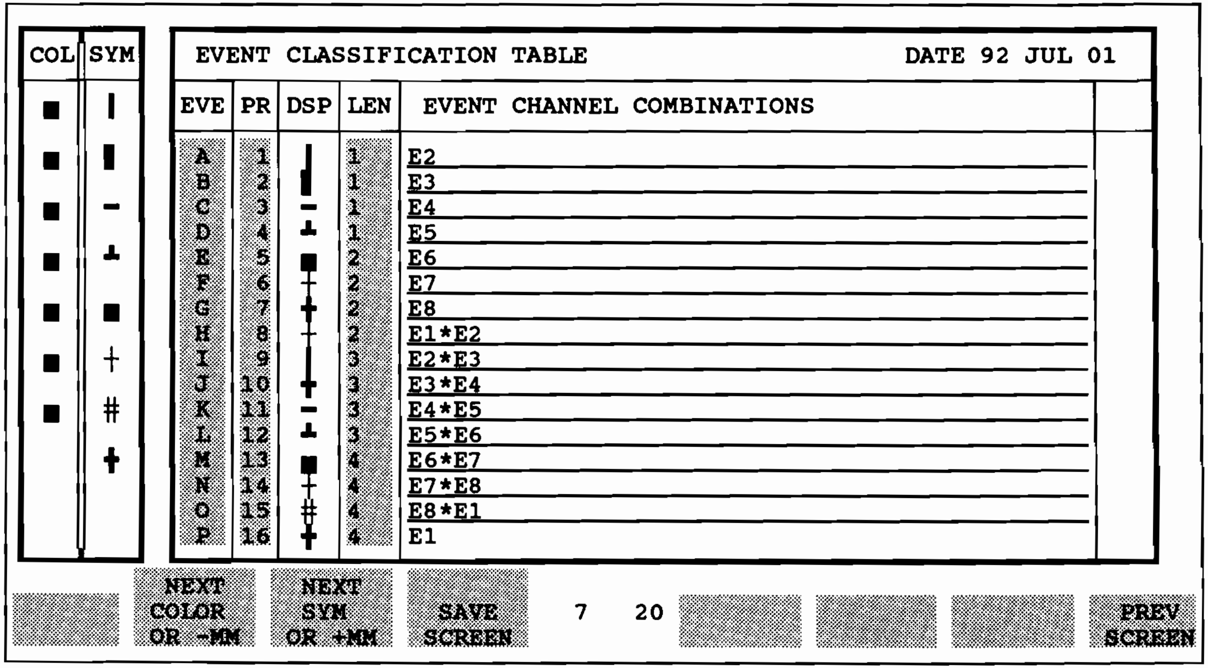
Provide Immeidate Feedback: When the operator has taken an action with a mouse button, something should change on the screen right away so that the operator knows intuitively that the action is accepted. Normally the change should be associated with the object most involved with the desired action.

A Practical Example:

A demonstration application was developed using the contributed MOUSELIB package, ICT's own internal screen package, and about 300 lines of actual application code. The application assumes that the real-time HP1000 is receiving events from some external source, and it allows the operator to develop a table for assigning the following attributes to combinations of those events:

1) The color for the display of an event combination.
2) The symbol used to display the event combination.
3) The length (duration) of a valid event combination.
4) The relative priorities of the various combinations.

Please refer to Figure 1 for the actual screen used in the demo. In actual practice it is in color, and it does lose quite a lot of impact in black and white.



COL	SYM
■	|
■	▮
■	–
■	┴
■	■
■	+
■	#
	✚

EVENT CLASSIFICATION TABLE DATE 92 JUL 01

EVE	PR	DSP	LEN	EVENT CHANNEL COMBINATIONS
A	1	|	1	E2
B	2	▮	1	E3
C	3	–	1	E4
D	4	┴	1	E5
E	5	■	2	E6
F	6	+	2	E7
G	7	✚	2	E8
H	8	+	2	E1*E2
I	9	|	3	E2*E3
J	10	✚	3	E3*E4
K	11	–	3	E4*E5
L	12	┴	3	E5*E6
M	13	■	4	E6*E7
N	14	+	4	E7*E8
O	15	#	4	E8*E1
P	16	✚	4	E1

Figure 1: Mouse Demonstration Screen with Color (COL) and Symbol (SYM) attributes to the far left, fixed EVEnt Name and PRiority columns, and changeable DSP, LEN, and Event columns. Mouse buttons correspond to f2, f3, and f4.

The first two attributes, the color (COL) and symbol (SYM) for the Display (DSP) column, may be selected either by circular selection or by a pick-drag-and-put procedure. Either method may be accomplished through the mouse and its buttons or by the cursor keys and softkeys, or by any combination of both. When the pick-drag-and-put procedure is used it is possible to pick both the color and the symbol, drag them to the DSP column, and put them both down in one field.

When a color in the COL column is picked, it expands to three times its original width and blinks, as immediate feedback to the operator that a pick was made. When a symbol in the SYM column is picked, it changes from white to the last-picked color and it blinks. When the selected color and/or symbol are Put in the DSP column, those special enhancements disappear and the new symbol with its color appear immediately in DSP.

The Length (LEN) attribute may be an integer value ranging from 0 through 999. It may be altered by incrementing or decrementing the selected field with mouse or the softkeys, or it may be keyed in directly.

The actual event combinations associated with the symbols and lengths are selected through keyboard input. Here the mouse is not used except possibly to position the cursor to the correct screen field.

Additional Information:

Logitech provides a LOGITECH Mouse Technical Reference and Programmer's Guide which is of some value to people needing to understand the various modes in which the mouse can operate, though it of course focuses almost exclusively on PC applications. It has two chapters: 1) Hardware; 2) Software. It covers all Logitech mice, and therefore would probably be useful for any compatible mice as well, which is nearly all mice ever made. The manual can be obtained upon request and payment of a modest charge. Call Logitech Technical Support at 510/795-8100.

Summary:

A high-quality three-button mouse can be used in combination with standard HP terminals and keyboards, providing an impressive modernization of the look and feel of applications on the HP1000.

What to Expect from RTE 6.0

Esther A. Heller
Rebecca L. Carroll
Hewlett–Packard Co.
Software Technology Division
11000 Wolfe Road, Cupertino, CA 95014

INTRODUCTION

The 6.0 release of RTE will feature many user enhancement requests, in addition to some new functionality. Enhancements include changes in the user's environment, extended capabilities for EMA/VMA, new utilities such as ls and grep, improvements to Mail/1000, and enhancements to AGP/DGL, networking, and Debug/1000. This paper will serve as the user's introduction to the 6.0 release.

CI Enhancements

At 6.0, the concept of a user's environment is greatly expanded. Command aliases and functions are being introduced. Both of these items and CI variables can be exported (placed in the Environment space) and thus, be available to all processes running in a given session. Several customers will find their requested needs met by export capabilities. In addition, several new variables are being added to enable the user to have more flexibility in giving commands and in Command Stack Editing. Unless otherwise noted, these enhancements are ONLY available in the CDS version of CI (92078A product).

Command Aliases

Command aliasing is a method which allows a user to create new commands or cause standard commands to perform differently by replacing the original command line with a new command which is called an alias. The new command can be a letter or short word which when typed will be expanded by CI into the alias value. Anything which followed the alias in the original line now follows the expanded value. The commands can be entered interactively from CI or via a CI command file. The syntax for this command is:

$$\text{alias}\ \ [-\text{x}\,|\,+\text{x}]\ \ [\ \text{aliasname} \left[\left\{ \begin{matrix} '\ ' \\ '=' \\ '\ =\ ' \\ ',' \end{matrix} \right\}\ \ \text{stringvalue} \right]\ \ ]$$

For users who are spending time on both RTE and HP-UX systems, aliases can make the transitions back and forth easier. For example,

```
alias man ?
alias less li
```

Of course, a user could always do similar aliasing on the HP-UX side if she or he prefers the RTE command names.

The command UNALIAS is used to unset an alias which is no longer desired. The syntax is:

```
unalias aliasname
```

The command ALIAS by itself will display all defined aliases. The command "ALIAS foo" will display just the alias foo by itself. This is comparable to "SET" and "ECHO $foo" for user variables.

Functions

Functions are similar to aliases. They can be thought of as memory resident command files. Functions can include positional parameters and IF-THEN-ELSE or WHILE-DO-DONE constructs. They can be entered interactively from CI or via a CI command file. The syntax for functions will be:

```
function [-x|+x] functionname {
string1
string2
.....
stringn
}
```

If a function is entered interactively, the user prompt will change to indicate what needs to be entered in order to terminate or close the function. This is also the case for the conditional constructs. See example. The closing brace must be the first character on a line followed by a space, comma, semi-colon or carriage return to be valid. If followed by a semi-colon, the next command will be executed. If it is followed by a space or comma, the remainder of the line will be ignored. Here is an example as it would appear on the user's display:

```
CI> function flink {
} > if ftn7x $1.ftn - -
THEN > then
FI > link $1.lod
FI > else
FI > li $1.lst
FI > fi
} > }
```

Incidentally, these prompt changes will occur if a user does an interactive IF-THEN-ELSE construct. Also, syntax checking is done in the definition of a function (as is done in CI transfer files) and if the function does have errors, such as the "THEN/ELSE must precede FI!" error, the function will not be saved.

The command FUNCTIONS is used to display all defined functions. The command UNSET with the −F option is used to remove a function.

Exporting Variables, Aliases and Functions

Many users have requested a way to use CI's user variables in other programs. Also, once a user has defined an alias or function, there should be a way for a secondary CI to use it, without having to define it again. The answer is exporting items to the Environment Variable Block (EVB). The astute reader will have already noticed the [−x|+x] syntax in the definitions of ALIAS and FUNCTION. At 6.0, the syntax for SET also changes:

$$\text{set } [-x|+x]\ [\text{ varname} \left[\left\{ \begin{array}{c} \text{' '} \\ \text{'='} \\ \text{' = '} \\ \text{','} \end{array} \right\} \text{stringvalue } \right]]$$

First observe that the "=" sign in the syntax is no longer required. So that:

set foo = bar
set foo bar
set foo,bar

all accomplish the same assignment. Also, at 6.0, variable, alias and function names can be 32 characters long. If having defined $foo to be bar, the user now gave the command: set −x foo. The variable foo would be removed from the "local" CI variable space and be "exported" into the EVB. A variable (or alias or function) can be exported at definition such as:

alias −x man ?

It is possible to have an alias or a variable with the same name but a different value in both the local space and the EVB. This could be of use in some situations. For example, the user has several processes running in the background which do logging based on $LOG. He or she would like all the processes to do the logging, but doesn't want the primary CI log function enabled. In the case of both a local and an exported copy existing, CI will always access the local copy. In this area, functions differ from aliases and variables. Only one version of a function can exist.

A variable, alias or function can be "imported" or explicitly placed in local space with the +X option. For example, set +x foo would remove $foo from the EVB and place it with its current value in local space.

Where is the EVB?

The EVB is kept in SHEMA. The amount of environment given to an individual user is configured by the system manager using GRUMP. In the sequence of GRUMP commands, after the "Enter #UDSPs:depth [0:0] :", there is now a command "Enter the size of the Environment Variable Block in pages [0] :". If a user's information is not modified, she or he will not have an Environment Variable Block available. Otherwise, one block will be allocated for each session of that user. This SHEMA block will show up via WH as:

```
Program                DataPartition CodePartition
Name    Prio   PC   Seg  Size Status   Size Status  Program Status
------------------------------------------------------------------------
Session  90  Capability 31  Superuser ESTHER
CI       51  6116c   1    28 in          61 sh      waiting for WH
             Shared EMA used: Environment 90
WH        5 12277         18 in                     scheduled
------------------------------------------------------------------------
Mon Jun 29, 1992   4:06 pm
```

Another consideration for System Managers will be resource numbers. EVB's will each have a resource number allocated to them. Thus the system generation should include another RN per session for the maximum number of sessions genned in.

Some customers have requested that we allow more space for user defined variables. Obviously the EVB accomplishes this. In addition, CI's own local space is now configurable by the system manager on a per system basis. The local variables (and aliases and functions) are now being kept in local EMA space. In the CI load file (now being called CI.LOD), there will be a LINK command:

```
em,,l
```

If CI.LOD is left unmodified, each user running the /PROGRAMS/CI.RUN will get two pages of local space. Previous versions of CI essentially gave one page for variables. To change this, simply edit the load file and relink. For example,

```
em,4,l
```

would give each user's CI four pages of local variable space.

Programmatic access

Individual exported variables can be accessed programmatically by name using the new EXEC 39 calls. There are four subfunction codes being provided at this release:

code = 1	Get the value of a variable.
code = 2	Define a variable with the given value.
code = 3	Delete a variable.
code = 4	Retrieve the modification count.

The calling sequences look like:

```
CALL EXEC (39, 1, name, value)
CALL EXEC (39, 2, name, value(1:vlen))
CALL EXEC (39, 3, name)
CALL EXEC (39, 4)
```

"name" is a FORTRAN string descriptor that specifies the name of the variable to be retrieved/defined or deleted. "value" is a FORTRAN string descriptor the receives/specifies the value of the named variable. The A-register return is used for status information. For code = 2, the B-register contains the length in characters of the value. For code =4, the B-register contains the modification count. The modification count of the EVB is incremented whenever any process begins to write into the block and then again when the write is completed. CI regularly checks the modification count against a copy that it keeps. So, if CI detects a change, it does any necessary updating to variables which it uses, such as $LOG or $AUTO_LOGOFF. Code = 4 is provided to allow programs to do the same kind of checking and updating.

Programmers should be aware that aliases and functions are stored in the EVB as if they are variables but with ":" as part of their name. The EXEC calls don't check for valid characters as CI does. So, a program can put other "illegal" characters in the names. CI won't do any expansion on them (such as "ECHO $foo%"), although they may be visible via the "SET" command. .

New Variables

The following variables are now predefined by CI: $COLUMNS, $DATC, $HOME, $IFDVR, $LINES, and $OLDPWD. In addition, $REPROMPT and $VISUAL are initially undefined but once defined have significance to CI. The following variables are automatically exported on start-up: $COLUMNS, $LINES, $HOME, $OLDPWD, and $WD. The last three variables must remain exported. $PROMPT can now have a value of up to 72 characters.

The cd Command and tilde Expansion

In addition to the WD command, CI versions of cd and pwd, commands familiar to HP-UX users are being implemented. $WD, $OLDPWD and $HOME are used in conjuction with cd and tilde expansion. When a user first logs on to a session, $HOME is set to be the start-up directory. If the user has an EVB, $HOME is exported. If the user then runs the path program to set up UDSP #0, path will reset $HOME to the given directory. Users who do not have Environment Variable Blocks can still use this feature by manually setting $HOME to their home directory. Also, this enhancement is available in the non-CDS version of CI, in a more limited version.

The previous VC+ command CD, used to display or modify CDS code partition size, has been renamed to CZ, to accommodate those users who need this functionality.

The CD command can take either of two forms. The syntax for these two forms are:

```
CD [−p] [argument]
CD old new
```

In the first form, it changes the current directory to "argument". If "argument" is '−' the directory is changed to the previous working directory ($OLDPWD). The default for "argument" is the value of the $HOME variable. The second form of cd substitutes the string "new" for the string "old" in the current directory name, $WD, and tries to change to this new directory.

When either the CD or WD command is used, the variables $WD and $OLDPWD are updated. The WD command always sets $WD to the physical name of the current working directory. When $VISUAL is set to any of the ksh-style modes (emacs, gmacs, or vi), the CD command will preserve the logical naming when treating symbolic links. For any of the other $VISUAL modes, the CD command will save the physical name of the current working directory in $WD. See below for more information about $VISUAL and the different ksh-modes. These are only available in the CDS-version of CI. If $WD is changed, $OLDPWD will contain the prior value of $WD.

Tilde substitution involves substituting values of certain variables for the character "~" in a file name. A "~" by itself is replaced with the value of $HOME. "~ +" is replaced with the value of $WD. "~ −" is replaced with the value of $OLDPWD. For example, $HOME = /EAH. Then, the command "set foo= ~/bar" would result in $FOO = /EAH/BAR. In order for a ~string to be expanded, it must occur at the beginning of a parameter; also, it must either by at the end of the parameter or be followed by a ",". Thus, "~bar" would not be expanded.

File Name and Command Name Completion

The file name completion for CI is based on the file name completion features of the UN*X shells (csh and ksh). The idea is to reduce the need for typing entire file names as arguments to commands. The user types enough of the name to be unique and CI will be able to fill in the rest, echoing the full name to the screen. There are two different modes, the csh-mode and the ksh-mode. In the csh-mode, the file name completion is triggered by pressing the ESCAPE key. In the ksh-mode, the completion is triggered by pressing the ESCAPE key twice. Pressing it once and then the "=" key will produce a list of what files in the directory match up to this point. This directory listing is done in the csh-mode via a CTRL−D. In either mode, if there is no file that's name matches, the terminal's bell is sounded. If there is no unique match, the name will be partially completed, up to the point of deviation and the bell will sound.

The program CMPLT performs all of the command and file name completions and the in-line directory listings. It should be RPed in the Welcome file.

Command name completion works much the same way as file name completion. In this case, CMPLT searches CI's internal commands, and the current values of the user's UDSP #1 and UDSP #2 to find executable and CI command files which might match whatever has been typed.

$VISUAL

The new variable $VISUAL, which may be either local or exported, is used by CI to determine which file name completion mode the user desires. This variable is further used to determine which style of command editing is desired. The values and their usage are:

```
set visual = emacs    ! ksh style emacs command editing

set visual = gmacs    ! ksh style emacs command editing with gnu ^T.

set visual = vi       ! ksh style vi command editing

set visual = csh      ! csh style name completion / directory lists.
```

For full utilization of the new editing capabilities (mentioned below) , setting visual causes the CI command stack to have duplicate entries. To have one of these modes without the duplicates, there is a nodupes option to $VISUAL. The syntax looks like:

```
set visual = vi,nodupes
```

In HP-UX, command completion is only available in csh. In RTE, it is available in all of the modes listed above.

Command Editing Capabilities

The UN*X Korn shell (ksh) allows the user to specify in-line editing options. These editing features, which allow the user to edit the command line before executing a command, have also been added to CI. These editing options are only available in the three ksh modes indicated as values of $VISUAL above.

The editing modes enable the user to look through a window at the current line. The default window width is 80, unless the value of COLUMNS is defined. If the line is longer than the window width minus three, a mark displayed at the end of the window notifies the user. The mark is a > (<, *) if the line extends on the right (left, both) side(s) of the window. As the cursor moves and reaches the window boundaries, the window is centered about the cursor.

The search commands in each edit mode provide access to the history file. Only strings are matched, not patterns, although a leading ^ in the string restricts the match to begin at the first character in the line.

In the "csh" mode, the "file name completion", "directory list", "command name recognition" and the "command line control" csh utilities are available. The csh style of command history substitutions is not available.

Extended EMA/VMA

Since this subject has been discussed in a previous Interex paper by Dave Medlicott and Todd Poynor, it will only be summarized here.

EMA/VMA Models

RTE-A now has three E/VMA "models":

- The Normal E/VMA model refers to the existing E/VMA scheme.
- The Large E/VMA model supports some extension of E/VMA functionality and can be used on non-A990 processors.
- The Extended E/VMA model makes use of the A990 microcode to provide the full extension of E/VMA functionality; *this is only available for A990 processors.*

New Features

The new features are summarized in the tables below:

Capability	Pre-6.0 RTE-A	6.0 RTE-A
Extended or Large E/VMA	NO	YES, *extended for A990 only*
ID segment size	48 words	45 words
ID segment extensions	NO	YES
Max no. SHEMA partitions	15	Dynamic, by free XSAM size
Length of SHEMA labels	6 characters	16 characters
Max no. programs using a SHEMA	63	255
Ability to have systems without SAM	YES	NO

Capability	Normal	Large	Extended
Max size of any one EMA in pages	1022	1022	32733
Max size of a VMA working set in pages	1022	1022	32733
Max no. of SHEMAs a program may access	1	64	64
No. SHEMAs accessible when local E/VMA used	0	63	63

grep

grep, along with the utility fgrep, is new for the 6.0 release. grep and fgrep search files for lines matching a certain pattern, much the way the UN*X version of grep works. The difference between the two utilities is that grep supports regular expression patterns (similar to EDIT/1000), while fgrep supports fixed strings (making it a fast and compact way of finding text strings). As matches are found, they are copied to the session LU. Both grep and fgrep are shipped with the RTE-A operating system.

The output of grep and fgrep can also be redirected to a file. This is specified in the runstring by either '>filename' or '>>filename', where the first causes output to be redirected to a non-existing, specified file and the second causes output to be appended to a specified file (existing or not). The utilities take either a file or a mask as input. Usage is as follows:

```
grep  [options] [expression] file|mask ...
fgrep [options] [string]     file|mask ...
```

The following example will search all files ending in .FTN or .BAK for any line containing 'if', 'then', or 'endif'. Lines will be displayed with the line number, and file access errors will be suppressed. Note that when the '-e' or '-g' option is used, the command line may not contain the expression/string argument.

```
CI> fgrep -e if -e then -g/endif/ -ns @.ftn @.bak
```

The next example will search the grep help file for lines matching the regular expression given and write the output to the file 'outfile'. Note the expression contains spaces and is case sensitive, requiring the '-u' option and back quotes to protect the expression from CI.

```
CI> grep -u >outfile '-[a-z] [e/f]` /help/grep
```

The last example will search files matching the mask '-o' for the '-e' pattern. Note that the '--' flag is required when the pattern contains a leading hyphen.

```
CI> fgrep -- -e -o
```

ls

ls and its related commands will list the contents of a directory on RTE-A. ls will report the name of any file matching the mask along with any other information requested. If no mask is supplied, the current working directory will be searched. The output is sorted in ascending collation order.

The output of ls can be redirected to a file. This is specified in the runstring by either '>filename' or '>>filename', where the first causes output to be redirected to a non-existing, specified file and the second causes output to be appended to a specified file (existing or not). If no output file is specified, ls will break the output into screen pages; paging is disabled when any output file is specified. The redirection string must be delimited by commas and is position independent. Multiple redirection strings may occur in the runstring; however, only the last redirection will be executed. Usage is as follows:

```
ls [-CDGLNOPRTUXYZcfl1 ] mask [ mask2 ... ]
ll [ls options] mask [ mask2 ... ]
lsf [ls options] mask [ mask2 ... ]
lsx [ls options] mask [ mask2 ... ]
```

The above notation uses "shorthand", and the shortcuts correspond to the following:

```
ll  --> ls -LY
lsf --> ls -f
lsx --> ls -X
```

Note that the options to ls are case sensitive. To specify a lower case option, the user must quote the option character to prevent CI from upshifting the character. See the following example:

Print a long listing of all the files in the /PROGRAMS directory. List the most recently modified (youngest) file first, followed by the next older file, and so forth, to the oldest. For symbolic links, follow the link and use the update time of the destination file. Append the output of the ls program to the file 'outfile' in the current working directory.

```
CI> ls -l\lt >>outfile /programs/
```

Since the default behavior of CI is to casefold the entire command line, *the options of the RTE version of ls are the inverted case of the equivalent options for a UN*X-based ls program.* (Note: the most commonly used options do not require quoting.) In the example above, a backslash '\' character was used to protect the lower case 'l' from CI.

There are two major listing formats for ls. The format chosen depends on whether the output is defaulted to the user's tty LU and can also be controlled by option flags. The default format for the user's tty device is to list the contents of directories in multi-column format, with entries sorted vertically by column. If the output is specified, the default format is to list one entry per line. The -c and -X options enable multi-column formats. In order to determine output formats for the -c, and -X options, ls uses an environment variable, COLUMNS, to determine the number of character positions available on each output line. If this variable is not set, 80 columns is assumed.

Symbolic Links

A symbolic link is a type of file that indirectly refers to a path name, which can be either a relative or absolute path name. A symbolic link can refer to any FMP file, FMP directory, or logical unit. Symbolic links can also be used to refer to remote files by using the DS transparency syntax.

For more information, please see paper #1010, Symbolic Links on RTE-A / VCPlus by Gary Gan.

Mail/1000 Enhancements

Mail User Interface

Many changes have been made to allow more flexibility in file and folder handling, visual-mode presentation options, etc. For example, at 6.0, the "ignore" command arguments look like the following:

```
mask[:lines]
```

where <mask> is a pattern which matches header field names, and <lines> is the maximum number of lines to print of matching headers (from 0 to 255, default = 0). For example, "ignore to:3" prints up to 3 lines of "To:" header and suppresses any following lines. Many other features will be found at the new release.

DNS Client Support

Client support for the Domain Name Service (DNS) will be provided at 6.0. If a customer's network is running a nameserver, then Mail/1000 may be configured to query that nameserver for mail routing and IP address info, in accordance with RFCs 1032-1035, 1123, etc. For more information, please see paper #1009, What's in a Nameserver? by Todd Poynor.

Other Mail-Standards

Sendmail will replace domains specified in message headers which are aliases for an official domain to the official name, as per the RFC-822 and RFC-1123 standards.

The mail system may now be configured to generate "From:" and "Sender:" addresses in the syntax

```
Real Name <local@domain>
```

instead of the default:

```
local@domain (Real Name)
```

Host Routing

In addition to DNS client support, some other changes have been made to support unusual network setups and such. For example, at 6.0, the file /mail/admin/domainalias.cf may contain entries in the format:

```
fully.qualified.domain : |runstring
```

Here, runstring is a string suitable for FmpRunProgram which will be used to forward mail bound for fully.qualified.domain. The name of the temporary file which contains the message will be appended to the runstring. The message file will contain the routing envelope information at the top; this file is suitable for copying directly into another host's /mail/queue/ directory with name QMSG_x.QIN for further Mail/1000 processing.

New Mail Notification

The 6.0 release will contain many enhancements in the area of new mail notification, including the ability for individual users to select different notification means based on message subject, sender, etc. Additional features include the following:

- "notify off" turns off notifications for your session; "notify on" restores notifications. All messages sent while notification is "off" are thrown away.
- Session numbers may be used in place of logon names to direct messages solely to a single session. For instance, "notify 90 'I'll be back.'" sends the message only to session 90.
- Special user name "all" sends the message to all logged-on sessions.

Installation Process

The InstallMail.cmd file uses CI variables to determine the directories in which software will be installed, much like the NS-ARPA/1000 installation file.

uuencode/uudecode

The uuencode/uudecode utilities prepare a file for transmission via mail. Usage is as follows:

```
uuencode [-a] [-d remotedest] input output
uudecode [-a] filename [output]
```

Uuencode takes the named source file and produces an encoded version; this version is in ordinary text form and can be edited by EDIT/1000. The encoding uses only printable ASCII characters; mode of the file and <remotedest> (for re-creation on the remote system) are included. The '-a' option causes RTE ASCII files to be translated to UN*X ASCII before the file is encoded (a newline character is inserted between each record). For files other than type 1 files or translated

ASCII, the "remotedest" will include a full file descriptor including the file type and size.

Uudecode reads an encoded file, strips off any leading and trailing lines added by mailers and recreates the original file with the specified mode and name. The '−a' option causes uudecode to translate UN*X ASCII to RTE ASCII. (Newline characters are stripped out and interpreted as record separators.) If the "remotedest" does not include a file type, uudecode will by default create a type 1 file. If the '−a' option is supplied, the data will be decoded and translated to a type 4 ASCII file.

Enhancements to Other Software

In addition to the changes above in RTE-A and VC+, there are also notable enhancements for AGP/DGL, networking, and Debug/1000 software.

AGP/DGL

There has been a lot of work put into the AGP/DGL software for 6.0, but the highlights have to do with accessing LUs greater than 63 and new handlers.

LUs > 63

The change to have AGP/DGL support LUs greater than 63 has been much-requested and will be implemented for the 6.0 release.

HP-GL/2 Handler

A new handler is required to drive new plotters and printers that support HP-GL/2. This handler will be supplied at 6.0 and will meet the needs of future graphics peripherals supported on the HP 1000. For specific information on this handler, see last year's Interex paper by Erin Solbrig.

PaintJet Support

A new driver will be added to support the following devices:

3630A PaintJet
C1602A PaintJet XL without C1608A HP-GL/2 cartridge
DeskJet 500C

Networking

FTP

Several enhancements will be in 6.0, including the following:

- FTP will transfer files between 6.0 systems with all the file descriptor attributes intact; the user need only specify the filenames. Destination files will have the same attributes as the source files.
- GET/PUT and MGET/MPUT will now display the name and attributes of the files being transferred.
- FTP binary file handling will be improved. Binary file transfers to and from unknown remote systems will use a 256 block extent size instead of the current 24 block size.

There are also several new commands at 6.0. A brief listing is below.

- DL: creates a detailed listing of the contents of a remote directory in RTE-A DL format.
- NLIST: creates a listing of the contents of a remote directory in HP-UX format.
- RTEBIN: makes transfer of binary files easier by setting the transfer type.
- SITE: performs server-specific services.
- SYSTEM: shows remote system type.

Other changes

Other enhancements include transportability of NS programs (system entry points are now in the transportable section of $VCTR) and changes in two modules, NSPEC and NSABP. NSPEC, which handled parity errors, was moved into PERR (which is partitionable), and the module itself is deleted. NSABP, which handles abort processing, is now partitionable.

DEBUG/1000

xdb Compatibility Mode

xdb provides an interface to Symbolic Debug/1000 which is similar to the xdb debugger for HP-UX. xdb provides a superset of Debug/1000 functionality; all existing Debug/1000 commands are available as well as the xdb commands. Any Debug/1000 command line may be entered by preceding it with a colon. Note that in xdb mode, the term "current location" refers to the location currently listed on the screen rather than the point of suspension of execution.

One useful feature of xdb is record/playback. This feature helps re-create program states and record all debugger output. It is particularly useful for bugs requiring a lengthy set-up.

Note: there is a significant difference between xdb/1000 and xdb for HP-UX in the area of record/playback. xdb/1000 does not implement separate files for "record" and "record-all". Turning on one of these features will close the file associated with the other, if any.

CONCLUSION

The 6.0 release has many new enhancements to RTE and other HP 1000 software. Users will have their choice of which features they'd like to take advantage of when customizing their 6.0 system. We hope you enjoy this much-awaited release!

PAPER NO.: 1006

TITLE: Interrupts and RTE

AUTHOR: David Medlicott
Hewlett-Packard Co
c/o Ella Washington
19091 Pruneridge Ave, MS 46Lg
Cupertino, CA 95014
(408) 447-1053

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

SoftBench Link/1000 Encapsulation

A State of the Art CASE Environment for the HP 1000

Interex 1992
Paper# 1007

This paper discusses Hewlett Packard's new *SoftBench Link/1000 Encapsulation* product which is a new program development tool for the HP 1000 A-series platform. It runs encapsulated within *HP SoftBench*, HP's CASE (Computer Aided Software Engineering) product for HP-UX systems.

This paper outlines the significance of SoftBench as a CASE tool, presents a simple tour from a a user-level perspective, and discusses how *SoftBench Link/1000* integrates the HP 1000 into the HP-UX environment.

Gene Marshall
Hewlett-Packard
11000 Wolfe Road-M/S 42UN
Cupertino, CA 95014
June 3, 1992

HEWLETT
PACKARD

SoftBench Link/1000 Encapsulation

A State of the Art CASE Environment for the HP 1000

Introduction

In the past several years CASE has grown from a concept to an industry. The integration of tools, such as the editor, compiler, and debugger, into a homogeneous, window-oriented environment leads to a more productive and better quality software development environment. With the standard SoftBench product, Hewlett-Packard provides HP-UX developers a software development environment consisting of both an integrated set of program development tools and a tool integration platform. SoftBench is embedded in a window-oriented environment, based on the industry-standard X11 Window System with OSF/Motif appearance and behavior. The standard SoftBench product consists of five tools which cover the construction, test, and maintenance phases of software development:

File Management:

> The **Development Manager** (DM) is the tool used to manage the files on which other tools operate. In particular, it organizes and maintains the program's source files during the development process. Execution of other tools (such as editors and compilers) can also be administered from this manager.

Program Construction:

> The **Program Editor** is a language-sensitive editor for source files, load files, etc.
>
> The **Program Builder** activates the compiler and linker using the HP-UX make utility and the appropriate makefiles. Automatic creation of the make files is also performed by this builder.

Program Analysis: [1]

> The **Program Debugger** tests the execution behavior of a program.
>
> The **Static Analyzer** provides information regarding the structure of a program.

While utilities such as these are not new to the development engineer, the SoftBench integration platform enhances this environment by providing:

> A consistent graphical user interface throughout all applications based on the OSF/Motif user interface,
>
> A pervasive, interactive help system, and,
>
> A Broadcast Message Server (BMS) which allows for a task oriented and partly automated working environment in the edit-compile-link-debug cycle.

SoftBench is not a closed subsystem. The Broadcast Message Server is designed to integrate other software packages within the SoftBench environment. *SoftBench Link/1000* is one such package. By linking the HP 1000 into this environment, developers continue to use tools they are familiar with (such as the vi or emacs editors) while developing for the HP 1000 platform.

[1] While the **Program Debugger** and **Static Analyzer** are standard tools within SoftBench, they are not supported for use on HP 1000 applications or within *SoftBench Link/1000*.

From the user point of view, *SoftBench Link/1000* transparently integrates the HP 1000. The additional utilities which make up *SoftBench Link/1000* provide a core set of tools needed for RTE-A application construction, testing, and maintenance from the HP-UX platform.

Programmers will edit and administrate HP 1000 source code on an HP 9000 HP-UX platform with the standard SoftBench functionality. *SoftBench Link/1000 Encapsulation* insures that architectural dependent tasks (such as compiling, linking, and debugging) are executed on the HP 1000, transparent to the user.

For the first time, the HP 1000 user can utilize standard techniques in software development such as revision control, automatic generation and usage of makefiles, as well as tool communication functions to automate tasks in the edit-compile-link-test cycle. With improved windowing capability and a simple user interface, *SoftBench Link/1000 Encapsulation* results in increased productivity and quality in the software development environment for the HP 1000 A-Series system.

A tour of SoftBench Link/1000 Encapsulation

Before we take a look at exactly how *SoftBench Link/1000* performs this transparent integration, let's take a brief tour of the product to understand what the development process is like:

Once *SoftBench Link/1000* is installed and configured, on the HP 9000, programmers start SoftBench as normal. The only difference they may notice at startup is the presence of the HP 1000 communication monitor: SBL_c_1k. This is the process which performs all the background communication with the HP 1000 system.

The most likely scenario would be to continue work on an existing project in a team environment. This is made up of many files organized in several directories, some of which would be *checked-out* to other developers. In our simple tour, however, let's write a new program.

The central operating window within SoftBench is the **Development Manager**. It's from this point the programmer will determine which directory he or she would like to work in. Editors, building and debugging will all be initiated from this window.

The first concept which must be understood is that of a *base directory*. Since our source code will eventually transfer to the HP 1000 for compiling, the *base directory* we use on the HP-UX system will compare to the HP 1000's root directory ('/'). You might want to think of the base directory as your default home directory which is set when you initially log onto the HP-UX system.

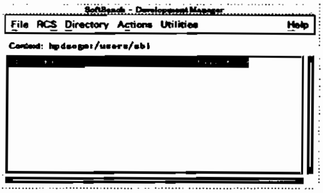

SoftBench - Development Manager
File RCS Directory Actions Utilities Help
Context: hpdaoget:/users/sbl

Figure 1

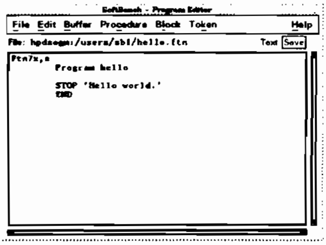
In this example, let's assume when *SoftBench Link/1000* was configured, the HP 1000 user was set to *SBL*, and that account sets an initial working directory of */SBL*, and */SBL* has been created. On the HP-UX side, the SBL user was configured for a base directory of `/users`.

Since we are writing a fairly simple program (one without multiple executables, libraries or include files), let's just place it in our home directory of the HP 1000, which is */SBL*. By looking at the Development Manager window above, we see that we are already positioned at `/users/sbl`. Since our base directory is configured as `/users`, any files we place in `/users/sbl` will be transferred to the HP 1000 under directory */SBL*.

All file names on HP-UX should be lower-case. They are converted to upper case when transmitted to the HP 1000. Likewise, when directory and file names are sent from the HP 1000 to HP-UX, they are down shifted. If your HP-UX directories or files names consist of upper case letters, they will not be match the down-shifted filenames.

We now continue by selecting *File, Edit* from the menu and name our new source file: hello.ftn. An edit window appears and we may now enter our program as it appears in figure 2.

Once entered, we click on *Save*. Back in the Development Manager window, we notice that hello.ftn appears in our file list as a FORTRAN RTE source. We select the *Directory* pull-down and *Build current directory....* The Program Builder window appears and we get an error message that a makefile does not exist. To automatically create one, we select *Makefile* and *Create Program....* In the small pop-up that appears we enter the name of the executable file we desire: hello.run (the standard HP 1000 type-6 file name). Shortly after we click on OK, we see rte.mkf appear in our directory. Our makefile is now complete and all we do is click on *Build*.



File: hpdaegm:/users/sbl/hello.ftn

```
Ftn7x,s
        Program hello

        STOP 'Hello world.'
        END
```

Figure 2

The building progress that follows is monitored within the program builder's window:

- It determines whether any source or include files in this directory need to be uploaded to the HP 1000. Since our hello.ftn file doesn't exist on the HP 1000, it will be copied.

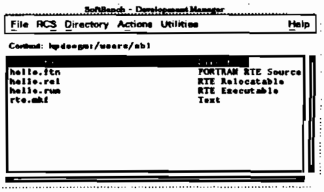
- The HP 1000 FORTRAN compiler is scheduled with output visible in the builder's window. It is here we can see any syntax errors causing compiler errors.

- Assuming an error-free compile, LINK is scheduled, and if successful, the build process completes without an error.

When this has finished, the Development Manager's window will be similar to that shown in Figure 3.

You might wonder why hello.rel and hello.run appear on the HP-UX system; after all, this HP 1000 code is of little use on HP-UX. Well actually these files are only directory entries; they are zero-length files used as placeholders. We can insure that the edit-compile-link process was successful by looking at this list and verifying that the `.run` file is present.

Associated with each of these entries is a time-stamp. If we were to edit hello.ftn, its timestamp would become newer than hello.rel and hello.run, signaling the make process, which is timestamp driven, that a new compile and link is required.



Context: hpdaegm:/users/sbl

```
hello.ftn                          FORTRAN RTE Source
hello.rel                          RTE Relocatable
hello.run                          RTE Executable
rte.mkf                            Text
```

Figure 3

Hello.ftn needs to remain on HP-UX for future edits, and rte.mkf is required for making the software. Rte.mkf is not copied to the HP 1000.

So far we've done everything but run our program. Back in the development manger we double-click on hello.run. Since SoftBench knows this is an executable by the `.run` suffix, it will automatically open a new hpterm window, sign onto the HP 1000 via telnet, and schedule the program. After the program output appears, type *ex* at the CI> prompt and the hpterm window will close.

Once our program runs the way we intended, we can check the source code into source control. SoftBench integrates the standard HP-UX revision control utility, RCS. From the Development Manager, we pull down the *RCS* menu and select *Create initial version*. Additional editing of these source files will require that these modules be *checked out* of source control. This helps maintain file integrity and revision control by allowing only one user to have one version of the software checked out at any one time. (For more information on RCS, see the RCS man page).

As you can see, *SoftBench Link/1000* provides an easy means of creating and maintaining HP 1000 applications from the HP-UX platform. The only knowledge of the HP 1000 we needed in this example was an understanding of file name extensions and an understanding of how to terminate a CI session using the *ex* command. In fact, we didn't have to know what system to log onto or how to do it!

Now let's take a closer look at the individual *SoftBench Link/1000* components.

Components of SoftBench Link/1000 Encapsulation

SBL_c_1k and Tool Communication

As previously discussed, communication among SoftBench and encapsulated tools is accomplished by the BMS. Communication between the HP 9000 and the HP 1000 is accomplished with the *SoftBench Link/1000* tool SBL_c_1k.

Broadcast Message Server (BMS)

The Broadcast Message Server is the heart of SoftBench. On the HP 9000, the SoftBench tools communicate in a networked, heterogeneous environment via a broadcast communication facility designed to support close communication of independent tools. Message requests allow one tool to invoke the functionality of another tool. For example, when we completed the edit of our file, the development manager was triggered by the editor and knew a new file was created in our directory. Notification messages allow tools to define triggers which respond to events and initiate other actions. Triggers are cause/effect relationships. They can be caused by system events, and in turn cause a user-defined action to occur. In this manner, triggers link one or more tools together to support a task or process.

SBL_c_1k

Using the SoftBench platform for RTE-A program development requires that some tasks be performed on the RTE-A system (compiling, linking and debugging). These actions are initiated through appropriate statements in the makefiles executed by the Program Builder. SBL_c_1k is the communication server which accomplishes all *SoftBench Link/1000* communication needs.

At startup, SBL_c_1k determines the HP 1000 to connect to by reading a configuration file (`$HOME/.SBL_config`). User logon information is also stored in this binary file. It then initiates two sessions with the HP 1000:

The first session is a programmatic DSCOPY session associated with the file transfer pipe. This pipe is used whenever files need to be uploaded or downloaded.

The second session is used for remote command execution. This is a standard telnet session and SBL_c_1k waits for a `CI>` prompt to be issued to determine a successful connection. It is important when setting up this SoftBench account on the HP 1000, that the prompt issued is `CI>`. If you are using KCI or a shell other than CI, use a hello command file to set the prompt to exactly `CI>`.

As a result of these two sessions, all functionality can be contained within *SoftBench Link/1000*. That is, no software is resident on the HP 1000! Remote command execution is accomplished via a telnet session, and file transfer via a programmatic DSCOPY session. The connection must be maintained throughout the development process. If this connection is closed, the pipes will be closed and all subsequent communication will fail.

SBL_config - *SoftBench Link/1000* Configuration

The configuration file `$HOME/.SBL_config` is used by SBL_c_lk and contains information necessary for command and data exchange between the HP-UX and RTE-A systems. This information includes:

- HP 1000 nodename, login, and password,
- base directory for the HP-UX system, and
- directory name for *make* template files.

The SBL_config tool provides a windowed environment for setting these values:

HP 1000 node and user information

This information determines which HP 1000 is used as a development machine and how SBL_c_lk should sign-on. All compile, link, and debug processes are executed on this host. Unlike SoftBench which allows us to distribute these tasks among several HP-UX machines, SBL_c_lk requires use of a single HP 1000 for any one build.

The HP-UX Base Directory and the Context Directory

The *base directory* must be set using SBL_config when you set up your development environment. This directory must exist on your HP-UX system and is equivalent to the HP 1000 root directory ("/").

The current *working directory* is then defined by your *context*. If the context directory is *not* set below the HP-UX base directory then subsequent file transfers will fail. This happens when you click on directories from the development manager window. If you back-up your directory tree in HP-UX and transverse down a different path, transfers will fail since you are no longer *below* you base directory. Remembering that the base directory is a configured parameter and your context is a dynamic parameter may help you understand this.

This concept of a *base directory* and *context directory* may be confusing for the first time user, but after using SoftBench a short time you'll learn that this a handy way of pointing to your data. It consists of a host name, directory name, and optionally a file name.

When the context is correctly set below the HP-UX base directory, *SoftBench Link/1000* interprets the context by parsing off the HP-UX base directory and setting the corresponding RTE-A global directory to the next directory in the path. In our example, the HP-UX base directory is set to `/users` and the context directory was set to `/users/sbl`. This meant that when transferring files, all source, include and load files from `/users/sbl` on the HP-UX were transferred to `/sbl/` on the HP 1000.

Make file templates

The Program Builder in SoftBench accomplishes its task through use of the HP-UX *make* utility. This *make* utility requires a makefile which is normally named *Makefile*, and can be automatically created within the builder by using another utility: *mkmf*. *Mkmf* requires templates to start with. Standard SoftBench keeps these templates in `/usr/lib/mf/`. There are two template files, `C.p` and `C.l`, which are used when creating makefiles for programs or libraries respectively. When *SoftBench Link/1000* is installed, new `C.p` and `C.l` files are installed in `/usr/SB_Link/config/templates/`. These templates configure a makefile to use SBL_ftn7x, SBL_lindx and SBL_link, rather than their HP-UX counterparts.

Under normal conditions, all *SoftBench Link/1000* users would use the templates found in /usr/SB_Link/config/templates/. However, these templates need to be edited if non-standard file extents are to be used. *SoftBench Link/1000* knows files ending in .finc are FORTRAN include files. If you preferred to use .FTNI, your makefiles need to know this. By copying the default templates from /usr/SB_Link/config/templates/ to, say, $HOME/templates/, you can edit them for your own use. You also need to edit the system-wide configuration file /usr/SB_Link/config/type.ex. This file contains a list of all file extents known to the SoftBench subsystem. When using your own templates file, don't forget to use SBL_config to specify your new templates directory.

SBL_update - File Transfer and Consistency Check

SBL_update provides us three functions:

- Creates directories on the HP 1000, including global directories,
- Provides a means of moving an existing software development environment from the HP 1000 to our HP-UX system, and,
- Provides file consistency checking.

During the normal build process, any directories which you create on the HP-UX system below your base directory are automatically created on the HP 1000, except at the global level. To create a global you either need to use SBL_update or create it manually from an RTE window.

If you have an existing software project on the RTE-A system and want to maintain it under *SoftBench Link/1000*, SBL_update creates the necessary directory hierarchy on the HP-UX system and copies all source and load files to their related HP-UX directories. SBL_update will copy all files and subdirectories under the specified global RTE-A directory to the HP-UX system (specified by setting your context and base directories).

SBL_update can also perform source file consistency checking. Since we work on two file systems (HP-UX and RTE-A), it is likely that files on your HP-UX system become newer than that on the HP 1000. This might happen when you continue to edit files on your HP-UX system but don't want to build them right away. A consistency check could be performed on a nightly basis to transfer these files to the HP 1000.

You should be careful not to edit files on the HP 1000. If this were to happen, the changes made on the HP 1000 file will be lost at the next build. Therefore, you should do *all* development on your HP-UX system.

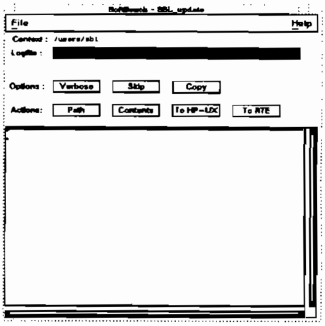
To run the SBL_update program, begin in the Development Manager window and select the *Utilities* pull-down and then *SBL_update*. A window appears as shown in figure 4.

If you toggle *verbose*, SBL_update displays each transfer command as it is being performed. You can watch each directory get created and each file being transferred.

During a transfer, we normally see files which qualify for transfer, not those which are skipped. Selecting *skip* informs SBL_update to show files that do *not* qualify for transfer. These would be files that don't need to reside on HP-UX (such as `.rel` and `.run`) or have an extent which is not configured into *SoftBench Link/1000* (such as `.txt` or files with no extents).

Setting *copy* transfers files (in addition to the directory tree) to the HP 1000. If you do not select this option, SBL_update creates only directories on the HP 1000 system. Unless you have added other file type extensions in the type.ext file (discussed shortly), only source files are transferred (.c, .ftn, .finc, .mrg, .lod, .cmd, .mac).

Path checks whether all path names on one system have a corresponding path name on the other system.



Context : /users/sbl
Logfile :

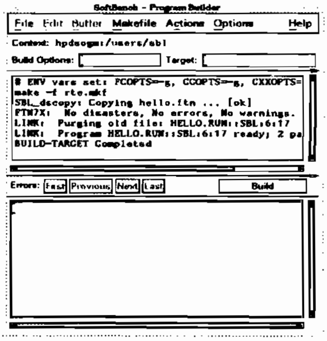
Figure 4

Contents checks both path names and the contents of the files. This is done by copying each file from the RTE system to the HP-UX system, then comparing the contents by doing a binary compare using the *cmp* utility.

To HP-UX starts data transfer from the HP 1000 to HP-UX.
To RTE starts data transfer from HP-UX to the HP 1000.

Building RTE-A Software

Once our program is at the point where it is ready to be compiled, we now use the SoftBench build utility (BUILD). The building process takes source code, include files, and any other application specific information and compiles and links them to create executables. BUILD performs this by using the HP-UX *make* utility, which in turn gets instructions from a *makefile*. *Make* will look for a *makefile* in the current directory. If no makefile exists, build will initially display an error, but we can automatically create a makefile by selecting *Makefile*, *Create Program* from the pull-down menus. An additional window will pop up prompting you to enter further information, such as the desired executable name, compiler options, load flags, library names, etc. Once this information is entered the makefile will be generated and the builder can be run to build the program or library. Figure 5 shows the builder after a successful file transfer, compile, and load.

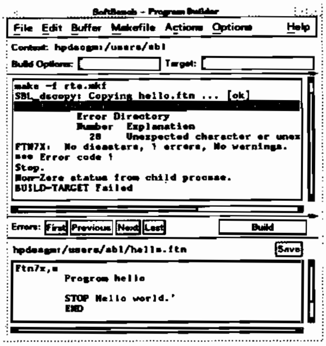


Context: hpdsogm:/users/sbl

```
# ENV vars set: FCOPTS=-g, CCOPTS=-g, CXXOPTS=
make -f rte.mkf
SBL_dscopy: Copying hello.ftn ... [ok]
FTN7X:  No disasters, No errors, No warnings.
LINK:  Purging old file: HELLO.RUN::SBL:6:17
LINK:  Program HELLO.RUN::SBL:6:17 ready; 2 pa
BUILD-TARGET Completed
```

Figure 5

If the program encountered errors during the build, the lower half of the BUILD window will display all error messages. By simply clicking on the error message, the file containing that error will be opened by the SoftBench editor and the cursor will be positioned on the line generating the error. This is shown in Figure 6. By removing an apostrophe from our previous `hello.ftn` example, we generated a compile error at line 4. Clicking on the error generated by the FORTRAN compiler, the builder placed us in an edit window and the text cursor is positioned at the offending line. After fixing our mistake we simply press *Save* and *Build*, all from within the build window.


SoftBench - Program Builder
File Edit Buffer Makefile Actions Options Help
Errors: First Previous Next Last
Build
Save

Context: hpdaogm:/users/abl

Build Options: Target:

```
make -f rte.mkf
SBL_dscopy: Copying hello.ftn ... [ok]

            Error Directory
            Number  Explanation
              28    Unexpected character or unex
FTN7X:  No disasters, 1 errors, No warnings.
*** Error code 1
Stop.
Non-Zero status from child process.
BUILD-TARGET Failed
```

hpdaogm:/users/abl/hello.ftn

```
Ftn7x,s
        Program hello

        STOP Hello world.'
        END
```

Figure 6

The building process is efficient: only the source files that have been modified are uploaded and recompiled. *SoftBench Link/1000* enhances the SoftBench makefile by providing an RTE-A makefile template that recognizes RTE-A specific suffixes, compiler options, etc, therefore allowing for automatic makefile generation for HP 1000 RTE-A applications. The resulting makefile, `rte.mkf`, contains all commands to either create a program or a library for an HP 1000 system. All architectural dependent actions (such as, recognizing that a FORTRAN program must be compiled on the HP 1000) are integrated into the makefile and are performed transparently to the user.

After a compile, the resulting relocatable remains physically on the HP 1000 RTE-A system. A zero-length file is made on the HP-UX system to retain the timestamp information required for future makes. If we are creating a program, LINK is invoked and a dummy executable is created on HP-UX. This dummy executable can be invoked, which then triggers the actual executable on the HP 1000 to either be run or debugged remotely, depending on how it was invoked.

When creating a library, the compiled modules are merged into a library which is then indexed. The resulting relocatables and library are also kept on the RTE-A system whereas on HP-UX dummy relocatable and libraries are created to satisfy the HP-UX *make* mechanism.

Debugging RTE-A Software Using SoftBench Link/1000

A successful build doesn't always equate to functional code. Logical errors and runtime errors often occur in a program development environment. *SoftBench Link/1000* provides window driven functionality to remotely run Debug/1000 via telnet. By highlighting our executable in the Development Manager window, hello.run, we pull down the *Actions* menu and select one of the two DEBUG options: *debug1* or *debug2.*

Debug1 opens an hpterm window, telnets to the HP 1000, signs us on, and starts Debug/1000 on our executable. This is similar to double clicking on our executable, except that Debug/1000 is in control of the execution.

Debug2 is similar to debug1, except two hpterm windows are established to the HP 1000. One is used to interact with Debug/1000 while the other is used for the application's output. This is very handy when applications force the terminal into states where it becomes hard for Debug/1000 to operate; specifically block-mode or graphic applications. As we will discuss later, we can change these hpterm windows into GFoX windows, which will support Forms/1000 and Graphics/1000.

(For more information on how Debug/1000 works, see the Symbolic Debug/1000 Reference Manual, part number 92860-90001.)

Standard SoftBench Tools

Development Manager (DM)

The Development Manager is the tool responsible for all file management actions as well as for the activation of other SoftBench tools. The Development Manger has been enhanced by *SoftBench Link/1000* to include a *Utilities* menu that contains all HP 1000 specific RTE-A programming tools, as shown in Figure 7.

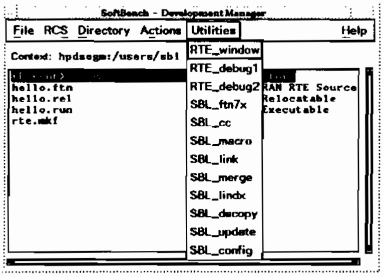


Figure 7

`RTE_window` Opens a telnet window to the configured HP 1000 and starts a CI session.

`RTE_debug1` Opens a telnet window and starts the debugger on the selected executable.

`RTE_debug2` Similar to RTE_debug1, except two windows are opened, one for debugging, the other for application output.

`SBL_ftn7x`
`SBL_cc`
`SBL_marco` These functions allow you run an HP 1000 compiler directly, instead of going through the make process. A pop-up dialog panel allows you to specify the source file.

`SBL_link`
`SBL_merge`
`SBL_lindx` As with the compilers, these functions provide access to the corresponding HP 1000 utilities through a dialog window.

`SBL_update` The utility for creating directory paths on either HP-UX or the HP 1000, transferring source files in either direction, and providing consistency checks.

`SBL_config` Configures the *SoftBench Link/1000 Encapsulation* environment by allowing you to set the HP 1000 host name, logon, and the base and template directories.

In addition to the *Utilities* pull-down, the Development Manager automatically configures the *Actions* menu to list only valid actions on the selected RTE-A file, based upon type extension. For example, `hello.ftn` can only be edited and compiled, whereas `hello.run` can only be executed or debugged.

One of the most important tasks of the Development Manager is the handling of the revision control system. Revision control provides features like multiple revisions, audit trail, access control, efficient storage, and flexible retrieval. RTE-A files, in addition to HP-UX files, can be checked in and checked out for operation by other SoftBench tools. This is an important feature to the RTE-A developer since there is no Revision control on the HP 1000.

Program Editor

The Program Editor is an easy-to-learn, programming language sensitive, mouse/menu based standard SoftBench file editor. It automatically synchronizes file views. If a file is modified by a tool in one window, the file is updated in the other windows where it is also being viewed. It will automatically adjust for programming language specific indentation requirements. The editor is highly customizable (for example, different keyboard accelerators can be specified).

Static Analyzer

The Static Analyzer provides information regarding the structure of a program. It provides cross-reference queries such as: *where declared*, *where defined*, *where used*, or *where modified*. This tool is particularly valuable while maintaining code or porting code. However, the generation of static information is a function of the HP-UX compiler. Therefore, only HP 1000 source code which can also be compiled on an HP-UX system can be used for static analysis.

SoftBench Link/1000 Encapsulation Configurability

In addition to the flexibility and functionality that SoftBench already provides, *SoftBench Link/1000* specific functionality is configurable as well. *SoftBench Link/1000* has been designed to allow every individual programmer a unique environment if desired.

Upon startup and throughout operation, SoftBench uses the following configuration files:

$HOME/.toolset

This file lists each tool to be started up automatically with SoftBench:

```
DM        hpdsogm  /users/sbl * * hpdsogm
SBL_c_1k  hpdsogm  /users/sbl * * hpdsogm
```

The first parameter is the tool name. Here we are starting the Development Manger and the SBL_c_1k utility. The third parameter is the context each of these tools should be initially set to. The second and last parameter are set to the HP-UX nodename. If you would like additional tools started, you can simply edit this file and include them the same way.

$HOME/.softinit

This configuration file tells SoftBench which tools to spawn as the user requests them. It is in this file we can:

- Substitute GFoX for an hpterm window for *filerun*, *debug1*, or *debug2*, or *RTE Window*.
- Substitute vi or emacs for the SoftBench editor.

/usr/SB_Link/config/type.ex

This file lists all the file type extensions which SoftBench recognizes. Any additional file extent you would like SoftBench to recognize as an RTE-A file, must be included in this system-wide configuration file.

/usr/SB_Link/config/templates/C.p
/usr/SB_Link/config/templates/C.l

These two files are used as a template to the *mkmf* utility which creates our makefiles. Unless specifically configured to be elsewhere, these files are read each time a user has the builder create a makefile for a program (C.p) or a library (C.l).

If unique file type extensions were to be used by a single individual, this person would most likely:

- ► Copy these default templates to their $HOME directory or some directory below that point,
- ► Edit these templates to reflect what the new extents mean (*ex:* `.data` should be treated as an include file).
- ► Use SBL_config and set the templates directory parameter to point to where these files are stored.
- ► Proceed to create makefiles.

Additional information on modifying template files is outlined in the manual.

In addition to the normal usage of *SoftBench Link/1000* as demonstrated to this point, each utility can also be run from an HP-UX system prompt. Each utility will report its runstring parameters if scheduled with an invalid runstring.

By running tools in this way, alternate configuration files can be specified, allowing a user to access another development environment on the same HP 1000 (not under the local base directory) without reconfiguring.

More information on individual tool usage is discussed in the *SoftBench Link/1000 Encapsulation Reference Manual*.

Summary

This paper demonstrated some of the unique benefits offered to the HP 1000 users who leverage HP-UX and X11 front ends into their programming or support environment. Let's summarize some of them:

- ► Improved quality control of RTE-A software development and analysis.
- ► Productivity advantage thanks to a rapid interactive program development environment.
- ► Automatic generation and use of makefiles.
- ► Protection against concurrent file changes.
- ► Automatic error browsing.
- ► Source code administration under revision control resulting in safe development and modification of source code and the possibility of restoring previous software revisions.
- ► Transparent use of RTE-A tools (compilers, linkers, debugger, etc.)
- ► Ability to debug blockmode or Graphics/1000 applications over the network (in conjunction with GFoX).
- ► The multi-window graphical user interface and integrated on-line help contribute to ease of use and fast learning.
- ► SoftBench provides a highly configurable development environment.

Acknowledgements

I would like to thank the following people for their support and help on this paper:
Wolfgang Oskierski, Gary Thomsen, Hilary Feier, and Kristin Anderson.

What's In A Nameserver?
Domain Name Service for the HP 1000

Todd Poynor

Hewlett-Packard
Software Technology Division
11000 Wolfe Road, Cupertino, CA

Among the new features which the 6.0 release of RTE-A/VC+ in conjunction with NS/ARPA may boast is client support for the Internet Domain Name System (DNS). This paper aims to introduce the DNS at a high level, as well as discuss a few details of the HP 1000 implementation.

Introducing the Domain Name System

The Domain Name System, or DNS, is essentially a distributed database of network information. It is estimated that over a million computers implement DNS, making it the world's most distributed database. The DNS is an *Internet* standard, meaning that its definition and use has been standardized by the Internet Activities Board (IAB). The IAB is an organization beneath the Defense Advanced Research Projects Agency (DARPA), which originally funded the development of the *TCP/IP Internet Protocol Suite*, of which NS-ARPA/1000 is a partial implementation. The IAB has also mandated that any full implementation of the Internet Protocol Suite use the DNS for certain purposes. Any computer system which is to be smoothly integrated into a TCP/IP network nowadays needs to support the DNS client function; by necessity, the HP 1000 is joining the rest of the industry in supporting this service. The remainder of this section is devoted to introducing DNS basics for readers not previously familiar with domain names.

Who Can Remember "15.0.88.167"?

To keep things simple, we will begin the discussion by concentrating on the major function of the DNS: to map human-readable names of computers to machine-readable network addresses. This process is familiar to RTE users who have encountered the /system/nodenames file used by the "transparent file access" feature of FMP and DS/NS, or who have maintained an NS/ARPA Nodal Registry input file. Users prefer to deal with symbolic names for computers, which are far easier to remember and which can communicate much more information about the computer (such as location or purpose) than the numeric addresses used by networking software and hardware. As an aid to understanding the DNS model of name-to-address translations, let's take a look at the system previously in place on the Internet, which the DNS replaced.

The HOSTS.TXT Era

In days gone by, the host name-to-address translation database for the entire Internet was contained in a single text file named HOSTS.TXT, maintained by the Internet Network Information Center (NIC). When an organization on the Internet wished to add a hostname to the tables, it had to wait for the NIC to approve the new name and update the official hostname database before the rest of the Internet could learn of the name. Each host which required an up-to-date copy of the database would periodically obtain a new copy of the entire file over the network, typically from the NIC itself. Both the amount of network bandwidth consumed and the load generated on NIC machines were extraordinary. By mid-1986, the database contained 3100 "official" names and 6500 "alias" names, creating quite a maintenance nightmare for the NIC. The problem was compounded by the high frequency in which name-to-address mappings tend to change. To make matters worse, the number of connected hosts within organizations already on the network was increasing rapidly as large mainframes tended to be replaced by numerous personal computers and workstations.

Another problem was presented by the host naming scheme in effect at the time, which simply represented a flat namespace. Each name was a unique identifier, without any provision for structuring the namespace in some logical fashion. Due to constraints imposed by software subscribing to antiquated host naming conventions, many of the names were eight characters in length. As you might expect, the possibility for conflict between any requested new name and an existing name was rather high.

The Domain Name Tree

The DNS solves the problems presented by the HOSTS.TXT scheme by imposing an organization on the host namespace and distributing authority over the namespace among the organizations which comprise the Internet. First, let's examine the structure to which host names must conform.

The DNS implements a "tree" of names, arranged in a hierarchy corresponding to the organizations which make up the Internet. This tree represents the entire host namespace of the Internet. The hierarchical structure allows each member of the Internet to assume responsibility for its own branch of the tree, as we shall see. The structure also allows each member to define its own "sub-tree" of names beneath its branch of the domain name tree, without need for a central authority to administer the definitions.

The tree thereby defines more than just host names. Since we are presently restricting our discussion of the DNS to the function of mapping host names to network addresses, we can think of the tree as containing two kinds of names. The "leaf" or "terminal" nodes of the tree correspond to actual host names within an Internet organization. The "non-terminal" nodes correspond to levels of the conceptual hierarchy just mentioned, such as an organization name. If this sounds a bit murky to you, then a closer look at the actual tree may help to clarify matters.

The Internet authorities have defined the top level of the tree, beneath which the organizations that comprise the Internet define their own name subtrees. Table 1 shows the most common top-level domain names. In addition to these top-level domains, a number of two-character names have been designated for the countries in which Internet members

Domain Name	Sub-Tree Represented
COM	Commercial organizations
EDU	Educational institutions
GOV	Government agencies (typically U.S.)
MIL	Military agencies (typically U.S.)
ORG	Organizations otherwise categorized

Table 1: Common top-level Internet domains.

reside. This creates two structures of the Internet namespace tree: one structured by organizational type, and one structured by geographic location. Each organization joining the tree may choose which structure to join. The presence of these two overlapping hierarchies continues to be the subject of some controversy within the Internet community. For the sake of simplicity, we will focus on the namespace hierarchy structured by organizational type in the remainder of the discussion.

The Internet authorities also control the registering of second-level names within the tree. An organization which desires to join the namespace will apply to the NIC for a second-level domain name, specifying which top-level domain the new name should be registered. For instance, Hewlett-Packard has registered second-level domain name "HP" beneath top-level domain "COM". The namespace truly is a tree, and so many other commercial organizations have registered names beneath the "COM" top-level domain.

There ends the responsibility of the NIC in registering domain names: each organization is responsible for maintaining its own subtree of the namespace. This keeps the administrative load on the NIC to a manageable level. So long as the organization follows certain rules set forth by the DNS standard, it is free to invent its own conceptual hierarchy of domain names below its subtree.

For example, HP, which reportedly owns the largest subtree of the "COM" top-level domain, has chosen to divide its sizeable namespace by the names of the physical sites which house the computers to be described. As an example of HP's scheme, the computer being used to write this paper is registered within a sub-domain named "CUP", since this author works at the HP Cupertino site. Other sub-domains of HP's namespace include "SVALE" for the Sunnyvale, CA. site, and "SWEDEN" for HP Sweden. Each of these sub-domains are structured below HP's officially-registered sub-domain "HP", which is in turn structured below top-level domain "COM".

At the lowest level of the tree are the actual host names for which the network addresses are to be defined. For example, the HP 1000 which this author most commonly works on is named "MAGIC", and is registered beneath the "CUP" sub-domain mentioned above. The DNS imposes no requirement that host names be at a certain level in the tree. An organization may define more than one level of sub-domains within its portion of the tree, or it may define none at all.

The Internet authorities force a certain structure on the top two levels of the domain hierarchy, since these are assigned according to the names and types of organizations which make up the Internet. The member organizations have free reign to arrange their own sub-domains as they see fit. The DNS does not require the chosen naming conventions to correspond to the physical layout of the underlying networks. That is, the naming hierarchy is entirely conceptual, and not necessarily related to physical network topologies in any way.

World's Most Decentralized Database

Now that we've laid out the hierarchical structure of the namespace implemented by the DNS, let's return to the subject of the decentralized database in which the namespace is stored. Each of the names at "non-terminal" levels in the tree described above provide the hierarchical organization of the namespace which allows the authority for the tree to be distributed among its participants. As already mentioned, the Internet authorities no longer have to concern themselves with accounting for each change in the name-to-address mappings of the Internet as a whole. Instead, only the top two levels of the hierarchy are controlled by the central authority; it is up to the individual organizations to manage their own portions of the namespace. This solves one of the main problems with the obsolete HOSTS.TXT scheme mentioned previously.

The Internet namespace is becoming quite large. This is an understatement. In 1990, the namespace included more than 137,000 host names. A single host would be ill-advised to try to maintain an up-to-date copy of the entire namespace. Rather, the database remains spread across the organizations which comprise the Internet. The DNS provides a scheme for hosts requiring name-to-address mapping information to locate a database which contains the desired information from among the multitude of databases on the network. Also defined is a means for the host in need of the name-to-address mapping to obtain just the desired information over the network. In addition, the client may cache the information locally for some time, such that repeated queries for the same information need not consume network bandwidth. We'll look at the process by which the DNS transfers information more closely in short order. For now, suffice to say that two major problems with the HOSTS.TXT implementation are hereby conquered. The first is that the complete namespace need not be continually downloaded to the entire Internet. The second is that the burden for the transfers performed no longer falls on a single database host.

Domain Name Syntax

Now to return the original purpose of the DNS: to allow humans to enter reasonably friendly names for computers instead of numeric network addresses. The syntax in which these domain names are entered specifies each name, called a "label" in DNS terminology, in order of decreasing "locality", each separated by a dot. This means the local host name is entered first, followed by the next-higher name in the hierarchy, and so on, up to the top-level domain. For example, the HP 1000 mentioned in an earlier section may be represented as:

magic.cup.hp.com

As you may recall, the host name is "magic", which resides within HP-Cupertino's subdomain "cup.hp.com", which resides within HP's organizational domain "hp.com", which is registered in the commercial portion of the Internet, top-level domain "com".

Notice that the examples have mixed upper and lower case representations of the same names, treating the case as insignificant. This is in accordance with the DNS standard: case is never significant in domain names.

The DNS imposes a few rules on the syntax of domain names. A full domain name must be 255 characters or less in length, and each label within the name must be no longer than 63 characters. The set of characters which are legal in a label is comprised of the alphabetics A-Z, the digits 0-9, and the hyphen ("-"). A label cannot start or end with a hyphen.

A General Network Information Database

Under DNS domain name syntax, it is impossible to tell the difference between host names and organizational sub-domains based on the name alone. This is because a host name is simply a particular case of a domain name, one which has a network address associated with it. Up to now, we have treated the DNS purely as a name-to-address mapping database. This simplified the discussion somewhat, particularly since it is the function for which the DNS was primarily designed. From this point forward, however, we will speak of the DNS in more general terms, for it is not restricted to this one function.

As was stated at the outset, the DNS implements a distributed database of network information. It defines a hierarchical namespace, where each name may have associated entries in the database which define some attributes for the name. We have seen that one of the attributes which may be defined is a network address; these are associated with names that correspond to host names. Each of these attribute entries is termed a *resource record* (RR) in DNS parlance. A full list of the types of RRs which may be defined is beyond the scope of this paper. However, a partial list appears in Table 2. This paper will also not attempt to detail the precise format in which the attributes are represented. Obviously, some of these RR types require more explanation than is offered here; this list serves only to illustrate that more than just network addresses may be stored in the DNS.

Any name in the database may have more than one associated resource record. For instance, a host may have more than one IP address, or may have both an IP address and a mail exchange host defined.

A name which serves as an organizational sub-domain will also have RRs associated with it, as if it were what we've considered to be a "leaf" node up until now. For example, the subdomain "cup.hp.com" has RRs associated with it, such as NS RRs which tell which hosts run "authoritative" name servers for the domain (this terminology will be explained in a moment), and MX RRs which tell where to send mail addressed in format "`user@cup.hp.com`".

In general, a client seeking information on a domain name from the database will specify

RR Type	Attribute Defined
A	Network IP address
CNAME	Canonical (official) domain name for an alias name
MX	Host name to which mail for the domain should be sent
NS	Host name which runs an authoritative server for the domain
TXT	Any string you like

Table 2: Some resource record types.

both the name and the type of information of interest, and the DNS will return all the information of the requested type known for the requested name.

The DNS Client-Server Model

We have spoken of DNS clients which request information from the DNS database, without going into any appreciable detail of how the request is made and to whom or what the request is directed. The DNS subscribes to the usual TCP/IP "client-server" paradigm, whereby a client which requires information from the DNS contacts a DNS database server somewhere on the network. We won't go into the particulars of the communication in this paper. Suffice to say, the question posed by the client and the answer returned by the server are sent over the network using protocols of the Internet protocol suite, either the User Datagram Protocol (UDP) or the Transmission Control Protocol (TCP).

Name Resolvers

Clients are called *resolvers* in the DNS lexicon. Name resolvers provide the means for applications to query the DNS for information and obtain an answer from a server somewhere. Typically, this is a set of library routines which may be loaded with an application which performs the appropriate calls to the routines.

In general, the task of a resolver is to format a legal query according to the DNS protocol and contact a name server which can answer the question. In most implementations, the resolver knows of a small handful of local hosts which run name servers and can be counted on to process a query through to completion, contacting other name servers to obtain the answer, if necessary. [1]

The resolver also handles the automatic qualification of abbreviated domain names. The DNS itself makes no provision for abbreviated names: it accepts only full domain names from resolvers for processing. But people normally prefer to avoid entering full domain names each time a local host name must be typed in. Accordingly, the resolver will accept abbreviated domain names from users and automatically tack on the upper portion of the

[1] The standard allows for resolvers to do more of the work themselves, but most implementations behave as described.

domain which has been omitted before passing the name on to the DNS. For example, this author can simply enter "`telnet magic`" from a local host, and the local resolver will discern that the full domain desired is "magic.cup.hp.com".

Name Servers

On the other side of DNS query transactions are name servers. These are commonly processes executing within a host's operating system environment (as opposed to resolvers, which tend to be a set of library routines). Name servers can perform two major functions: to hold "authoritative" data about some portion of the domain tree (which we'll describe in just a moment), and to answer queries for information generated by both resolvers and other name servers.

Question Authority

These name servers implement the DNS distributed database, storing the data which is made available to resolvers across the Internet. As mentioned previously, authority for an organization's portion of the domain namespace tree resides within the organization itself. Each organization implements this by running name servers which hold the data relevant to that organization. The data stored by a name server for which the server is the Internet authority is termed the *authoritative data* for the sub-domain. Whenever the information for the organization changes, the authoritative data is updated to reflect the change, and future queries to the name servers which are authoritative for the sub-domain will return the new information.

So one of the tasks for a name server attempting to answer a query which asks for information which does not fall within that server's authoritative data is to figure out where a name server is which does contain the data. As we saw in Table 2, there exists a resource record type within the DNS database information, the "NS" RR, which supplies the names of hosts which run authoritative name servers for a given domain. These RRs appear in the next-higher level in the domain hierarchy. So, for example, the NS RRs which describe the name server hosts for the "hp.com" domain appear in the authoritative data of the "com" top-level domain. The servers for the top-level domains are termed the *root servers*. The names of the hosts which run these are, by necessity, controlled by the Internet authorities and advertised to the entire Internet by a means other than the DNS.

A name server which has received a query for a domain which it knows nothing about can therefore start at the top of the domain tree and navigate its way down the NS RRs for the various labels in the domain. This continues until the domain in question is found. Let's say the local name server receives a query for A ("address") RRs belonging to domain name "steno-pool.big-swifty.com". The local name server might first query a root name server for NS RRs for sub-domain "big-swifty.com". Assuming that at least one NS RR is found, the local name server would then query the name server specified therein for any A RRs pertaining to domain "steno-pool.big-swifty.com".

This hierarchical structure of name servers may be replicated within organizations, such as HP, which define sub-domains within their own sub-domain. For example, an authoritative name server for domain "hp.com" may contain NS resource records which name the hosts running authoritative name servers for domain "cup.hp.com". These name servers would contain the authoritative data for that sub-domain of HP's namespace.

To insure that authoritative data is always available for a sub-domain, the DNS standard stipulates that "redundant" authoritative name servers must exist for each set of authoritative data, that is, more than one name server must exist which is authoritative for the data. These redundant name servers must execute on separate hosts, and must avoid common points of failure, even to the extent that they not depend on the same electrical power source.

All You Need Is Cache

As we have shown, name servers will generate their own queries to other name servers in the process of tracking down information requested by a resolver. If the server were to generate these queries each time the same information was requested, then a large amount of network bandwidth would be consumed, and each non-local name lookup would take a considerable amount of time.

To improve the efficiency of DNS queries for information outside the local "zone of authority", information which is obtained from remote name servers will be cached locally for a period of time. If the server receives a subsequent request for the same information, it will return the cached information together with an indication that the data has been cached, and therefore may be out-of-date. Such cached information is termed *non-authoritative*. An application which requires absolute accuracy of the DNS information it requests may insist on the use of only authoritative data, but most will accept cached data without reservation.

But information should not be cached forever, as it is likely to become out-of-date in the face of continually changing network topologies. Accordingly, each resource record contains an expiration date. The expiration date is actually called a *Time To Live* (TTL), and specifies the maximum amount of time allowed for the RR to remain valid when cached. The TTL is set by the domain administrator in charge of the authoritative data from which the RR is cached. The administrator may adjust the TTL values of various RRs, according to the likelihood that the information will change soon. When a name server finds that it has a cached RR which would satisfy a query, but its TTL has expired, it must discard the RR and generate a new query to the authoritative name server.

So What *Is* In A Nameserver?

As we've already seen, a nameserver[2] may contain authoritative data if the server is authoritative for one or more domains. The server may also contain cached data from outside its authority which has been provided by other servers.

When a name server receives a query, it first checks to see if the domain of the query names a domain for which the server is authoritative. Answering a query with cached (that is, non-authoritative) data for a domain which is under the authority of the server is a definite

[2] Both "name server" and "nameserver" are accepted terms for the same thing.

"no-no." If the domain is outside the authority of the server, it checks to see if any RRs have been cached which match the domain name and RR type requested. If so, it returns the cached data.[3] Otherwise, the server locates an authoritative name server for the desired information, queries that server, and caches the information returned.

You now know the basics of the Domain Name Service. Among the details which we've glossed over or ignored are the mechanics of the network transfers, the formats of the resource records and of the DNS queries and replies, some particulars of how domain authority is delegated between servers, and various nuances of resolver and server processing. See the references list at the end of this paper for sources of more information if you're interested.

The RTE Implementation

Up at the top of this paper, we stated that RTE-A will supply *client* support for the DNS at 6.0. Now that we're hip to the DNS lingo, we know that name *resolvers* are what is supplied. More specifically, RTE implements what may be termed "caching stub resolvers".

A "stub" resolver is one which relies on *recursive* name servers somewhere on the network to do much of the work in answering queries. Name servers as described in the preceding overview of the DNS are recursive servers. Basically, the term means the server will query other servers for information rather than requiring the resolver to perform this function. Most nameserver implementations are of recursive servers, and most resolver implementations are of "stub" resolvers.

A "caching" resolver is one which caches the data returned by servers. Earlier, we ascribed this function only to servers, but resolvers may also perform caching in an identical manner. This is to use the DNS standard terminology, however; in practice, the distinction between resolvers and servers begins to blur at this point.

Slave Mode Servers

The RTE implementation may be thought of as actually supplying stub resolvers and name servers which operate only in what some DNS implementations call "slave mode". This means that the server will simply forward queries which cannot be satisfied from its own cache on to full-service name servers elsewhere on the network. The server will cache the data returned, such that a subsequent request for the same information will not require another network access to a full-service name server, at least until the Time To Live of the data expires.

The RTE name servers cannot hold authoritative data. There is no facility for reading configuration files which define authoritative data. The "zone transfer" feature of the DNS is not currently supported. Zone transfers allow servers to obtain an entire set of authoritative data from another server over the network. Servers with this capability can be designated as "secondary masters" for a domain: alternate authoritative servers which are guaranteed to have a complete set of almost up-to-the-hour authoritative data for the domain.

[3] This discussion oversimplifies the processing of "wild-card" queries, and ignores the possibility of caching negative responses which indicate that information does not exist. But, hey, cut me some slack.

The only real function of the RTE servers is to cache data obtained over the network, thereby speeding up repeated access to the same information and reducing the load on the network. This is particularly important for the RTE implementation, due to its unusual network requirements, which will be discussed presently.

The Resolver Library

The RTE resolver is implemented as a set of FORTRAN-callable routines, gathered into a library which an application may search at Link time. The application must be a CDS program, and the NS/ARPA libraries must be available to supply the underlying networking routines used by the resolver. Versions of the resolver routines which have a calling sequence more convenient for applications written in C may be supplied later if the demand exists.

The RTE library provides versions of each resolver routine that HP-UX supplies at 8.0, albeit with calling sequences modified to ease their use in FORTRAN programs.

The resolver routines use NS NetIPC calls to contact name servers and transfer queries and replies. The RTE slave server is not required to be present for the resolver to function. The resolver may talk over the network to a name server executing elsewhere on a host reachable by NS/ARPA; however, the feature of local caching of data would be lost. The resolver routines execute in exactly the same fashion, regardless of whether configured to query remote servers or to query the local server. Normally, the resolver is configured to connect to the slave server on the local host, which will oversee the processing of the resolver's queries and cache the answers for future reference.

Arbitrary Domain Names

One question which might arise is, *"How can DNS for the HP 1000 deal with all these domain names which are not legal NS/ARPA node names?"* For example, consider a domain name containing more than three labels, such as "`magic.cup.hp.com`". This name cannot be entered into the NS Nodal Registry (as in an NRINIT configuration file), and cannot be entered as an argument to a program which will use NetIPC routines IPCDest or IPCLookup to resolve the name to a network address.

Such restrictions remain. Fortunately, the NetIPC routines just mentioned accept IP addresses in dotted decimal form, such as "`15.0.88.167`". This capability is all the DNS needs for its communication purposes. Indeed, this is the main purpose of the DNS: to translate the domain names into IP addresses for networking purposes. Accordingly, domain names are never passed to NetIPC routines or any other name-handling component of NS/ARPA, only the resolved IP addresses are.

Under this model, it is possible to code DNS-savvy "wrappers" around existing utilities, such as TELNET and FTP. The new program can use the DNS resolver to lookup the IP address of arbitrary domain names entered in place of the usual NS nodename command-line arguments. The "wrapped" utility is then executed, but with an IP address in dotted decimal format appearing in the runstring instead of the domain name.

Network Protocol Issues

As mentioned previously, DNS/1000 performs some unusual networking processing. In particular, the DNS standard mandates that resolvers and servers support the User Datagram Protocol (UDP) for queries, as well as supporting the Transmission Control Protocol (TCP). UDP is the protocol by which most queries should initially be made, except for zone transfers. TCP is normally used only if the amount of information to be transferred is too large for the UDP protocol to handle, for reasons which need not concern us here. Typically, a query is generated first over UDP, and the server sets a flag if the answer had to be truncated due to protocol limitations. At that point, the query is repeated, this time using the TCP protocol, which can handle arbitrarily large volumes of data. The reason UDP is first attempted is for sake of efficiency. This protocol places a lighter load on the network than does TCP, which involves a great deal of overhead.

The problem is that NS/ARPA supplies the TCP protocol, but not the UDP protocol. DNS/1000 therefore cannot be quite "up to standard" in this respect. However, the standard does require name servers to also support TCP, at least after UDP has already been tried. The TCP-only RTE implementation has worked quite well on all name server implementations with which it has been tried thus far. In fact, the latest amendment to the standards, document RFC-1123 (Braden, *et. al.*, 1989), appears to relax the rule that UDP always be tried first, for consenting parties:

> By private agreement, name servers and resolvers MAY arrange to use TCP for all traffic between themselves. *(Section 6.1.3.2, page 76, emphasis original)*

However, the same section also reiterates the demand that UDP be tried first, and it is unclear what bearing private agreements have in the context of standards. The above quotation, along with other material from the same section, does seem to indicate that it is unlikely most server implementations require an attempt at a UDP query before a TCP query will be honored.

Regardless, the fact remains that the RTE implementation must use TCP for all queries, thereby causing a greater load on the network than if UDP were used. In the interests of good network citizenship, it behooves DNS/1000 to implement a caching mechanism which reduces the number of network accesses to be made. Hence, the inclusion of the slave mode servers into the 6.0 release.

The RTE Server Implementation

The slave mode server is a program called `NAMED` ("name daemon"), after the HP-UX program which fulfills a similar function. This program may be scheduled any time after the network is brought up. Only one copy of `NAMED` may execute at one time on a host. This restriction is due primarily to the representation of the cache as a VMA area, which is difficult to share.

Some Networking Considerations

`NAMED` uses NetIPC calls for all network communication, and for communication with the local resolver routines. All incoming queries are normally generated by the resolver on the local host, although this does not have to be the case. For instance, a local network of HP 1000s may wish to share a DNS cache located on one of the machines, such that data retrieved from the general network on behalf of any one host is made available to the entire local network. NAMED also has no requirement that a query come from an HP 1000, so long as the query is made using the TCP protocol.

Because there may be only one copy of NAMED active at a time on a host, NAMED performs its own listening for new network connections. INETD cannot be used for this purpose, since it spawns multiple copies of the servers for which it listens. NAMED listens on the "well-known" TCP port assigned by the standard for name service, port 53.

The DNS standard emphasizes that it is unacceptable for a server to block all incoming requests while it waits for a reply over a slow network link during processing of a forwarded query. NAMED therefore makes liberal use of the NetIPC IPCSelect call, reading data from a connection only when data is actually present. This allows NAMED to remain free to accept new connections and new queries on existing connections while remote servers are formulating replies.

NAMED may have several TCP connections open at a time: connections established by local resolvers generating queries, and connections initiated by NAMED to other name servers which are handling forwarded queries. NS imposes a limit of 32 sockets per program, so NAMED has provisions for queuing up connections, both incoming and outgoing, in case all its sockets are in use during a particularly busy time.

At most times, NAMED is blocking on an IPCSelect call, waiting for some network activity on one of its active sockets. The bitmaps supplied to IPCSelect will indicate that NAMED is interested in each active virtual circuit (*i.e.*, TCP connection) which has data present on the connection or is "exceptional", probably indicating that the remote side is closing the connection. The bitmaps will usually also specify that exception conditions on NAMED's call socket are to wake NAMED up, since a new connection is incoming. However, if NAMED has used up all its allowable virtual circuit sockets, then it cannot receive the new connection until an existing connection is closed. During a time that this is the case, exception conditions on the call socket are not selected for, since it would always select immediately, but NAMED cannot accept the connection. NAMED will reinstate listening for new connections as soon as an existing connection is shut down.

Cache Storage

The resource records which NAMED enters into its cache are stored in VMA. The cache is represented as a forest of binary trees. One tree exists for each label of each domain name cached; each tree stores the possibly-empty set of labels which exist directly beneath the parent label in the domain hierarchy. And, as you might expect, there is also a "root" tree, which stores the labels for the top-level domains cached.

This is a rather straightforward encoding of the domain namespace hierarchy we've been talking about all along, save for the representation of "sibling" labels. In our previous conceptual models of the namespace, we imagined a label to have multiple branches springing from it to the next level down: one for each label, or sub-domain, beneath it in the hierarchy. But the cache tries to implement a more practical representation of the namespace, and arranges the lower-level siblings into a binary tree, rooted at the upper-level label. Or, in the case of the top-level domains, at the "root" of the namespace. This is practical for two reasons: it avoids an awkward storage scheme where an arbitrary number of lower-level branches could be allocated for any given label, and it provides a binary sorting of each set of siblings, thus allowing binary searches to be performed when a name lookup occurs. The ordering of the binary tree is based on the lengths and ASCII values of the labels which comprise it.

So the cache describes the domain namespace tree in the hierarchy of these new binary trees of sibling labels, each rooted at the parent label, or at the root of the namespace. Labels are also created for domain names which are found *not* to exist, such that future requests for the same name do not require network activity to determine this all over again. These labels are specially marked, such that a query which locates this domain will be answered with an "unknown domain" indication. These entries have a certain Time To Live associated with them just as resource records do, so that the server will not permanently refuse the domain the possibility of existing.

Each label may also be the head of a linked list of resource records cached for that domain. There is currently no support for the caching of "negative" RRs which indicate that no RRs of a particular type exist for the domain.

Notice that the cache storage format would serve perfectly well as a mechanism for storing authoritative data, should the RTE name servers be enhanced in the future to allow this. Each set of authoritative data could be stored as a separate forest of trees with its own namespace root. The cache would maintain its separate root to insure cached data is never confused with authoritative data, and to facilitate clearing of the cache, if desired.

Mail/1000 Support for DNS

The needs of customers using the mail system supplied the main impetus for providing DNS functionality on the HP 1000. In fact, Mail/1000 is the only RTE subsystem which has DNS support built into it at 6.0.

Two major customer complaints have driven the decision to support the DNS in mail at 6.0. The first common complaint concerns the difficulty in maintaining mail connectivity to non-HP 1000 hosts which generate full domain names in mail messages. Many of these domain names are not in acceptable NS nodename syntax. This makes it difficult for Mail/1000 to reply to the messages.

Another manifestation of the same problem is that users on the HP 1000 are typically entering unofficial domain names for the remote hosts. These unofficial domain names are shortened to be legal NS nodenames, which allows Mail/1000 to contact the destination host over NS. But Mail/1000 has no way of knowing that the domain name entered is not the official domain name, so the shortened name is left intact in the message. These truncated domain names tend not to work as intended when they are interpreted on hosts other than the HP 1000s, causing problems if the message is replied to or forwarded to someone else. To add insult to injury, leaving an unofficial domain name in a message header is a violation of Internet mail standards, precisely because alias names tend to cause these sorts of problems. Ideally, Mail/1000 should use the same domain namespace as the rest of the Internet community.

The second common customer complaint which has forced the DNS issue concerns the load which Mail/1000 places on other computers in the network. The Internet standards specify that the DNS be used for certain mail routing purposes, as will be detailed below. But Mail/1000 has not had access to the DNS, of course. It therefore may perform incorrect mail routing. More commonly, Mail/1000 must simply pass the lion's share of outward-bound mail to another host which does know about the DNS. Mail/1000 should keep up its share of the load and properly route its own messages!

Domain Canonicalization

As we have already shown, the name resolvers provided at 6.0 can translate any arbitrary domain name into an IP address, regardless of whether the syntax is acceptable as an NS nodename. By straightforward use of the resolver routines to map domain names to addresses, Mail/1000 is able to transcend the NS nodename restrictions fairly easily.

Mail/1000 should allow "alias" names for domains to be entered in messages, since these aliases usually exist to help users who prefer to type that name for some reason. Alias names are registered in the DNS, and are located via queries in exactly the same manner as are official domain names. However, as we just mentioned, it is a violation of the standards to send a message out over the network which contains any domain names which are not "fully-qualified official domain names." Therefore, Mail/1000 must perform some DNS queries to identify alias domain names in message headers, and must replace the aliases in the header with the official domain names.

Way back at Table 2 we listed some resource record types, including a type named "CNAME" which defines the "canonical domain name" for an alias domain name. The canonical name is the "official" name we've been referring to here. So, the task of replacing alias domains boils down to Mail/1000 querying the DNS for each domain name entered in a message header field, looking for CNAME RRs associated with that domain. If a CNAME is found, the alias domain is replaced by the canonical name contained in the resource record.

This same processing allows automatic domain qualification to be performed in an intelligent manner. If an abbreviated official domain name is passed to the resolver routines, the resolver will locate the correct fully-qualified domain name. When the query for CNAME RRs is performed, the resolver will inform Mail/1000 of the fully-qualified domain name, and that no CNAME RRs are associated with it. Mail notices this and updates the domain in the header to the fully-qualified name.

Mail Routing

Table 2 listed a resource record type named "MX". This RR tells the mail system the host which is to receive mail bound for the associated domain. In many cases, an MX record simply informs the mail system to send mail to the host named by the associated domain name. This means the domain associated with the MX RR is the name of a host which may receive mail directly from the Internet.

But the MX RR allows a great deal of flexibility in setting up Internet mail routing schemes. For instance, it can eliminate the need for "source routing" in mail addresses. Source routing is the use of host routing information in the local-part of a mail address. This normally takes one of two forms: a UUCP-style path of host names separated by exclamation points, such as "`romlee!user@trendy.big-swifty.com`", or a "magic percent" path, such as "`user%romlee@trendy.big-swifty.com`". Both of these example addresses tell the Internet mail system to route the message first to host "trendy.big-swifty.com". That host will forward the message on to another host named "romlee", which will deliver the message to "user". Presumably, "romlee" is not visible to the general Internet for some reason. This is an awkward means of performing this function. The mechanism also tends to cause a lot of problems when mailers become too clever about interpreting the source routes.

The DNS is solution is to register host "romlee" in the database as domain "romlee.big-swifty.com". An MX record is also created which directs all mail bound for "romlee.big-swifty.com" to host "trendy.big-swifty.com". Now the address may simply read:

```
user@romlee.big-swifty.com
```

The average mail user need not be aware of the involvement of host "trendy". The routing information is now stored in the DNS database, rather than in the address itself.

The use of MX records also facilitates the use of "organizational" domains in mail addresses, without specifying a host name. For example, this author may be reached at Internet address "`todd@cup.hp.com`". Note that no host name is given in the address; only the sub-domain in which the host name appears is given. This is rather handy in the face of changing host names. If I ever switch to a different host within the Cupertino site, mail can easily be redirected to the new host. I do not have to advertise a new address at each change. This is accomplished through the use of MX records for the "cup.hp.com" domain, which direct mail to a local gateway host. This local host will locate the appropriate host my mailbox currently resides on, and forward the mail along.

Mail/1000 at 6.0 will query the DNS for MX records pertaining to each destination domain of a message, and route the mail accordingly. In the case of multiple MX records, the hosts will tried in order of preference, which is specified in the record, until a transfer is successful. To avoid mail loops, Mail/1000 will look to see if the local host is named in any of the MX records. If so, all MX records of equal or lower preference than that specified by the MX record for the local host are discarded. This prevents the local host from sending the message to a less-preferred host, which is likely to consult the DNS for MX records and then route the message right back to the local host again.

To Conclude

The Domain Name Service has become firmly entrenched world-wide as a vital component of

Internet networking, and the HP 1000 has had to follow suit to maintain interconnectivity. We at Hewlett-Packard hope you find this new feature useful in allowing your HP 1000s to peacefully coexist on your TCP/IP network.

References

[1] Comer, Douglas E. *Internetworking with TCP/IP Volume I: Principles, Protocols, and Architecture.* Prentice-Hall, Englewood Cliffs, New Jersey, 1991. Contains a good "friendly" introduction to the DNS which provides somewhat more detail than this paper.

The DNS standard and its interaction with the mail system is specified in a number of "Request For Comments" (RFC) documents available from the Network Information Center at the Stanford Research Institute in Palo Alto, CA. The RFCs of primary relevance to this paper follow.

[2] Partridge, Craig. *RFC-974: Mail Routing and the Domain System.* 1986.

[3] Mockapetris, Paul. *RFC-1034: Domain Names - Concepts and Facilities.* 1987.

[4] Mockapetris, Paul. *RFC-1035: Domain Names - Implementation and Specification.* 1987.

[5] Braden, R., *et. al.* *RFC-1123: Requirements for Internet Hosts – Application and Support.* 1989.

Symbolic Links on RTE-A / VCPLUS

Gary Gan

Hewlett-Packard
Software Technology Division
11000 Wolfe Road
Cupertino, CA.

Introduction

At revision 6.0 of RTE-A/VCPLUS, symbolic links have been added to the FMP file system. Symbolic links are a useful feature of UNIX[1] file systems which allow files to indirectly refer to other files. A symbolic link file can specify either an absolute path to a different file or it can specify a relative path. For example, the symbolic link file '**/help/fgrep**' may contain the path '**/help/grep**'. A program which requests to read from the file '**/help/fgrep**' will actually read data from the file '**/help/grep**'. Symbolic links are expanded when the system interprets a path to a particular file. If a component of the path being interpreted is a symbolic link to an absolute path, all of the components up to and including the name of the symbolic link are replaced in the path with the contents of the symbolic link. When a component of a path is a symbolic link to a relative path, the name of the symbolic link is replaced by the contents of the symbolic link.

Implementation on RTE-A

Symbolic links on the RTE-A file system are implemented as a new file type. All symbolic links are created as type -1 FMP files. The symbolic link file itself must reside on a hierarchical file system volume and can point to any FMP file, FMP directory, or logical unit. The symbolic link may not be a FMGR file. Since negative file types have never been allowed in RTE, this eliminates the possibility of choosing a file type which has already been defined by a user. A major advantage of creating the symbolic links as type -1 files is that this allows the RTE-A directory structure to remain unchanged. This also leads us to one of the limitations of this implementation. Since symbolic links are created as files, we are restricted to creating symbolic links in either a global directory or a subdirectory. It is not possible to create a symbolic link in the root directory of a disk logical unit. Thus, we cannot have a global directory which is a symbolic link. There is also no means to create a symbolic link to a system level root directory.

A symbolic link may refer to any arbitrary path name and may span different disk logical units. A link may also refer to a remote file using the DS transparency syntax or the link may refer to a logical unit number on the local system. Links to FMGR directories and links to FMGR type 0 files are not supported. The path name may be that of any type of file (including a directory, or another symbolic link) or it may even be invalid if no such path exists in the system. Since the link contents are not checked at creation time, it is possible to create symbolic links which point to themselves or to other symbolic links in such

[1] UNIX is a registered trademark of AT&T.

a way that they form an endless loop. This situation is detected by limiting the number of symbolic links which FMP will traverse while translating a path name. The protection and ownership of a symbolic link is ignored by the system; but, the protection and ownership of the actual file being accessed is still checked by the system.

D.RTR Program

All FMP calls which access a directory entry are handled by a single directory manager program, D.RTR. The path name resolution required to access any file is performed by D.RTR. To support symbolic links, D.RTR was modified to resolve absolute and relative symbolic links during this path name interpretation. Symbolic link support is only available in the CDS version of D.RTR. For systems which do not need symbolic link support, a non-CDS version of D.RTR is still shipped with the RTE-A product. The CDS version is included with the VCPLUS product.

When FmpOpen is used to open a file, the path name of the file is passed to D.RTR by the FmpAskDDot routine. Path names specifying a remote file using DS transparency cause D.RTR to return an error to FmpAskDDot. The same open request is then routed to DSRTR which handles the remote open. If the path name specifies a local file, D.RTR then begins searching for each component of the path. For absolute paths, the search begins in the global directory specified in the path. For relative paths, the search begins in the user's current working directory. When the path contains references to subdirectories, each subdirectory is parsed from the path and D.RTR then searches for the subdirectory specified. If the subdirectory is found, that directory is read and the process is repeated until D.RTR finds the last component of the path, the name of the file being opened. At this point D.RTR opens the file and returns to the calling program.

To support symbolic links, changes were required in the path interpretation. The check for symbolic links must be made as each subdirectory component of the path is resolved. When a subdirectory refers to an absolute path, a new path name is created by concatenating the remaining components of the original path to the contents of the symbolic link. For example, suppose the path name passed to FmpOpen is '/a/b/c/d.ftn' and '/a/b.dir' is a symbolic link to an absolute path, '/x/y/z.dir'. When D.RTR encounters the symbolic link at 'b.dir', it would create a new path by concatenating the remaining portion of the original path, 'c/d.ftn', to the contents of the symbolic link. The new path would be '/x/y/z/c/d.ftn'. At this point D.RTR will set up to search the global directory '/x.dir' and restart the path interpretation.

It is also possible that the symbolic link points to a remote directory. In this case, the new path would be passed back to FmpAskDDot, which in turn would issue the open request to the DSRTR monitor. DSRTR will send the open request to a remote monitor, TRFAS. On the remote system, TRFAS will then issue the open request to the D.RTR on the remote system. Any symbolic links encountered on the remote system are then handled on the remote system by D.RTR. TRFAS will not schedule DSRTR if D.RTR on the remote system encounters a symbolic link to yet another remote system. What this means is that one cannot access a file on a remote system if that file is itself a symbolic link to a remote file. When this is attempted, a FMP error -262 is generated. Remote access of symbolic links which point to local files is supported; but, remote access of links which point to devices is not supported. This handling of remote symbolic links is different from the behavior on UNIX when accessing a symbolic link through NFS[2]. On UNIX, the contents of the symbolic link are read from the remote link and the path is interpreted on the local system. On RTE, the contents of a remote symbolic link are interpreted on the remote system. To interpret the contents of a remote symbolic link on the local system, separate remote resolution requests would have to be generated for each remote symbolic link in a path name. This would require D.RTR to wait for DSRTR/TRFAS for each of these requests and this delay is not acceptable.

When a subdirectory refers to a relative path, D.RTR builds the new path in the same manner as it does for an absolute path, but, the search for the first component of the symbolic link will resume in the same directory in which the symbolic link itself was found. Suppose that '`/a/b.dir`' in the previous example is a symbolic link to a relative path, '`m/n.dir`'. D.RTR would just create the new path as '`m/n/c/d.ftn`' and continue searching '`/a.dir`' for '`m.dir`'. The full path for the file would be '`/a/m/n/c/d.ftn`'.

It is also possible that the final component of the path name could also be a symbolic link. This is resolved in the exactly the same manner as the subdirectory symbolic link resolution. In addition to the possibility of the symbolic link pointing to a remote file, D.RTR must also handle the case of a symbolic link pointing to a logical unit. In this case, D.RTR will open the device pointed to by the symbolic link.

There are some operations which deal with the symbolic link file itself. In these cases, the final component is not resolved. The calls FmpPurge, FmpRename, FmpSetOwner, FmpSetProtection, FmpReadLink, and FmpMakeSLink always refer to the actual symbolic link file and never to the file pointed to by the link.

Unlike UNIX systems, the access time of a symbolic link on RTE is not updated each time the link is traversed. The performance penalty of an extra disk write just to traverse a symbolic link is too high. Every time D.RTR encounters a symbolic link, the contents of the link are required for the path name interpretation. This usually requires an extra disk read to resolve a path with a symbolic link. Since D.RTR also keeps a cache of symbolic links and their contents, this extra disk access can sometimes be eliminated. The cache is updated on a least recently used algorithm, and the size of the symlink cache is configurable at link time for the CDS D.RTR.

There are four new FMP errors specific to symbolic links.

```
FMP -260  - Too many symbolic links in path

            D.RTR traversed more than 8 symbolic links.   This is
            probably a closed symbolic link loop.

FMP -261  - Symbolic link results in illegal path
```

[2] NFS is a registered trademark of Sun Microsystems, Inc.

Symbolic link interpretation yields a path name greater than 63 characters.

FMP -262 - Symbolic link must not be remote

An attempt was made to access a symbolic link on a remote system and the remote symbolic link referred to a remote file.

FMP -263 - System does not support symbolic links

At attempt was made to create a symbolic link on a system which does not support symbolic links. A CDS version of D.RTR is required and programs accessing symbolic links must be linked with either $SFMP or $SCDS.

Changes to the FMP library

To support symbolic links, the FMP library also required some major changes. To minimize code growth for users who do not want symbolic links on their systems, different versions of the FMP libraries will be included with RTE-A and VCPLUS. The $SFMP library and the CDS version $SCDS both include support for symbolic links and the $FMP and $CDS libraries do not. The majority of the code growth is in the FMP masking routines. There are also some new FMP routines to create and access symbolic links. The FmpMakeSLink routine creates a symbolic link file and the FmpReadLink routine reads the contents of a symbolic link.

In general, the default for FMP's treatment of symbolic links is to follow the link. For instance, when the user calls FmpUpdateTime and passes the name of a symbolic link, the update time of the file pointed to by the link will be returned. If the link points to a non-existent file, a FMP error -6 will be returned by FmpUpdateTime. For all the FMP calls which return directory information for a given file name, there is an optional parameter which specifies how to process symbolic links. The default, when this parameter is not used, is to follow links. This allows symbolic link access to be transparent to most applications without any code changes required.

The default for FMP's masking code is to also follow symbolic links. By default, FmpNextMask will return the directory information for the file pointed to by a symbolic link. For symbolic links pointing to non-existent files, the directory information for the symbolic link itself is returned. A couple of new mask qualifiers were added to accommodate symbolic link handling. The 'L' qualifier tells masking to always return the directory information of symbolic links. The 'G' qualifier is also new at revision 6.0. This qualifier, to be used in conjunction with the 'S','K', or 'E' qualifiers, causes masking to treat symbolic links to directories as regular files. When the 'G' qualifier is specified, masking will not recursively search symbolic links to directories.

There are a number of restrictions on the use of symbolic links pointing to remote files. Since access of these files uses DS transparency, we are subject to the limitations of DS transparency. A user's working directory may not be set to a remote directory and remote type 6 files can not be executed. Another restriction deals with the use of masking and the recursive searching of symbolic links to remote directories. Fmp masking will not recursively search subdirectories which are linked to a remote directory. Masking will recursively search a remote directory when the search starts at the remote directory. For example, suppose we have a global directory '/home.dir' with a subdirectory 'local.dir' and a symbolic link 'remote.dir' pointing to a remote directory. The mask '/home/@.ftn.s' will only search '/home.dir' and '/home/local.dir'. The mask '/home/remote/@.ftn.s' will recursively search the remote directory.

Prior to this revision, FMP masking had an upper limit of 32 for the number of subdirectories which could be nested during recursive searches. Due to the 63 character path name limitation of RTE, this limit was probably not reached very often. With symbolic links, it is conceivable that this limit could be reached. The limitation is based on a 32 word stack which is defined in the DIRDCB array which is passed to the fmp masking routines. Increasing this stack size was considered; however, this would require code changes for many user written programs which use fmp masking. Currently, the upper limit is based on how many of the nested subdirectories were traversed from symbolic links. Of the 32 word stack, one word is used for each subdirectory being pushed on to the stack. An additional 2 words is required for each symbolic link to a subdirectory. These 2 words are used to store the parent directory's directory block address. These are not required for a "normal" push to a subdirectory because the parent's block address can always be read from a subdirectory's directory header, but this is not the case for symbolic links.

LNS utility

The LNS utility is the user level means for creating symbolic links on a system. LNS can create symbolic links to files, directories, or devices. To create a symbolic link to a file or a directory, the name referenced in the link must be in hierarchical format. The symbolic link file itself can be created in any FMP directory. The name of any symbolic link to a directory must also have a type extension of '.DIR'.

LNS can be used to create a single symbolic link or it can also be used with a variable number of fmp masks to create a group of symbolic links in a single destination directory. For example: the command 'lns /moreprogs/@.run /programs/' will create a symbolic link in the /programs directory for each file in the /moreprogs directory which has a type extension of '.run'. With this usage of the LNS command, it is important to be aware that when a mask which specifies a relative path is used, the contents of the symbolic links being created will contain this relative path. When one of these symbolic links is evaluated, the relative path will be evaluated relative to the link itself and not relative to the working directory in use at the time the link was created. For example, the command 'lns @ /newdir/' will create a group of symbolic links in the /newdir directory which point to themselves. The mask '@' will find all the files in your current directory, but, since the mask was relative, the relative path will be written into the symbolic link. The symbolic link /newdir/new.ftn will point to 'new.ftn'. This will result in a circular link and D.RTR will reject any access with a -260 error.

Examining the contents of symbolic links

There are two easy methods to examine the contents of symbolic links. The DL program has been enhanced to report the content of a symbolic link when the 'Z' option is requested. The other method is the use long format of the LS program, which is also new at revison 6.0. (The commonly used short hand for 'ls -l', 'll', can also be used. The file '/programs/ll.run' can be made a symbolic link to 'ls.run' and "ls" will notice that it has been invoked as "ll".)

When using DL and the 'Z' option is not requested, the default mask qualifiers are used. This means that symbolic links are followed by default. The directory attributes of the file pointed to by a symbolic link are reported for a symbolic link. Specifying the 'L' mask qualifier will also cause DL to report the directory attributes of symbolic link files. For performance reasons, if DL is used just to display the file names in a directory and no other directory attributes are requested, DL will automatically qualify the mask with the 'L' qualifer.

The LS program, by default, does not follow symbolic links. All masks used by LS are qualified with the 'L' qualifier unless the LS '-l' option is specified. When the LS '-f' option (lsf) is used, files which are symbolic links are designated with a trailing '@. (Note that the options to LS are case sensitive.)

Programming Considerations

For the majority of programs, no changes are required to support symbolic links. There are a few areas which may require some attention.

The FmpFileName routine will always return the full canonical path name of the file which is open, which will not include any symbolic links used to specify the file. (The same is true for the FmpShortName routine.) This may or may not be desirable depending on the application. For example, Edit/1000 uses FmpFileName to report the name of the file being edited. In this case, we decided that Edit should report the name of the actual file being edited. Sometimes the name of the symbolic link is desired instead the name of the destination file. Prior to 6.0, Ftn7x used the FmpFileName routine when building the default list and relocatable file names. With symbolic links, we decided that the name and location of the default files should be based on the name and location of the symbolic link file and not the files pointed to by the link.

When an application needs to overwrite an existing file, sometimes rather than rewriting the existing file, a scratch file is created in the same directory as the file overwritten. If the scratch file is written successfully, the original file is purged and then the scratch file is renamed. In this case, if the destination file specified is a symbolic link, it must be decided which file is supposed to be overwritten. For the CI 'CO' command (FmpCopy), FTP, and DSCOPY, when the destination file is a symbolic link, the file pointed to by the link is the file which is created or overwritten.

Another area which requires some consideration is the use of fmp masking. Programs which use the fields in the directory entry directly may need to be modified. The MPACK and FOWN utilities were modified to always add the 'G' and 'L' mask qualifiers to every mask. For these utilities, it is not desirable to follow symbolic links. Changes are also required for any programs which perform a logical backup. Again, the changes depend on the desired action. For FST, the default operation is to backup the symbolic link files and not follow the links. Optionally, the user may request that FST follow symbolic links and backup the data pointed to by the link. TF will not follow symbolic links; it will only backup the contents of the link files.

Usage considerations

The D.RTR in the BOOTEX system is a non-CDS D.RTR. This means that the BOOTEX system does not support symbolic links. It is important to make sure that your system's boot command file does not attempt to reference any file which is a symbolic link.

Conclusion

The addition of symbolic links to the RTE-A file system is a useful feature which can be utilized in many different applications. Symbolic links can be used to alleviate disk space problems for some directories. Instead of moving entire directories to larger disk volumes, symbolic links can be used to spread the contents of one directory across several disk volumes. The capability of referencing device logical units in a symbolic link allows for an easier interface to fmp device files. Symbolic links are also very useful in the software development process. The creation of shadow development trees consisting of symbolic links to a master source tree allows for an easy development tree without requiring the disk space for a copy of the entire master tree. Only the files being changed need to be copied into the user's development tree.

Appendix A

```
LNS -- Create a symbolic link to a file or directory.  92570-17018 REV.6000

Usage: lns [-pv] filedesc symlink_filedesc
       lns [-pv] file1|mask1 [ file2|mask2 ... ] dest_directory/
       lns [-pv] directory symlink_directory

Options:
       -p  : OK to overwrite existing files
       -v  : verbose mode
       --  : end of options (required if mask begins with '-')

Description:

      lns creates symbolic links between:

      - 'filedesc' and a new or existing file 'symlink_filedesc'
      - 'file1' and  a new or existing file named 'file1' in the
        existing 'dest_directory'
      - 'file1|mask1', 'file2|mask2', ... and new or existing files of
        the same name in the existing 'dest_directory'

      'filedesc' is the name of the file to be referenced in the symbolic
      link file, 'symlink_filedesc', which will be created.  The symlink
      file must be a FMP file and can be created in any existing
      directory.  (The symlink file itself may not be a FMGR file.)

      For symlinks to FMP files,  the contents of the link, 'filedesc',
      must be in hierarchical format ( /dir/file ).

      For symlinks which reference directories,  the type extension
      'dirname.dir' must be specified in the file descriptor.

      Symlinks to FMGR directories, FMGR files and FMGR type 0 files are
      not supported.

Examples:

      CI>  lns subdir/name fname
        Create a symbolic link to 'subdir/name' in the file, 'fname', in
        the current working directory.  Listing the file, 'fname',  will
        show the contents of 'subdir/name'
```

```
CI>  lns /progs17/@:::6 /progs18/@:::6 /programs/
  Create symbolic links to all type 6 files in '/progs17' and
  '/progs18' in the '/programs' directory.

CI>  lns /system.dir sys.dir
  Create a symbolic link to the /system directory which will
  look like a subdirectory 'sys' in the current working directory.
  A directory listing of the 'sys/' subdirectory will display the
  contents of the '/system/' directory.

CI>  lns 8 /devices/magtape
  Create a file "magtape" in the devices directory which refers
  to LU 8.

CI>  lns -- -prog::dir1 /dir/
  Create links in the /DIR directory to all files in the /DIR1
  directory which pass the mask '-prog'.

CI>  lns /mydir/tests/@>bucko ./
  Create links in the current directory to all files in the
  '/mydir/tests.dir' directory on the remote system 'bucko'.
```

PAPER NO.: 1013

TITLE: Advanced HP 1000 Assembly Language:
Writing CDS Code

AUTHOR: Alan Tibbetts
Hewlett-Packard Co.\Consultant
3498 Gibson Avenue
Santa Clara, CA 95051

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

PAPER NO.: 1014

TITLE: Using BSD IPC to Synchronize RTE and HP-UX System Times

AUTHOR: Wendy King
US Naval Observatory
34th & Massachusetts Ave NW
Washington, DC 20392-5100
(202) 653-0909

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

Paper # 2001

SOFTWARE LIFE CYCLE CONTROL

EVOLUTION OF SOFTWARE CONFIGURATION MANAGEMENT

Jim Sterne
Softool Corporation
340 South Kellogg Ave.
Goleta, CA 93117
(805) 683-5777

INTEREX
1992

Library control and source code control are insufficient to ensure software change and configuration management in today's complex software development environments.

Solutions are required which manage and synchronize all software components. They must run on multiple hardware platforms and support network architectures. A configuration management product must support the entire software development and maintenance life-cycle. It must also be flexible enough to adapt to development methodologies already in place, and address the requirements for distributed computing environments, integration CASE frameworks, client/server architectures. They must be able to meet the challenge of tomorrow's object oriented environments.

History of Code Control

Back in the 1970's there was source code. Tracking it was not simple because it can be easily changed, so tools were created to help. Library management tools kept a lock on sacred directories and utilities like SCCS (Source Code Control System) on Unix were provided to stop two people from changing the same file at the same time. These tools kept track of all the different versions.

In the 1980's configuration management tools came along like Softool's Change and Configuration Control (CCC). CCC was designed to know which version of each file belonged to a given release of the whole application. To help automate compilations, these tools also tracked object code and the executables. The whole point was to manage the migration of code from development to test to release.

Modern code control has expanded to include more component types:

1970's	1980's	1990's
Source Code	Source Code Object Code Executables	Source Code Object Code Executables Documentation DBMS Calls Diagrams Manuals Charts Models 4GLs Etc.

A 1990's configuration management system must manage all of these across the entire software life-cycle, across heterogeneous networks, and while communicating with other CASE tools.

Enterprise Life-Cycle Management starts with the basics of component management, application management, package management, and dependency control. Then, it takes into account the integration of tools in the software engineering environment.

Component Management

Component management is the most basic of all these facilities. Component management involves tracking files from version to version and recording:

who made the change
what changed - line by line
when the change was made, and
why the change was made (STR, ECO, etc)

While SCCS can only handle ASCII files, today's SCM tools go further in that they can support all types of files.

Application Management

Using SCCS relies on the software engineers to remember which version of each file will survive a system integration test. Today's SCM tools have an automated way of providing different views of the code. This is a much easier way of tracking which version of each file or object makes up a given release of the application. Which changes belong to development, which are in test and which have been released are tracked for the developer.

Package Management

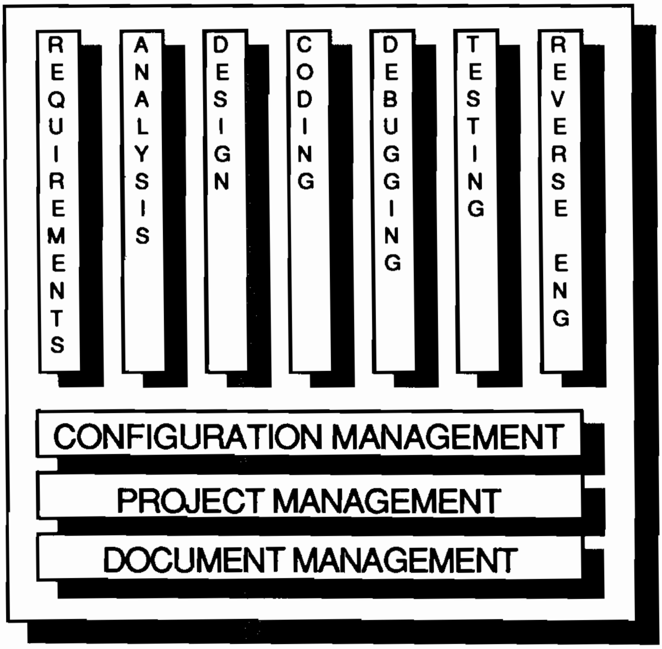
Package management assures the flow of changes through the life-cycle. No changes are left behind, no incompatible changes are promoted at the same time, and backing out harmful fixes, which cause more problems than they solve, is relatively easy.

Dependencies

The Unix *make* utility insures that unaltered source need not be re-compiled. Today's SCM systems go a step further by scanning the source code for include files, source calls, database calls, etc. Using this information, they can create dependency trees. Products like CCC take on the responsibility of automating and controlling the build process, to insure the correct versions of code get compiled together.

Environment Integration

The modern software development environment is frequently described in current literature by the "toaster" model. The vertical slots represent tools which address the specific software life-cycle functions, while the horizontal tools cross the entire life-cycle:



Surrounding these different tools is the environment. This determines how the tools communicate with each other, and how they are viewed by the end user.

Why CASE Integration?

To understand the issues of tool integration, it's necessary to understand the purpose. Integrating these tools reduces the efforts required to create software by sharing responsibilities for data management, process management, user interface requirements and more.

Data Management

Data management is very important to the vertical tools. The requirements data should be useable by the analysis system which should be able to output it's findings to the design tool, which should be able to feed the code generator.

For the moment, it is enough for the configuration management system to be able to capture changes made to any type of data created by any type of tool. As long as the SCM tool can recreate previous versions, return them to the proper editor in compatible format, and determine which versions belong together, it is serving its purpose.

Today's configuration management systems don't try to interpret the contents of these objects. Whether they should and what their responsibility should be is the subject of another paper.

Process Management

Process management commands much more involvement from SCM tools. SCM is responsible for the release cycle and, therefore, must be in the middle of approval and promotion functions. Moving software from development to release is much easier if the SCM system interacts with the tools used to create the software. As one integrated CASE user said, "When doing SCM is easy, the engineers actually do it!"

User Interfaces

Another factor in the easy-to-use category revolves around the user interface. Motif, OpenLook, Microsoft Windows and the whole GUI/drag-and-drop world are attempting to make the user feel at home. The Macintosh interface is the best implemented, in that there are common commands and methods among all Macintosh applications.

In a CASE environment, the level of interface integration for SCM tools should extend well into the other tools. It's not enough to have a familiar look and feel. While SCM does not yet need to have an in-depth knowledge of a file's contents, it does need to be integrated to the point where the software engineer has easy access, without inhibiting their day to day operations.

SCM must be available without being intrusive. The check-in/check-out paradigm is the most common approach. The engineer shouldn't have to invoke another application to get this job done. Instead, it should be a simple selection in the tool in use. If a design diagram is being edited, there should be a check-in button in the editor's window. The same goes for requirements documents and source code. Check-out, edit, save, check-in commands must be the standard.

Methods of Integration

The integration of CASE tools is evolving. There are three steps on the road to integration: point-to-point communication, broadcast messages, and true object technology. So far we have taken the first two.

Point-To-Point Communication

Requests by CASE tool customers have caused CASE tool vendors to write interfaces between the tools. These point-to-point integrations are hard-coded so that one tool can share data with, or call a function of another tool. This method has some drawbacks.

The modifications are specific to specific tools. It is not possible for vendors to write interfaces to all the tools their customers may be using. The result is of mixed value when only half the tools you're using are on speaking terms.

This approach also creates overhead for vendors. They have to write an interface to each tool they wish to communicate with. Maintaining release level compatibility among those tools requires additional resources and creates yet another complex configuration management problem.

The best example of this approach is from IDE (Interactive Development Environments, San Francisco). Their C Development Environment is integrated to specific "Best-of-Class" tools. Instead of trying to interface to the world, they have selected tools which best compliment each other. IDE's Software Through Pictures, CenterLine's CodeCenter, InterLeaf's document management system and Softool's CCC are tightly coupled.

The upside of this approach is that IDE is delivering products today which are fully integrated and useful right out of the box. Because they are oriented toward this particular environment, they can offer a deeper level of support for the integration. They know what to expect from the interactions.

Broadcast Message Systems

The next step in the evolution of CASE product integration involves broadcast messaging. Here, an underlying facility handles the transmission of generic messages between tools.

A generic message to Softool's CCC from IDE's Software Through Pictures (StP) might be to check-out a particular structured design diagram. StP does not know that CCC is the SCM tool in use. It only knows to send a generic request in a pre-defined format via the message server. The message server finds whichever SCM system and submits the request. CCC does not know that StP has made the request. It only knows to deliver the requested object to the requested directory.

The advantage of this approach lies in the generic nature of the messages. If one of the tools must be replaced, the integration can still be valid, provided the new tool follows the protocol of the message server. In this case the window used for editing has a generic check-out selection which emits a generic check-out request to whichever SCM tool is registered at the time.

A great deal of time and effort for the different vendors is saved with this approach. They only need to write one check-in/check-out routine, instead of one for every SCM tool available. Similarly, Softool need only write one interface to cover all the possible CASE tools which would store their components in CCC.

There are a number of technologies supporting the broadcast message paradigm. IBM's AD/Cycle is still in the process of being born and is more often described as "marketecture", rather that technology. DEC's Fuse is further along, but is suffering from rumors of being replaced by DEC with alternative technology. ToolTalk from Sun has just seen the light of day and is offered as part of the operating system, rather than an additional product to purchase. Sun has just licensed ToolTalk to Silicon Graphics, so it's gaining momentum.

The best example of messaging today is provided by HP with SoftBench. HP has licensed SoftBench to both IBM (called WorkBench for their AIX RISC system) and to Informix (called ToolBus), to position SoftBench as a *de facto* standard. SoftBench surrounds HP's Broadcast Message Server with a Motif-based user environment.

This interface comes with a suite of tools such as file management, editors, and debuggers. Other vendors incorporate their tools into the SoftBench environment with HP's interface-building tools. The result is a master SoftBench window supporting tools from many different vendors with a common look-and-feel. The software engineer need only learn one interface paradigm. This is the closest we've seen to the Macintosh concept of similar tools using similar commands and methods.

Object Orientation

Object technology is getting more and more attention along with client/server computing and open systems. As we try to arrive at a common definition of object orientation, CASE vendors like Softool have found a way to get part of the way there through broadcast messaging systems.

One important feature of object orientation is information-hiding, or encapsulation. Individual software modules communicate freely with other software modules, instead in the linear way they do now. Each module is made up of an interface or specification portion which handles the interactions with other modules, and an internal or method portion which handles the calculations or processing.

The intended result is software that is far easier to maintain due to its modularity. Either the communication portion needs to be debugged and upgraded, or the internal algorithms need work. Either way, the level of integration and testing required diminishes greatly.

While nobody is inclined to re-write all of their applications to take advantage of these benefits, we can begin to move toward that paradigm through broadcast messaging systems.

In the SoftBench environment, Softool's CCC is treated as an application object. The other tools need not know about CCC's internals. They only need to know how to communicate with it. This is the first step toward encapsulation.

The world of object technology is getting serious about standards and we will soon agree on inter-application communication protocols. Then we can agree on inter-module communications. There are dozens of individual modules within Softool's CCC capable of communicating with modules from other applications. When the standards are put into practice, CCC will be a full fledged member of an object CASE integration.

The Future

There are many other attributes of object technology (inheritance, polymorphism, etc). SCM systems will have to address these tomorrow, just as they are managing compilation dependencies today. It will start making suggestions about what types of changes to make and which to avoid based on object impact analysis.

The SCM system will be responsible for knowing which objects are compatible with which others. This is a large task, considering that objects will not be in a confined environment, but loose on the network. SCM tools will have to address each object interacting and trading data with the next: Version 12 of object X is compatible with version 6 of object Y, but not version 5 of Y.

SCM will become the responsible party for software reuse. It might be tied into the syntax of the object environment to advise the developer of previously built objects which might fill the need of the new one being designed. SCM will point out that changing one more attribute of an object might cause it to be identified as a whole new species.

Object oriented operating systems are going to have to incorporate configuration management as an inherent function. Then, versions of software objects can be kept in synch with versions of spreadsheets, contracts and payroll deductions.

SCM tools will grow in the future to become the process management system, incorporating the automated test environment and tied directly to the project management system. They will lead the developer through the development/test/release cycle based on input from product managers' scheduling systems and test results. They will identify softwarc components which should be re-written due to excessive maintenance and provide upper management with progress reports.

Conclusion

As the deployment of software becomes more and more sophisticated, the tools we use to develop and control software must keep pace. The implementation of integrated CASE environments will go a long way toward moving us into an object paradigm. Software configuration management will become more and more integrated into the foundation functionality necessary to create software.

We can be sure that we will face new challenges and demand more and more from our computers. This will certainly keep the progressive hardware manufacturers and the software developers in business for a long, long time.

PAPER NUMBER: **2002**

TITLE: **ACCESS & RESOURCE MONITORING**

AUTHOR NAME: **Y. D. S. ARYA**

ADDRESS: **COMPUTER CENTRE**
INDIAN INSTITUTE OF TECHNOLOGY
KANPUR - 208016 (INDIA).

TELEPHONE: **250151-8759**

1. INTRODUCTION

A Computer System is a collection of its various hardware modules, including attached peripherals. The cost incurred on the system towards hardware and software is recovered through the utilization of the system resources like: CPU, Memory, Disk Storage, Tape usage, Terminal usage,and peripheral usage. Taking into account the cost of the different modules and their expected life, the unit cost of the each system resources can be calculated.

System Accounting generates data for the main resource usage like CPU, KCS(Kilo-core seconds), TTY Connect and Disk storage. The data for the peripheral usage like printers can be generated through Device Interface Script. This data can be used to charge every user for the resources usage within one accounting period, usually a day.

At the time of installing a user in the System, the user may be assigned an initial Computational Budget. The daily charges are to be subtracted from this budget. The Computational Budget should be updated after every accounting session and the user may be denied system access after no more Computational Budget is left.

To have a fair distribution of some of the critical resources (not directly controlled by the Operating System) like Line Printer, and Plotter, the users should be allowed a limited amount of such resources within a period. A user may be denied the resource access after reaching the limit, irrespective of the Computational Budget.

Denying a resource access beyond a certain limit and disallowing the Login after the Computational Budget is exhausted, enforces a discipline among the users to use the system resources in an optimal way. This increases the system productivity. In addition, the budgeting policy may be used to allow users from other organizations to use the System on a rental basis.

This paper describes the implementation of the above for a Network of several computers running the HP-UX Operating System. Awareness of HP-UX system accounting is assumed as the resources usage data is collected through System Accounting (a HP-UX tool). Section 2 gives an overview of the existing accounting facilities in HP-UX and its shortcomings in a multi-user environment . Section 3 illustrates the implementation of the system in the environment of a Network of computers connected via LAN. Limitations and suggestions are given in Section 4, and we conclude with Section 5.

2. EXISTING FACILITY ON HP-UX

The HP-UX system Accounting facilities [1] provide the means to:

- record disk space usage for individual users,
- record connect session data,
- collect resource utilization data (such as memory usage and execution times) for

individual processes, and

- generate summary files and reports that can be used to analyze system performance and usage.

2.1 Shortcomings

The facilities do not provide any utilities for charging users for the resources they have used. Further, there is no mechanism to have a "budget" in a system and to deny access to users who have run out of budget. As a consequence, there is no billing mechanism either. Lastly, there is no monitoring, and regulating the use of peripheral resources such as printers and plotters.

3. THE PROPOSED SYSTEM

The components of the Access and Resource Monitoring System are: charging policy for the system resources, creation and maintenance of Accounting Master File and various other data files, enforcement, multiple systems accounting, and the Monitoring Software.

3.1 Charging policy for the System Resources

The charging policy is based on the recovery of the total value of the System including overheads like cost of air-conditioning, electricity etc. in ten years (expected life of the System) time. The following initial assumptions have been made.

(a) Costing calculations assume an uptime of 80% while actual uptime may be more.

(b) The average utilization of the System is 50%.

(c) A typical job uses utmost 50% of CPU time.

(d) The depreciation rate for the total value of the system is 10% per year.

The resources to be billed are: CPU Seconds, Kilo-core Seconds, blocks of storage used per day (in hundred block units), number of hours of terminal connect; and the number of pages used on the different printers and the plotters.

3.2 Accounting Master File

Accounting Master File contains data pertaining to every user of the system. The data consists of information fetched from the system password file and group file, and additional information like Computational Budget, daily charges, cumulative charges. At the end of a fiscal period, Address File, containing the billing addresses of the users, is used for the preparation of bills.

3.3 Monitoring Software

A package of shell scripts is there for the tasks of: maintenance of Master File, creation of intermediate data files, updating of system password file and running of the accounting programs.

Also, in the case of peripheral accounting, the existing interface scripts have been modified to yield the resources usage data. This data is converted into charges at the time of daily accounting.

3.3.1 File Maintenance

Tasks involved are: creation of new user record, update of user record, loading of user charges, and information retrieval .

The following files are created through programs for the purpose described.

- Charge file: for resources usage data,

- Excess disk usage file: users exceeding the disk storage limit,

- Negative balance file: users with no more budget left to their credit.

3.3.2 Modification in Interface Scripts

The peripheral interface scripts are modified to record the usage by individual users. Care is taken to allow rational quota to different users depending on the need.

3.3.3 Multiple System Accounting

Although the resources usage data is collected on individual systems separately, the account monitoring is done at a single place only, to facilitate single point account administration. For this the restrictions imposed are of an unique user-id across all systems, there is only one accounting master system, and for the enforcement of policies consisting of a number of systems is subjected to a single action.

3.3.4 Enforcement

Users with no more Computational Budget to their credit are to be deactivated from the System. System password file is modified for the purpose. The same action is also taken for exceeding disk storage limit.

The monitoring software runs with a single command to carry out its complete task with the help of command files which are in-built within the monitoring software.

4. IMPLEMENTATION

The hardware configuration of the system for which the "Access and Resource Monitoring System" is implemented is as follows. There are four HP-9000/850S (super-minis), four Turbo-SRX workstations, four HP-9000/360 SRX workstations, twelve HP-9000/330 workstations, and one Convex C220 (mini-super) computer. The systems are interconnected via LAN. For access purpose, the systems are divided in four separate groups and so there are four distinct Password files. A user may be given access to one or more of the four groups. One group contains more than one systems with one password file, which is managed through Yellow Pages. The Login-name and id for a user is unique across all the four groups, to facilitate single point computational budgeting apart from other advantages.

The unit cost of the system resources for each group has been calculated separately taking into account the costs involved for the systems in each group. Users are provided the rate list to make them aware of the costs of different resources.

4.1 Master File

For every user of the system, there is one entry in this file. The user record contains: Login name, User Group, User's Personal details, Charge head for each day of the month(fiscal period), computational budget, cumulative charges, previous charges (to facilitate yearly accounting). After assigning a login name to a user, an entry is also created in the Master File. Actually, the login form for a new user also contains the computational budget, apart from the other details of desired password and other personal details of the user.

4.2 Resource Usage Data Collection

System Accounting is turned-on on each system on the Network. Through "cron", daily accounting scripts: Runacct and Dodisk are processed daily. The script Runacct contains an extra local script which converts resources usage units into charges and prepares a file which contains Login-name and charge for every user of the system.

The resources usage data for peripherals: line printer, plotter, laserjet, is recorded at the time of processing the request through the device interface script. This data is also processed once in a day to convert these usage units into charges.

4.3 Accumulation and Updating of Charges

There are twenty-five computer systems for which single budget accounting is done. The Master File is maintained only on one of these systems called the Accounting Master System. The charge file of each system is copied onto the Accounting Master System. These charge files are loaded into the Master File to update the charges of every user for the usage on all the systems in the network.

4.4 Suspend File Preparation

After update of charges for the day, the Master File is processed to generate a list of the users with no more Computational Budget left to their credit. Another list is prepared from the disk usage data which contains the users having excess storage occupation, as disk storage is counted to be a critical resource.

4.5 Password File Update and Login-name Deactivation

The list prepared in the last Sub-section contains the users who should be denied access of the systems. It requires modification in the system password file of each group for these users, so that the initial login shell is not given to these users when logging in. The user entry for such a user in the Password file is modified to contain a script name in place of the Shell. The script displays a message giving the reason of deactivating the Login-name and quits. The change of password is not preferred as re-activation later becomes easy as only the shell field need be changed.

4.6 Re-activation of Deactivated Login-names

After Computational Budget is exhausted the user is allowed to get it enhanced. This added budget is taken into account at the time of the next deactivation. In fact, before the daily account update, all the requests of budget extensions are processed and before preparing suspend files, all the Login-names are activated, so that in the session which follows immediately, only those users are listed whose present status requires suspension. The same procedure is adopted for disk storage limit extension also.

4.7 Peripheral Accounting

Every user of the system is allowed a fixed number of prints per week from Line Printer, Laserjet Printer , and Plotter. However, the provision has been made for enhancing limits on a per-user basis on demand. The print scripts do not obey the request of a user who has consumed the quota for the current week. The user is sent a mail explaining the reason of disobeying the request.

4.8 Monthly Billing

At the end of every month, a usage report is generated. This report is in the form of a bill. Separate bills are prepared according to the grouping of users. The grouping is based on the common financial head of the users.

Apart from billing, other useful reports are also prepared like: high charges user list, total charges of all groups at a glance, summary reports. These reports are used to check possible misuse and to help in suggesting better computing techniques to heavy users.

4.9 Utilities for User Help

As there are a number of restrictions for resources usage, users are provided various commands to tell the resources usage statistics by a particular user. The statistics of interest are: computational budget left, pages printed in the week and plots taken in the current week, and also extra quota for the resources given to an individual. This is to save user from unhealthy situation of denial of access to a resource or the system access itself, all of a sudden.

Budget extension requests are accepted through mail. Of course a user denied access can not send a mail. This encourages users to monitor their usage regularly.

5. LIMITATIONS AND SUGGESTIONS

The deactivation of login name is done through the change of login shell which has its limitation in the network environment as the user can still access the files via FTP and RCP commands. However, the user is still charged for the files stored in the area. Care has to be taken for disallowing the CHSH command which enables a user to change the login shell.

The Monitoring System implemented keeps the day wise record of the cumulative charges of all the systems for every user. It may be desirable to keep the record of charges of individual system separately for later reference. Also, the resource units are not stored in the master file to keep it of reasonable size as there are twenty five systems to monitor. However, the provision can be made to keep these details also if there are less number of systems to be monitored.

Another thing which may be quite important is that the accounting interval is twenty four hours. After every accounting, the user is given access to the systems even if there is a very small budget left, which may be insufficient for the next twenty four hours. This is more significant in case the user is an outsider to the organization. There are two solutions to this problem, one is that a policy of certain advance budget may be made to take care of overshooting the budget; secondly the accounting interval may be reduced.

- Access & Resource Monitoring -

Hard disk quota has been implemented in HP-UX 8.0, but the soft quota policy may still be used with the older versions of HP-UX. Soft quota allows the user to use any amount of disk space but penalizes the user by not allowing system access next day.

6. CONCLUSION

The policy of access monitoring has been very successful. The resources are used in the most efficient manner by the users. Notably, users do not leave unwanted files on the disk as it consumes budget, users are encouraged to minimize the terminal connect time as this is a big resource of consuming the budget. The wastage of resources like: Line Printer, and Plotter has been minimized because of the weekly limit. No user activity of system misuse goes unnoticed. The organization is able to sell computer time to outsiders and there is a rational charge instead of flat clock charge for the user.

There is extra work for the system administrator because of the Monitoring System. The System Accounting is to be run regularly and user requests for the account update have to be processed promptly.

REFERENCES

[1] HP 9000 Series 300/800 System Administration Concepts, HP-UX 7.0, September 89.

Paper 2003

Auto-Backup for UNIX / MPE / DOS

By Uwe Hinrichs, Andy Ibbotson, Richard Preece

HI-COMP Hinrichs GmbH
Eichenlohweg 24
2000 Hamburg 60
Germany
+49 40 630 4011

Why this Strategy?

An automated Backup, Archival and Retrieval system is a concept that is essentially market driven - it follows computing trends and demands. Current advances in open systems and networking play a major role in the demands on any strategic solution. The system must be capable of running on all major versions of UNIX , and indeed, on any future version too. After all, this is one of the reasons for the development of networked systems. Additionally, it should support Personal Computers and PC networks and, though harder to achieve prior to POSIX, existing proprietary operating systems too.

Cost Justification

Before one dictates how things should be done, the cost must be justifiable. Unfortunately, because they usually have their hands on the purse-strings, accountants need to have concrete price-versus-performance figures. Accountants may talk about intangible assets amongst themselves but the technical man who mentions them to the accountant would be ridiculed. This is not to admit that talking money is the refuge of those who have lost the technical argument - the benefits must, quite rightly, be accountable.

The greatest long-term cost is certainly the unproductive operator during systems backup. This can be reduced by the following:

- Reducing the backup time by means of data compression and faster storage.

- Reducing the amount of backup storage media to be managed.

- Backup automatically, freeing the operator from the necessity to start the process.

- Backup unattended, freeing the operator from such unproductive duties as manually changing the media.

- Having simple and identical user-interfaces on all platforms, reducing needs for training.

This will free the operator to perform more productive tasks. Other savings will be realized by reduced media usage due to selective storage privileges and media management and storage requirements.

Potentially disastrous costs are incurred when critical data is lost. Consider the loss of just one file necessary to perform a task that lies on the critical path of a design project. This example is especially important in the increasingly market-driven businesses where the cost emphasis is on the speed of product inception - hence the current acronym DFM - Design For Manufacture. Immediate access and retrieval of that vital data is paramount.

Current computer industry trends show a dramatic increase in desktop computer power and increasingly sophisticated software (not to mention the use of color), which is resulting in a 20 to 30% annual increase in data storage on disc and mass storage media. These trends are presenting users with numerous backup, archival and retrieval management problems.

Distributed computing environments, frequently containing hardware from different vendors, need a consistent and compatible tool, capable of backup, archival and retrieval of data across the network. This must be easy to configure and control for the many complex networks that exist.

Let us take an objective view of what is needed in order to ensure that vital data is not lost. While many current tools address only the need to backup or to archive, other requirements are to keep the stored data secure from theft, damage or disaster, while allowing it to be recovered as quickly as possible as-and-when necessary.

Backup and Archival Requirements

- The software should be compatible and identical (in the perception of users) across the whole network.

- There must be enough online data storage capacity to allow unattended operation.

- Backup and archival should be automated.

- Full advantage should be taken of top-end mass storage devices as centralized backup devices for multiple machines.

- The process should be flexible enough to utilize distributed devices (when necessary).

- In the event of failure of a scheduled job on the central device, an alternative job should be activated automatically.

- Errors should be logged in order that the jobs can be monitored and reconfigured, if necessary. Confirmation that the job was sucessful is important.

Security of the Stored Data

- The centralized device should be in a physically protected environment, safe from disasters such as fire, flood or earthquake.

- Logical security protection of the stored data should be provided.

Location and Retrieval of Stored Data

- All attributes of stored data should be logged for retrieval searches.

- The data should be directly restorable, across the network, to the original operating system.

- The fastest restore possible should be achievable, with random access to the relevant stored data.

Achieving the Goals

To realize the above points the following are required:

- A suitable (centralized) mass storage device.

- Software on each node to perform the backup.

- Software on the "backup engine" (the machine to which the mass storage device is connected), in order to configure and automate the storage tasks and to enhance retrieval facilities.

A combination of these elements will provide the necessary features of an automatic, unattended backup, archival and retrieval system.

Element One: The Mass Storage Solution

That the conventional and ubiquitous half-inch tape has had its day cannot be denied. Its data capacity is limited and the hardware is extremely bulky. Its unreliability and the costs of management and storage render it obsolete.

The quarter-inch cassette was an attempt to make magnetic tape more manageable, with a cheaper entry level. The disadvantage of relatively low storage capacity may be reduced by using a casette autochanger. However, it is more expensive than tape, has a relatively low data-transfer rate and lacks standardized format.

Helical scan magnetic tapes (8mm or 4mm) are usually known as DAT or DDS. These devices are very small and cheap, and they hold an astonishing amount of data. However, they are still hampered with the disadvantages of delicate media and mechanical contact between media and read-write heads. The drives themselves are proving to be unreliable too. The possible loss of gigabytes of data due to one failure (hardware or software) is not a pleasant thought. It is interesting to note that backup to this media is forbidden by some high security and legal bodies; small is not beautiful when somebody can walk out of an establishment with Gigabytes of data in a pocket.

The latest buzz-word in mass storage is "re-writable optical disc". Data changes require a simultaneous magnetic field and high temperature from a laser, avoiding any mechanical contact. This makes data far more secure than any of the previously mentioned devices which rely purely on magnetic fields to write. Manufacturers guarantee the media for ten years, while some experts estimate 100 years media life to be realistic.

Standard re-writable optical discs currently hold 325 Mb per side. This is clearly a capacity disadvantage when compared with helical scan tape - thus operatorless backups cannot be achieved. However, optical disc autochangers (jukeboxes) will hold enough discs to store tens or even hundreds of Gigabytes.

Currently, the only available combination of hardware device and media that satisfies all requirements is optical disc with a jukebox. Although the initial capital costs of the machine and the storage media are relatively high (albeit falling rapidly) a cost analysis will reveal that, on unattended and automated backup alone, the costs are justifiable if more than just one 1.3 Gigabyte DAT tape per day is otherwise required.

However, one must be able to control the jukebox and utilize the capacity of the storage media to its utmost potential. Many current tools cannot, for example, append stores to one side of an optical disc. Frequently, volume spanning cannot be achieved, where a store continues automatically on the next disc side when the previous side is full. This is clearly unsatisfactory when alternative technology exists.

The following facilities must be available to allow:

- Volume-spanning over as many disc sides as required.
- Append data to previously-written data on a disc side (no waste of expensive disc space).
- Configuration of numbers of disc sides as named groups, reserved for storage of (say) operating system types, individual machines or projects.
- Re-configuration of groups without physical handling of discs, and without having to transfer data.
- Automatic checks of the media to ensure that the data is going to the correct group.
- Performance of simultaneous storage and / or retrieval jobs using the multiple drives available in the jukebox (up to four).

Element Two: Software on each Node

The three basic functions that the backup software should achieve, can be classified as management, storage and retrieval functions.

(1) Management functions must control:

* Output of information pertaining to the job, ie. whether it is directed at run-time to a file, to a screen or to a printer.

* Setup and maintenance of optical disc groups in the jukebox.

* Storage options at run-time such as compression on or off, naming volumes, storing a raw disc partition, or waiting on a busy jukebox.

* Restore options at run-time such as target directory / account, restore permissions etc.

(2) Storage functions must include:

* Conventional backup to a device of any type (local, or remote to any other node - to the "backup engine", for example).

* Archival to a device of any type (backup with subsequent remove of the source data).

* Option for compression on or off.

* Transfer of backups on system disc to a device of any type.

(3) Restore functions must include:

- Automatic decompression (if necessary).

- Choice of automatic restore of most recent data, or page-through of all stored filesets matching the one required.

- Restore of individual files, as well as specific filesets, directories or raw disc partitions.

Element Three: Software on the "Backup Engine"

The previously mentioned software on each node only goes partway towards supplying the ideal solution. The backup manager would still have to access every machine in order to define the data to backup. What is required for large networks, or small networks with large numbers of backup jobs, is a centralized management system. This should have the following features:

- Software loaded onto the backup engine, typically the node to which the centralized mass-storage device is connected.

- Facility to control the backup software on all nodes in the network for automatic backup and retrieval without the need for localized intervention.

- A menu-driven database containing descriptions of the remote node jobs and the associated schedules, for automated backup of each node.

- Dynamic records of all the attributes of all backup jobs that have been performed, to be used for retrieval purposes.

- A logfile of errors and / or completion messages from all remote nodes.

- Direct archival of data that is no longer required to low-cost tertiary media such as DAT. This enables data to be copied before it is overwritten by data that is subsequently backed up to the same group. The facility to restore directly from the tertiary media to the source machine must be provided.

Comments and Conclusions

It is interesting to note that the demand for this particular type of strategy has been led by some of the larger engineering organizations (aerospace, automotive, and telecommunications in particular). They have previously used large centralized machines, but have come to recognize the latent potential of heterogeneous networked environments, and have migrated accordingly.

While nobody can claim that the large central machine with multiple users is obsolete, the largest movement in the computing market is most definitely in the direction of open systems. Many users in this area have developed their expertise in the personal computer environment, where the needs for standards and security are not so obvious. They have, however, recognized the potential of a heterogeneous networked environment, and have migrated accordingly.

The key is flexibility. Data must be backed up automatically, without operator intervention. The data must be stored on the most reliable media possible, but must be accessible and retrievable. The software solution must not limit the users' future choice of hardware - it must be forward compatible. It must also not limit the choice of backup strategy - whether daily fulls, weekly fulls and daily partials or weekly fulls and daily accumulative - all must be accomodated.

It is essential that everybody be aware of the possible dangers, hidden costs and administrative overheads of open systems, and then minimize them without compromizing data-security, while at the same time limiting capital expenditure. Careful analysis of your needs will prevent "potential" from turning into "pitfall".

PAPER NO.: 2004

TITLE: Changing The Rules for Data/Voice Integration

AUTHOR:

Richard Preece
Hi-Comp GmbH
Eichenlohweg 24
2000 Hamburg 60
Germany
+49406304011

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

Paper Number 2005

LAN Manager - A Case Study

Lisa M. Moose
Hoechst Celanese Corporation
Celco Plant
Route 460
Narrows, Virginia 24124
(703)921-1111

"Two terminals do not two interlocutors make."
-Jean Baudrillard

ORIENTATION

In this case study of a local area network, more than three mutations of LAN Manager were observed. From Hewlett-Packard's (HP) LAN Manager/X to Microsoft's (MS) LAN Manager 2.0 to a combination of HP LAN Manager/X and MS LAN Manager 2.1, and beyond, the human LAN Manager's efforts to achieve optimal network services were sincere. Yet, when the time came to interface and bind a horde of end-users with LAN Manager services, confusion, misunderstanding, and unfulfilled expectations were the result. This study probes the LAN Manager's challenge to maintain a high level of current network technology while hoisting up a low level of end-user interface capabilities to meet the balance of a "win-win" situation.

It is my intention to admit openly the temporary nature of the subject configuration information since successive versions of software releases rapidly outdate each other. Also, here at the very outset, it is essential to qualify the unorthodox writing style that will be employed to create a flexible prototype for the mapping of an ever changing LAN-scape. In the final section, ANALYSIS, this prototype composed of a network of seemingly disparate metaphors will result in a user-friendly *self-image* of a mechanical system that, by its very nature, cannot have this anthropomorphic capacity.

CASE STUDY

Prior to any network proposal, an initial business investment had been made consisting of stand-alone PC's, HP 3000's (both classic and MPE-XL) distributed terminal controllers, terminals, a time division multiplexor linking a 3174 cluster controller to a remote IBM host, and more. Therefore, the subject HP 9000 (HP-UX) and HP 486 PC (OS/2) servers, DOS PC client workstations, 802.3 10 base T workstation connection, X.25 interconnection solution, TCP/IP protocol suite, HP LAN Manager and MS LAN Manager 2.1 network operating systems and other product selections were based on the existing capital investment coupled with a stated network objective. The stated objective was to implement an open, high speed communications network that would support both current and future distributed applications.

The original network set-up utilized HP LAN Manager/X running on a HP 9000 (HP-UX) server with HP LAN cards installed on the PC (DOS) client workstations. This environment proved to be stable, but provided no LAN management functionality such as logon services, audit trails or other basic control devices. Hence, the introduction of MS LAN Manager 2.0 on the supplemental 486 PC OS/2 server. This other LAN Manager boasted the above mentioned control devices that were missing from HP LAN Manager/X, but the installed HP LAN cards were not a configurable option for MS LAN Manager 2.0. The integration solution was to install a dummy network card on MS LAN Manager 2.0 to modify portions of the PC client's config.sys and protocol.ini files.

In the config.sys file, the ethernet driver device was modified.

```
DEVICE=C:\LANMAN.DOS\DRIVERS\ETHERNET\HPLAN\HPLAN.DOS
```

In the protocol.ini file, the [HPLAN] and [TCPIP] sections were modified.

```
[HPLAN]
  DRIVERNAME = HPLAN$

[TCPIP]
  BINDINGS=HPLAN
```

Additionally, ELNKII network cards were purchased for the next group of PC client workstations. Since MS LAN Manager 2.0 is currently outdated, and in the interest of clarity, the rest of the user modified configuration parameters will reflect only the HP LAN Manager/X and MS LAN Manager 2.1 environment. Suffice it to say that the changes required for MS LAN Manager 2.0 are different from the unwritten requirements of MS LAN Manager 2.1.

The standard MS LAN Manager 2.1 installation does not set-up the sections in protocol.ini which enable Hewlett-Packard NS Services to run. Therefore, the following sections were appended to the protocol.ini file. (Note that xxxxx indicates information that is unique to each LAN.)

```
[PROBE]
  DRIVERNAME=PROBE$
  BINDINGS=TCPIP_XIF
  NSDOMAIN=xxxxx

[TCPGLOBAL]
  HOSTNAME=xxxxx
  NETFILES=xxxxx

[TICL]
  DRIVERNAME=TICL$
  BINDINGS=TCPIP_XIF
  ACBS=xxxxx
  CMDS=xxxxx
  VCS=2
  PORTS=xxxxx

[HP-VT]
  DRIVERNAME=VTCOM
  BINDINGS=TCPIP_XIF
```

In general, the required modifications to HP LAN Manager/X and MS LAN Manager 2.1 were minimal because these products have been carefully engineered -- in contrast to some end-user applications that are shared on this network. The HP 9000 (HP-UX) server structure, design and performance proved no less than superior, requiring no second-hand modifications.

In the HP LAN Manager/X /usr/lib/lm/lanmanx.ini file, we set umask=000 so that the default file permission constraints did not prohibit the PC e-mail application from functioning. Also, the maxtreeconns= parameter had to be increased from the default setting.

MS LAN Manager 2.1 required fewer modifications than MS LAN Manager 2.0. The script file creation in MS LAN Manager 2.1 is relatively straightforward and allows the administrator to direct the net uses to shares appropriate for each end-user or end-user group. For the sake of the network-illiterate end-user, it is recommended to have the net use /persistent:no parameter included in the appropriate script files. If this parameter defaults to yes, MS LAN Manager 2.1 attempts to establish connections at logon time for every share listed in the script file by prompting the end-user to say whether the connections should be re-established in current and then in future attempts. Although this may be programmatically logical, nearly all of the end-users in this situation would initially answer "yes", and then would suspect an error as the program continued to question each share for current then future attempts. Therefore the end-user would next answer "no" in an attempt to cancel the process.

The net result of these inconsistent responses by the end-users would be a PC lock-up, and a required reboot. Although these types of situations seem ridiculous, they are a major source of distressful confusion to the end-users and contribute negatively to the user community's image of the network.

One of the more intriguing problems presented involved upgrading the 286 PCs. When upgrading the HP ES 286 to a 386 model by replacing the motherboard, we found the default settings of the VGA card, the HP LAN card and the new motherboard were incompatible. Depending on the VGA version, some portions of memory had to be blocked and/or certain dip switches changed and/or IRQ location changed; and the settings of each PC were unique! Another problematic combination error occurred with a new 386 motherboard, the EGA card and the ELNKII network card. The solutions were surmised case-by-case on a time consuming trial-and-error basis.

The HP 386 Vectra PCs were network configured more readily than the 286 PCs. The 386 PC's config.sys files consistently required a block to memory for the area that the ELNKII network cards directly addressed. When the DEVICE=C:\QEMM\QEMM386.SYS RAM X=C800-CC00 statement was included in the config.sys files for HP 386 PC's using Quarterdeck Memory Manager, the memory map (Manifest) would display C800-CBFF 16K UNUSED! This exemplifies yet another case of a working function that appears, by read-out of external monitoring devices, dysfunctional.

This case study manufacturing location was subject to corporate subnet addressing constraints which required IP addressing above xxx.xxx.xxx.255. The MS LAN Manager 2.1 installation process does not allow IP addressing to exceed xxx.xxx.xxx.255. To work around this restriction, a dummy IP address was used at installation time, and the correct IP address was later entered in the protocol.ini file.

Another unusual failure point identified was the sequence in which the mouse driver was loaded onto the system. It was determined that the MS LAN Manager 2.1 client PC configuration required the mouse driver not be loaded in the config.sys file, but rather in the autoexec.bat file after the TCP/IP, network services and drivers load, but before the net logon command was issued. If this load sequence was not performed, then the following PC lock-up scenario would occur. When a user submitted a file to the network printer que, MS LAN Manager 2.1 would return a net pop-up message informing the end-user of the print job completion, and the screen would next turn completely white. The PC would totally lock-up as the net pop-up message disappeared. The end-user would have no choice but to cold boot the PC, and invariably lose the multiple page document that they had worked on for hours without saving.

As far as the TN3270 connectivity software is concerned, at the time this article was completed, the remote IBM host site was continuing to refuse to enable our IP address for testing. Therefore, the case study connection to remote IBM host continued via the IRMA or Attachmate connectivity cards, and the 3174 cluster controller set-up.

With the MS LAN Manager 2.1 workstation client, the IRMA card must be run non-resident or the PC will lock-up when returning from a remote IBM logon to a PC application. The Attachmate card behaves similarly, therefore the host window must be disabled before running a PC application. Also, a local VAX running Pathworks configured in a TCP/IP environment successfully connected to the HP 9000 in this case study network.

The HP 9000 network connection to the HP 3000 machines was a simple software solution (Reflection 7). For example, choosing from a menu on the core application triggered a batch file that loaded the following information and then called a Reflections command file (note again that xxxxx indicates information unique to each LAN):

```
ECHO OFF
NET USE I: \\xxxxxxxx\xxxxxxx
I:
CD \xxxxxxx
R7 xxxx.CFG O:\xxxxxx\xxxxxx.CMD
CD \
NET USE I: \delete
O:
```

The command file then would transmit the connection/termination information shown as follows.

```
QUIET COMMAND ON
DISPLAY "^[&k1C"
DISPLAY "Please enter your initials: "
ACCEPT V1 LIMIT 4
TRANSMIT "^M"
WAIT 00:00:05 FOR ">"
IF NOT FOUND
    GOTO ERRORCODE
ENDIF
TRANSMIT "CONNECT XL2^M"
WAIT 00:00:05 FOR "XL2#"
IF NOT FOUND
    GOTO ERRORCODE
ENDIF
```

```
DISPLAY "[&k0C"
TRANSMIT "HELLO $1,xxxxx.xxxx^M"
HOLD FOR "Disconnect"
EXIT
```

After the HP 3000 connection was closed, commands were issued to unload what was previously loaded in the Reflections initialization process as listed above.

The core application chosen by remote decision makers for this network was WordPerfect Office (for a combination of its e-mail, calendar and schedular capabilities). Since corporate offices were running this core application on Novelle's Netware product, it was yet another challenge to achieve e-mail connectivity. The working solution was to load a version of Portable Netware onto the HP 9000 server. Then all of the e-mail application directories were mapped identical in both Portable Netware and HP LAN Manager/X. Hence, all of the inbound e-mail was delivered by Portable Netware to the appropriate file location on the HP 9000 server, and served by HP LAN Manager/X to the client workstations. Outbound e-mail worked vice versa. All of this core e-mail application functioned readily from installation except for the administrative utilities. These e-mail administrative utilities required a PC client workstation that was configured as the HP client. When the utilities were run on the client MS LAN Manager 2.1, the programs would always abnormally terminate on the first run.

As with any purchased software package, this core application did not fulfill all of the end-users' requirements. Specifically, managers needed to enable their administrative assistants to have calendar file editing capabilities while not allowing the other end-users these permissions. The simple solution was to group this set of managers and administrative assistants at the HP-UX level and limit the full file permissions to the owner and group only (rwxrwx---). Control of the file permissions at this level successfully enhanced the application by meeting the end-users' demands.

As a rule, the packaged PC software applications loaded without a hitch from the DOS workstations to the HP 9000 server, with the exception of Lotus 123 network version 2.3. This software simply would not load direct to the HP 9000. The work-around for this conflict was to load the software onto the HP 486 PC (OS/2) server, and then xcopy the files to the HP 9000 server. The files were then functionally shared to the clients.

In summary, all of the purchased products functioned more or less as advertised. Some products, such as the HP 9000 server, performed at a higher standard than other pieces of the network puzzle. Yet the majority of the impasses could be resolved by parameter modifications and/or other variations determined on a trial-and-error basis.

ANALYSIS

A good LAN Manager who understands the business goals, network objectives, product constraints, and end-user perspectives should be able to effectively orchestrate a network that avoids the confusion and misunderstandings that this case study observed at the human level. Obviously there are cost-effective products far superior to the PC DOS client workstations, but many businesses must realistically deal with the PC investments they made in the late 1980's.

The greatest network integration challenges reside in the area of the end-users. It is far too easy to drown in the details of setting the multitude of parameters associated with the integration of the operating system, network operating system, application, to say nothing of the positioning of the hardware. The LAN Manager can become lost to the end-users who are islands of information that in order to contribute to long-term business success, need to upload data in a summarized format to key decision makers. These decision makers in turn must be able to reach down to the detail level when business demands. If users are not brought to an initial understanding of the network advantages, goals and objectives, they will rebel out of ignorance and continue to seclude valuable business information by storing data on the floppy diskettes they have learned to understand and protect. The time and effort required to build the users' network understanding, trust and participation is rarely, if ever, accounted for in the planning stages. Yet an astronomically high percentage of the support team's time is spent answering requests for help that could have been easily avoided if a little attention had initially been paid to the end-user needs!

Such oversights are very costly to the network both in terms of productivity and image. The image of a high-tech, solid network will quickly deteriorate if the majority of end-users experience frequent PC lock-ups, lose files, and share no understanding of the technological structure. It is a fierce, uphill battle to change end-users' first impression of the network. Also, end-users' spread their opinions with the speed and efficiency of a computer virus.

Once this case network was installed on the PCs of network-illiterate end-users, every PC problem, even a power strip failure, was blamed on the network! This is ridiculous! Mostly because the end-users mistakenly view network peripherals as central to the entire system.

The end-users are like a misguided map-reader who fails to realize the importance of north being the orienting direction connecting map and territory mapped. For a person to understand the directional concept of north and the subsequent methods of orientation, an educational process must have occurred providing the individual with a navigational image of the world. Without a similar base orientation to a network, the users will never be able to take the first steps away from their stand-alone islands of information into an integrated and strategically critical network path where business profit can be most effectively realized.

For end-users not only to use the network but also network to understand it, a non-technical image of the system should be envisioned by all parties. The end-users do not have to posses an intricate understanding of the network, but rather conceptual one. In fact, a finely grained detail schematic is more often than not confusing to those not involved with the architecture. This bewilderment can be best understood by using an analogy. Take the human anatomy which can be broken down into various sub-systems such as neural, cardiovascular, respiratory, skeletal, muscular,etc. Consider how these various workings of the body, though essential to the physician when diagnosing certain pathologies, are not readily apparent when a person is functioning in the social network.

The human body, with all its complexities is simplified by the individual who is comprised of those sub-systems into a self-image. One could even say that one's self-image is the biological network of life being conscious of itself.

Realizing that boards, cards, wires, disks, keyboards and monitors have to this date not resulted in a computational device that is remotely conscious, much less self-conscious, it becomes apparent that it is up to the facilitators of the network to project LAN managing capabilities in the form of a network *self-image* to the entire domain, if that LAN-scape is expected to be anything but feudal.

PAPER 2006

PORTABILITY NEEDS UNDER HP-UX

by P. Viallefont and C. Lamant

C.N.E.S

18, Avenue Edouard Belin

31055 TOULOUSE Cedex FRANCE

Tel: (33) 61.27.46.59.

INTRODUCTION

Portability under Unix is still far from being perfect. Indeed, while all of the Unix systems claim to comply with standards such as Posix, X/OPEN, OSF, Unix International, SVID, BSD, etc., it is sometimes very difficult to distinguish between standard and non-standard characteristics of a Unix system. The reality is that one may buy a Unix system which is compatible with almost all of the previous standards, use it to develop an application and then realize that it is only partially portable in other Unix environments.

Another problem is the multiplication of standards. The user might quite understandably be confused by the gamut of standards proposed by Unix systems. Which is the most important ? To what do they correspond ? How should one use them ? Which of them will ensure the most portability ?

The purpose of this paper is to share our experience as users in this area. First of all, we shall try to clear a way through the jungle of Unix standards. We shall then finish the paper by presenting our approach (in terms of portability) during the development phases, under HP-UX, of a computer based application for satellite control.

THE UNIX STANDARDS

The number of standards and normative bodies for Unix is so great that we shall restrict ourselves to the most important ones.

There are several types of normative and standards bodies to be considered : those which specify only the interface between the system and the applications (IEEE with Posix, X/OPEN with XPG3), those which propose only existing standard products (Berkely with BSD4.3, OSF with OSF/Motif) and those which propose both (Unix International with the SVID interface specification and the Unix System V product).

Posix

Posix (Portable Operating System Interface X) is undoubtedly the most important standard under Unix. Historically it was a group of American users (the "/usr/

group") since renamed Uniforum, which began the work of standardizing Unix. From 1984 on, the "/usr/group" published a document specifying the interfaces between a portable Unix system and software applications : the "/usr/group/Standard". In 1985, the IEEE resumed this work and created several Posix working groups. Each group, consisting of technical experts, was to propose a standard document. When this document was sufficiently developed, it was submitted to the ISO JTC1 WG15 group which circulated it as a Draft Proposal (DP). From then on the document, edited in its basic form by the IEEE group, followed the conventional ISO circuit before becoming an International Standard (IS).

The ISO Posix standard consists of three parts :

- IS 9945 -1 for system calls,
- IS 9945-2 which focusses on commands and utilities,
- IS 9945-3 for system administration.

The following are the main IEEE Posix working groups :

Posix 1003.1: This group began in 1985 by continuing the /usr/group's work. Posix.1 is mainly concerned with standardizing interfaces for the Unix operating system. For an applications programmer, Posix.1 corresponds to the complete description of Unix system calls (chapter 2 of the reference manual). A first version of the 1003.1 standard came out in 1988 : the IEEE Standard 1003.1-1988. However the final document is the IEEE Standard 1003.1-1990 which corresponds to the ISO IS 9945-1:1990 standard. This group is actually the most advanced since it is the only one to have achieved an ISO standard. The document describing the standard is written with a C-language interface. It cannot really be considered to be a programming manual.

Posix 1003.2 : This group's work is pending at the present time and hopefully will lead to an ISO standard towards the end of 1992 (ISO IS 9945-2). The group is working on shells and utilities. P1003.2 is only concerned with the characteristics necessary for the portability of shell-scripts and management of background jobs, and not with the user interface (history, command management, etc.) Their objective is to make programmes portable. However an extension to the standard is currently being developed : Posix 1003.2 UPE. This extension includes improvements and extra commands to provide a standard user interface.

Posix 1003.3 : This group's objective is to produce a standard methodology for testing conformance in the implementation of Posix standards.

Posix 1003.4 : This group is busy developing a set of system interfaces covering real-time aspects. These works are due to be finalized at the beginning of 1993. The results will be included in the Posix.1 :ISO IS 9945-1 standard.

Posix 1003.5 : At the present time, the Posix.1 standard is published with a C-language interface. This group's work consists of defining an Ada language binding to the basic system services of Posix 1003.1.

Posix 1003.6 : This group deals with security aspects under Unix such as access control lists, file labels, user privileges, audits, etc. Its results will be incorporated in the Posix.1 ISO IS 9945-1 standard.

Posix 1003.7 : This group is mainly concerned with system administration. This committee's objectives are to provide an object-oriented interface and network inter-operability. Its results will be the basis for the Posix.1 ISO IS 9945-3 standard.

Posix 1003.8 : This group is studying network aspects such as distributed file systems, network programming interfaces, remote procedure calls and OSI protocol programming interfaces. This group is fairly recent and its results are not expected before 1993.

Posix 1003.9 : Like the P1003.5 group, this group's goal is to rewrite the Posix.1 standard in Fortran.

Posix 1003.10 : This group deals with features which are specific to super computers. Functions such as batch management, restart points, scheduling, efficient input-outputs, etc.

Posix 1003.11 : This group deals with all of the transactional aspects.

IEEE		**ISO**
P1003.1	1990 →	IS 9945-1
P1003.2	1992 ? →	IS 9945-2
P1003.3		
P1003.4	1993 ? →	IS 9945-1
P1003.5		
P1003.6	? →	IS 9945-1
P1003.7	? →	IS 9945-3
P1003.8	? →	IS 9945-1
P1003.9		
P1003.10		
P1003.11		
...		

Fig 1. The IEEE Working Groups and Posix standards

Other IEEE groups have started up, as for instance the P1201 which is tackling the standardization problems for graphic interfaces.

At the present time, the great majority of Unix systems conform to Posix.1 (1988 or 1990). Certain manufacturers follow the works of IEEE groups (Posix 1003.2 and Posix 1003.4) by complying with the most recent drafts. Certain system owners

(VMS, MVS, etc.) are starting to comply with Posix standards.

X/OPEN

X/OPEN is a group of manufacturers which was created in 1984 on the initiative of European manufacturers. Since then the consortium has expanded to include non-European partners and in 1990 consisted of about twenty manufacturers.

X/OPEN's goal is to propose a common applications environment (CAE) using the main market norms and standards. X/OPEN's approach is more an attempt to define models for norms and standards (thus providing a general solution) than to develop the standards themselves. X/OPEN was largely inspired by SVID and POSIX for system calls, command and utilities aspects and C-language.

Thus X/OPEN suggests solutions for system interfaces but also in terms of programming languages, data management (ISAM, SQL), networks or windowing environments (X11).

The X/OPEN specifications are available in one document : the XPG3 (X/OPEN Portability Guide Issue 3) which itself consists of seven volumes :

1. Commands and utilities.
2. System interfaces and header files.
3. Supplementary definitions.
4. Programming languages (Cobol, Fortran,Pascal,C,ADA).
5. Data management (SQL,ISAM).
6. Window management.
7. Network services.

Many Unix systems claim to conform to X/OPEN. However there are several levels of conformance with X/OPEN :

- "X/OPEN Base" which includes system calls, libraries, commands, utilities, the C-language and the internationalizing functions.

- "X/OPEN Component" which includes "X/OPEN Base" with at least one of the following extensions : Cobol, Pascal, Fortran, ISAM, SQL, Terminal Interfaces, Window management (X11R3), XTI (network transport interface) and connections to PCs.

- "X/OPEN Plus" which includes "X/OPEN Base" and all of the "X/OPEN Component" extensions.

The XPG3 guide bears a great resemblance to a Unix reference manual. For strict conformance with XPG3, the programmer is advised to use the X/OPEN guide rather than the reference manual for the host operating system.

The XPG3 guide was written in 1989 and consequently is already a little obsolete. It does not altogether conform to certain more recent standards such as Posix and

ANSI-C . In addition, XPG3 only uses X11 R3 for window management and has not chosen between OSF/Motif and OPEN/LOOK. The next version of the XPG4 portability guide should fill these gaps.

At the present time most of the Unix platforms can carry "XPG3 base" except for SUN which conforms only to the old version of XPG, namely XPG2.

SVID

SVID describes the interface for AT&T's Unix system. It was one of the first Unix interface specifications to have come out (1985). SVID was largely inspired by the "/usr/group/Standard".

BSD

BSD means "Berkeley Software Distribution". Since 1970 the university of Berkeley has been developing Unix versions of which the most recent is BSD 4.3. There is no specification document describing the BSD system interface.

OSF and Unix International

The Open Software Foundation is a group of manufacturers which was created in 1988 in reaction to the American telecommunications giant AT&T's acquisition of shares in the SUN microsystems company. Indeed, as AT&T was owner of the Unix System V licence and as SUN was the undisputed leader of workstations, there was a risk of a monopoly in the Unix market. The other manufacturers (DEC, HP, IBM, etc), uneasy because of this alliance, then decided to participate in the OSF. Since its creation, the OSF has launched five RFTs (Request For Technology) :

- OSF/Motif for window management,
- OSF/1 for the operating system,
- OSF/ANDF for the distribution of portable software,
- OSF/DCE for distributed environments,
- OSF/DME for the consistent control and management of distributed systems.

At the present time, only OSF/Motif has been recognized as a standard. OSF/DCE appears to have a bright future as OSF/DME. OSF/1 has not yet been recognized whereas OSF/ANDF seem to be behind schedule and are only due to be published in 1933.

Unix International (UI) is an organization of manufacturers which was created in December 1988 to promote AT&T's Unix System V system and to control its evolution. This group was created in reaction to the creation of OSF. Its main members are AT&T, SUN, ICL, Control Data, Amdahl, etc.

Unix International's objectives are similar to those of the OSF with the following exception. Unix International has centered all of its works around its operating system, Unix System V Release 4. The OSF approach therefore seems to be more open as all of its products are not dedicated to one single operating system but are designed to be portable on any Unix and non-Unix platform.

In fact, the OSF, just like Unix International has accelerated a de-facto standardization process by promoting its products among its members while at the same time taking care to offer products which are independent of any particular manufacturer and which are technologically advanced.

To conclude, the most important standard under Unix is the Posix norm since it has the international recognition of all governments and is respected by both Unix and non-Unix manufacturers. Unfortunately this standard is still not complete. While it does take system calls into account, it still lacks shell and utilities aspects as well as security and management, real time and network aspects. While waiting for the Posix standards to mature, we advise using the X/OPEN group's standard : XPG3.

HP-UX AND STANDARDS

The HP-UX system is based on the AT&T System V, version 3.0. It conforms to the main standards in the Unix world which are :

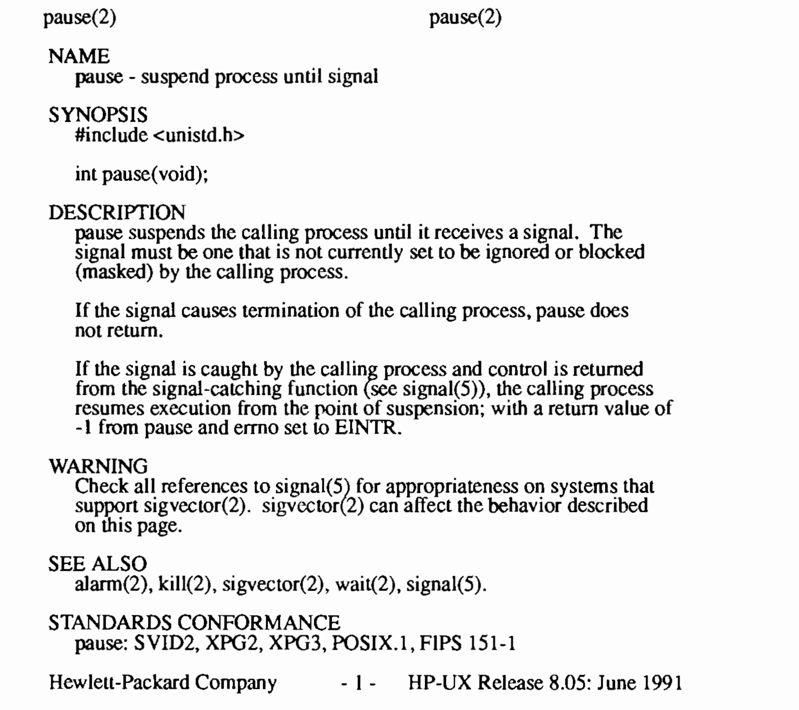
- Posix 1003.1 - 1990 for system calls,
- Posix 1003.2 draft 09 for shells and utilities,
- XPG2 and XPG3 base for system calls, commands and utilities, internationalization and C-language,
- SVID2 for user and administrator commands, system calls, C-language and libraries,
- FIPS 151-1,
- the ANSI C language and its standard libraries,
- X11R4 and OSF/Motif for window management,
- NFS in the 3.2 version.

In addition, HP-UX offers BSD and owner extensions (real time, security, etc.).

HP is one of the founder members of the OSF. The HP-UX system should in the future be capable of taking the different OSF technologies such as OSF/DCE, OSF/1, etc.

The HP-UX system is an open system which respects the main standards of the Unix world. It may however be criticized for the following reasons. First of all, the NSF version supported is relatively obsolete (3.2 in relation to current versions which are in 4.0). Also, HP-UX has not yet followed the P1003.4 drafts of work on real time aspects unlike certain other Unix systems.

Even though the HP-UX system conforms to the main current standards of open systems, this is no guarantee that an application developed under this system will be open to other systems. One of the great advantages of the HP-UX system is that it proposes, in its reference manual, a section titled "Standard Conformance". This section is of great help to a developer concerned with portability. Indeed, an application programmer can thus construct his application by choosing calls and standard commands while avoiding the pitfall of using features owned by the operating system. This does not mean that one should not use owner extensions which are sometimes very handy, but that one should remember that they do not conform to the standards. The HP-UX manual clearly indicates when a command conforms to XPG3. It may be that this command has non-standard options. In this event the programmer may feel betrayed as he assumed he was using a standard call. For example, the "ls" command conforms to XPG3, but the "-1" option for this command, used to display output in one column, is not included in the XPG3 guide.



```
pause(2)                                              pause(2)

 NAME
    pause - suspend process until signal

 SYNOPSIS
    #include <unistd.h>

    int pause(void);

 DESCRIPTION
    pause suspends the calling process until it receives a signal.  The
    signal must be one that is not currently set to be ignored or blocked
    (masked) by the calling process.

    If the signal causes termination of the calling process, pause does
    not return.

    If the signal is caught by the calling process and control is returned
    from the signal-catching function (see signal(5)), the calling process
    resumes execution from the point of suspension; with a return value of
    -1 from pause and errno set to EINTR.

 WARNING
    Check all references to signal(5) for appropriateness on systems that
    support sigvector(2).  sigvector(2) can affect the behavior described
    on this page.

 SEE ALSO
    alarm(2), kill(2), sigvector(2), wait(2), signal(5).

 STANDARDS CONFORMANCE
    pause: SVID2, XPG2, XPG3, POSIX.1, FIPS 151-1

 Hewlett-Packard Company          - 1 -     HP-UX Release 8.05: June 1991
```

Fig. 2: The “Standards Conformance” section in the reference manual

The "Standards Conformance" section in the HP-UX manual recognizes the following standards :

- Posix.1 and Posix.2,
- ANSI C,

- SVID2,
- XPG2 and XPG3,
- FIPS 151-1.

This section could be criticized for not mentioning other standards like X11/Motif or the BSD extensions.

We remind you that this manual is divided into several chapters :

1 - User commands,

1M - System administration command,

2 - System calls,

3 - Function libraries,

4 - File formats,

5 - Miscellaneous,

6 - Games,

7 - Special device files,

9 - Glossary.

A software application developed in C-language under Unix generally uses source programmes and shell-scripts. Hence the programmer uses the three first chapters of the reference manual for the programming.

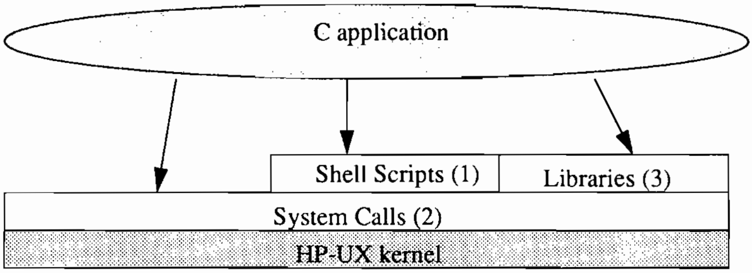
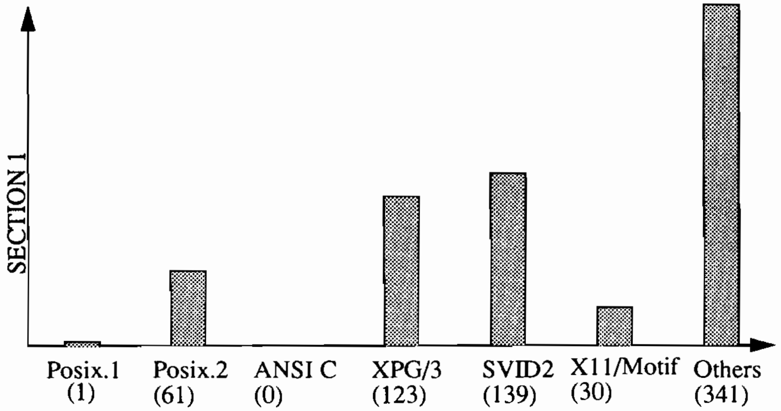


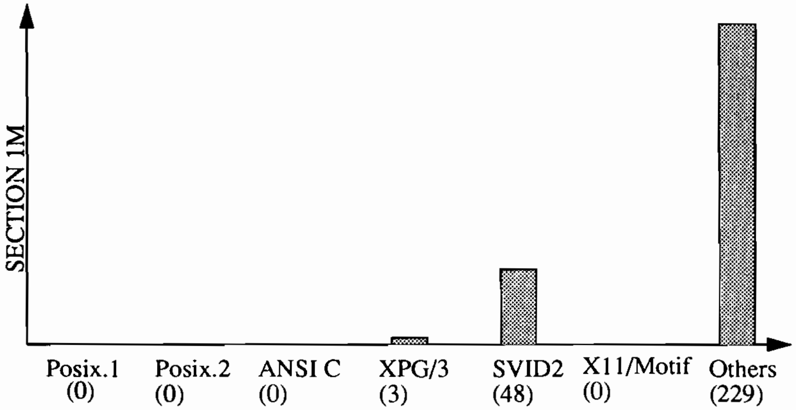
Fig 3: Interfacing a C application and a Unix System

We found it interesting to measure quantitatively the number and rate of standard and non-standard calls contained in the first three chapters of the HP-UX manual.



Total number of calls= 513

Fig 4: Statistics of the chapter 1



Total number of calls= 277

Fig 5: Statistics of the chapter 1M

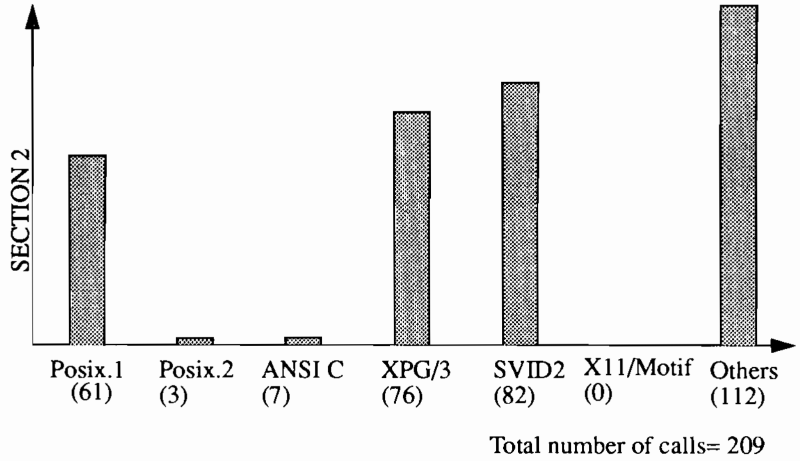


Fig 6: Statistics of the chapter 2

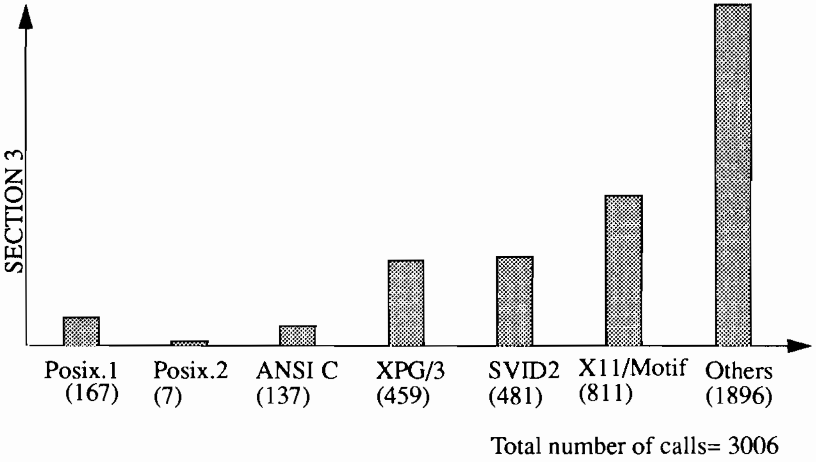


Fig 7: Statistics of the chapter 3

The results show that the Posix standards (1 and 2) only cover a small proportion of the HP-UX system commands and calls. Indeed the Posix.1 standard only covers

29% of the system calls.

Posix.2 only covers 11% of the manual commands. Allowances should be made for the figures obtained for Posix.2 since at the present time the standard does not take the interactive commands into account (vi, mail, etc.).

Outlines like SVID2 and XPG3 are of greater importance in the HP reference manual.

There is a large proportion of X11/Motif library functions (26%) present in chapter 3.

Finally, the non-standard calls included in the HP manual account for a large proportion. 66% for the commands and utilities, 82% for the administrator commands, 53% for the system calls and 63% for the libraries. Allowances should also be made for these figures by noting that the counting of non-standard calls includes the owner aspects of the HP-UX system (real time, etc.) but also the BSD extensions (TCP/IP sockets, r-commands), or the de-facto standards (RPC and XDR for SUN).

Several calls or commands in the reference manual conform to several standards. This overlapping is shown in the following figure.

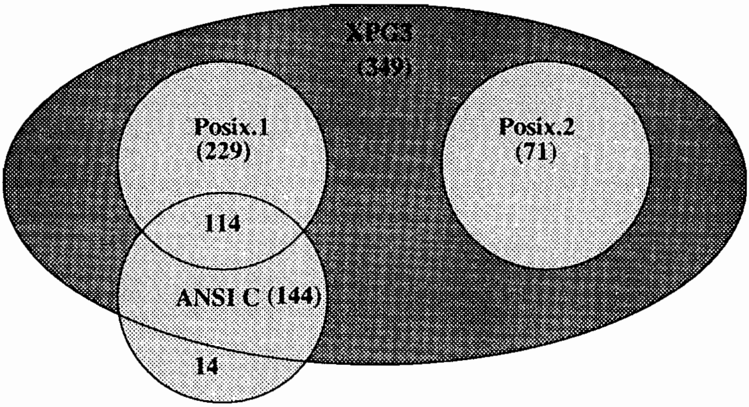


Fig.8: Overlapping of the main standards

Thus we see that the Posix.1 and Posix.2 standards share the same interfaces with XPG3. Posix.1 and ANSI C also share a large number of library functions. A certain number (14) of ANSI C library calls are not included in the XPG3 standard. Generally speaking one observes significant overlapping of the previous main standards.

PORTABILITY ANALYSIS TOOL

The SPOT4 project

The SPOT4 satellite is a French earth observation satellite. The architecture chosen for the control center of this satellite is based on HP9000 running HP-UX system.

A software layer, the Kernel Service Layer (KSL) was developed in order to facilitate and harmonize the various software developments. This layer is found between the HP-UX system and specific software development. The purpose of this layer is to offer high level (files, communications, etc.) shared services (libraries and tools)- to developers of the SPOT4 control center.

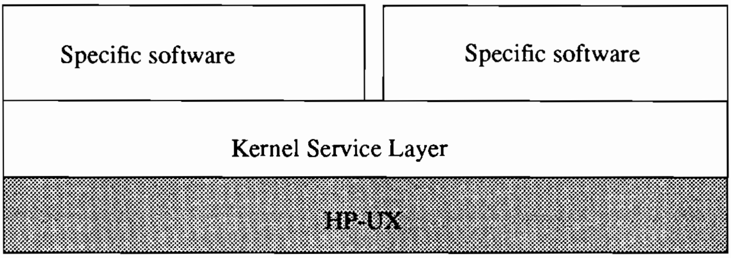


Fig. 9: Software structure of the SPOT4 Control Center

The development of this layer immediately required developments that were standard and portable. Indeed, the lifetime of such a project is rather great (of the order of 15 years). This durability constraint required that the software application be developed in a standard way which would make it possible to avoid having to modify the application source code every time there was a new version of HP-UX (suppression of certain owner features). Another aspect of portability for the KSL is the possibility of re-using this software on another Unix platform.

No portability constraint was introduced when the development began. However it seemed to us worthwhile to develop a tool for analyzing the system interfaces used by the KSL in order to identify the critical code parts in terms of portability and durability.

The systems call analysis tool

This tool is not a portability analyzer in the sense that it does not check whether the interfaces used include the good options or the good arguments, etc. The design approach was kept as simple as possible in order to avoid developing a tool which would be too complex to use and maintain. The tool is limited to identifying those interfaces of the operating system (chapters 1 to 3 of the reference manual) which are used in the programme sources written in C-language and in shell-scripts.

This tool is based on the "Standards Conformance" section in the HP-UX reference manual. It operates in two stages. The first consists of extracting, from the three first chapters of the online reference manual, the list of all the calls and their conformance to standards.

The list of interfaces was created under version 8.0 of the HP-UX system. It has the following form :

```
Xserver 1
adb 1
adjust 1
admin 1 SVID2, XPG2, XPG3
ar 1 SVID2, XPG2, XPG3
as 1 SVID2, XPG2
as_800 1
astrn 1
at 1 SVID2, XPG2, XPG3
awk 1 SVID2, XPG2, XPG3
banner 1 SVID2, XPG2, XPG3
etc.....
```

Fig 10: List of the interfaces in the HP-UX reference manual

Each line relates to a command, system call or library function. For each line the interface name comes first, then the number of the chapter in the reference manual and the list of related standards.

The list thus drawn up contains 4005 lines, in other words, as many commands, system calls and library functions.

As far as the choice of standards was concerned, we concentrated on POSIX.1, ANSI-C, XPG3 and SVID2. We omitted FIPS 151-1 and XPG2.

The second part of the tool analyzes (using the list previously drawn up) all of the system calls, utilities and library functions used in the source code of our software applications written in C-language and in shell-scripts. The makefiles and Fortran files were not analyzed. The analyzer was written in shell and uses the Unix utility **nawk**. It is able to work on a tree structure of source files.

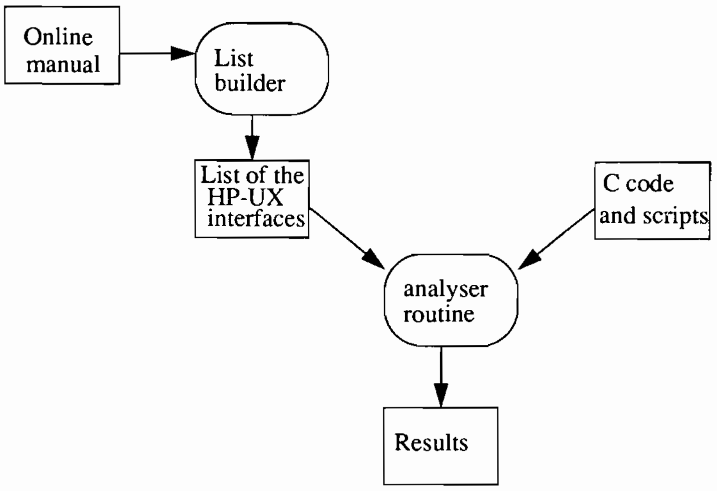


Fig. 11: Tool for analyzing HP-UX interfaces

The results are contained in three files :

- global : for each source file analyzed, the names of HP-UX interfaces and the corresponding lines of code are displayed,

- summary : this file uses the same information as the global file to filter the lines of code,

- skeleton : the HP-UX interfaces are listed for each application analyzed without specifying the names of files and the lines of code.

The skeleton level reveals the interfaces used by the application analyzed. The summary level is used to determine the interfaces used in each application file. The global level contains the most information. It is used to list the summary level information with in addition the lines of code where the interfaces appear.

By analyzing the results of the global level, we confirmed that the utility does provide good results. The most frequent errors concerned commands such as **du** or **l** (in **ls**) which were found in the character strings of the **echo** commands.

Manual analysis enabled us to detect certain commands called by the **system** and **popen** calls in the C sources.

Generally speaking, the results are 95% reliable. The remaining 5% require manual verification of the global level results.

CONCLUSION

Standardization of the Unix system has not yet reached maturity. Even when the Posix standards will have covered all aspects of the operating system, their conformance with a particular Unix operating system will not guarantee that an application developed under this system will be portable in other environments which conform to Posix standards.

This article has tried to pose the problem of portability within the framework of a space project with strong durabililty constraints, thus requiring that standards be respected. First we believe that it is imperative to write realization specifications which take portability into account. These specifications shall specify, in order of importance, the use of the standard ANSI C for the language, then of the standards Posix.1 for the system calls, Posix.2 for the shells and utilities and finally XPG3 for other complementary interfaces. Thanks to the "Standards Conformance" section in the HP-UX manual, these specifications are now easily applicable by the developer. However the specification alone was not enough. It was necessary to develop a tool making it possible to check that the interfaces used by the SPOT4 application did in fact conform to the portability standards. This tool thus enabled us to isolate the critical parts of the non-standard code and hence to measure the degree of dependence of the SPOT4 application on the owner features of the HP-UX system. This helped us to evaluate portability costs of software applications.

PAPER # 2007

Migrating to "OPEN"

John M. McCabe

Lever Brothers Company
818 Sylvan Avenue
Englewood Cliffs NJ 07632
201-894-6567

Introduction

In 1985 Lever Brothers developed and deployed manufacturing supervisory systems into its plants. These systems were developed for the HP-1000 A-Series hardware platform using the RTE-A operating system, IMAGE data base management system, FORTRAN programming language, 3rd party forms and report writer packages. These systems were successful as core packages that could be customized to fit the various needs of different manufacturing departments.

As the users became more sophisticated in their use of computers, the ability to satisfy user requirements with the existing hardware/software environment became increasingly more difficult. The implementation time and cost to have more advanced features added to the system became prohibitive using our "standard" environment. Application tool kits were on the market along with relational data base management systems but they mostly ran on UNIX systems. Most recently the price/performance offered by RISC architectures made moving away from the HP-1000 even more attractive.

Certainly Lever Brothers would move to the HP9000 product line and HP-UX but several key issues needed to be resolved:

1) Availability of all required interfaces
2) Training or retraining of staff
3) Impact on ongoing projects
4) First project selection

In 1991 all aspects of migration came together to permit a switch of environments from HP-1000 RTE-A to HP9000 HP-UX.

Background

In order to better understand the systems that are discussed, a brief explanation of the manufacturing environment shall be given.

Each plant has a host computer for its business/commercial data processing functions. This is an HP3000 system. A plant may have one or more production facilities on site. Generally the production facility is divided into a process and a packaging area. Process areas have factory floor controllers such as Distributed Control Systems (DCS) and Programmable Logic Controllers (PLC). Packaging areas have mainly PLC systems for equipment control. These systems are connected via various networks which are proprietary in nature. The in-between layer or supervisory systems being discussed in this paper are mini computer systems that support the process and packaging areas. These systems track consumption of raws and product produced as well as providing data historian and trending capabilities.

Migration

Process Systems:

The functionality previously mentioned as consumption of raw materials can be categorized into two main areas; BATCH and CONTINUOUS. In the batch environment ingredients are measured and combined in process vessels in a discrete fashion under the control of the DCS. A batch is considered complete when it moves out of the one process vessel into the next stage either packaging of another process being either continuous or batch. A continuous process involves a constant flow of ingredients that are simultaneously being added or mixed without the need for a holding vessel.

In the HP1000 environment these data collections were handled by Fortran programs that extracted data from the DCS and then populated records in detail data sets within IMAGE. Interaction with the data base was provided by Fortran programs combined with a forms package and a report writer. The main areas that needed improvement were in the use of IMAGE with report writers and Fortran based Forms applications. It was understood that the integration opportunities with available relational data base management were far superior to what we were using. A direct port was possible, using tools such as Allbase and Port - RX, but it was felt that this would be less than satisfactory. Therefore by moving to a new environment, using application tools with a relational data base, the goals would be more closely met.

The other functionality mentioned was of a data historian capability. The DCS supplier had provided a satisfactory solution on the HP1000. Over time it became unsupported and the suppliers next offering only permitted the use of another proprietary hardware/software platform. Another solution was needed that could extract data from the DCS, provide data historian functions and provide links into relational data base systems. An opportunity arose when an installed HP1000 system required additional work due to a process area expansion. A decision was made to port the system to HP-UX rather than reinvest in the HP1000 system.

Packaging Systems:

The packaging systems collected and reported on material consumptions, finished product production, line downtime reporting and limited views into line status. Direct line control provided to the operator was done through industrial displays hooked directly to the PLC.

The HP1000 applications were again done in Fortran with data storage into IMAGE. Custom Fortran with Forms and report writers provided terminal based interaction with historical data and limited real time information. A project was proposed to provide a packaging system for an entire packing hall. The specifications from the users required the use of a relational data base and functionality beyond what we had previously provided. Quality data collection, consistency between operator interface displays and supervisory displays and a more integrated system overall were desired. It was decided that this application would be based on an HP9000 system.

Timing Considerations:

Many opportunities came together to make the migration of these systems a reality. Projects in both areas were approved within the same year. Therefore a group synergy was developed in which experiences were shared. Training for systems engineers in HP-UX and relational data base technology could occur in a short period of time. Additionally, the DCS supplier finally provided a product that would allow an interface between the HP9000 series 800 system and their proprietary data highway. The major decision was to select tool kits and data historian packages to satisfy the requirements. Through this selection process, the relational data base was not considered an issue. The major suppliers under consideration all provided links to any of the relational systems under consideration. For these two applications Ingres was used. Use of other relational data base systems is being considered.

HP-UX Process System: BATCH

In migrating to UNIX from RTE, three main challenges existed: 1) a new operating system, 2) a new data base management system and 3) potentially a new programming language. It was decided that since the custom code that needed to be ported was in Fortran, and that Fortran existed for HP-UX and could be used to connect to the relational data base and to the DCS interface that these modules would remain intact and not ported to C. This was a good decision in that the module names and functions all remained intact and therefore a generalized understanding of how the system worked was maintained.

The biggest problem in porting existing code was encountered in instances where call by value structures were used. These needed to be changed wherever a reference or call was made to an external routine. A call by value made within our own Fortran code was still valid and did not need to be converted. The second major area of difficulty was in the include file structure used within the UNIX development environment. Most everything for application development is setup for C. Although UNIX supports Fortran and other vendors support programming links to their software from Fortran, the C language is better supported in most ways, ie documentation, manuals, on-line help, MAN pages, and phone-in support. This was not insurmountable but it did add additional time to the development process. So for the back-end, data collection and data base population, Fortran routines were migrated successfully over to the UNIX environment.

The front end, or user view of the system for configuration and reporting purposes that was originally done in Fortran was not converted in Fortran. Rather, this part of the application was redone in it's entirety using the Application by Forms 4GL that was part of the relational data base package. By doing this, we were able to move away from custom code, and use a better prototyping environment. The net result was an improved application that was easier to modify and took less time to develop.

RTE system calls were ported to UNIX system calls mainly using FORK and EXEC functions. We did find that it was better to remove some levels of multi tasking in UNIX. Things done in RTE to co-ordinate resources could not be done by setting up a program as a system utility in UNIX. This was accommodated by simply moving the function into the main code and allowing the data base to co-ordinate the resource. Probably the biggest advantage in simplifying the application by moving to UNIX was the CRON function and the Crontab File. All time scheduling methods used in our applications were moved in to the CRON function. Again this eliminated the need for custom code to execute different programs at different times.

HP-UX Process System: CONTINUOUS/DATA HISTORIAN

The main objectives for this area were: 1) support of both PLC and DCS environments, 2) long term application support, 3) support of relational data base management systems, and 4) moving away from Fortran custom code. The installed systems all lacked in these areas. The software applications that we looked at could provide us these advantages if we would use a UNIX platform. As an HP9000 800 was being provided for the BATCH system discussed above, the path was cleared to select which software would be purchased. Based upon previous experiences with their products and that their product was selected for the line monitoring system discussed in the next section, ISI's CIM/21 product was selected for continuous data collection and Data Historian requirements. Data from the DCS is collected into a real time data base. From here it can be shown on real time trends or process mimics, archived for future retrieval or processed and sent to a relational data base. Real time trends and process mimics can replay archived data for quality control analysis. Management reports are generated from the relational data base system either by using CRON or forms applications within the relational data base product. A feature of the CIM/21 product that made our migration easier was that on UNIX, ISI supports both the X environment as well as the older graphics terminal environment. Thus in moving from the HP1000 environment, we were able to reuse our investment in Intel 286 PC's that were running graphics terminal emulators.

HP-UX Packaging Systems

Besides those items already mentioned above about relational data base management systems and custom code, the major item for packaging systems was consistency between the views presented to each user. The factory floor devices were of the METRA/NEMATRON variety while either X terminal, graphical terminals or dumb terminals running forms applications were used at the supervisory level.

We found that there existed two types of systems on the market that provided the functionality we required. First were Intel X86 PC systems running on DOS with an occasional SCO-UNIX variety. Second were minicomputer or work station UNIX systems. Our architecture called for a minicomputer at the supervisory level and an Intel X86 system for the plant floor. Some software suppliers mentioned above were moving in a direction which was favorable for us. Suppliers of PC based systems were beginning to provide UNIX systems and suppliers of UNIX based systems were beginning to provide PC systems. Starting with half a dozen potential suppliers and narrowing it down to two, a final selection was done after detail specifications, RFP's and vendor presentations were done. The CIM/21 product from ISI was selected. Together with the hardware architecture, we were able to provide the same look and feel on all displays hooked into the system. The CIM/21 package was run on an Intel 486 system for the factory floor and supported graphics on an IBM 7554 19" monitor. At the supervisory level, X terminals supported the same graphics displays. As the users did not want a mouse on the factory floor, navigation at that level is via function keys on a limited function keyboard while the supervisory users are able to use a mouse or trackball with their X terminals. A 10 Base T ethernet was used to network the systems together. The line operators can control the lines, view line status and enter quality data through their stations. The factory floor system is also available to maintenance personnel for a detailed view of equipment status. Production data is automatically collected as are equipment downtimes and efficiencies. Production data is fed into an Ingres data base and reports are automatically generated or can be produced on demand.

Lesson Learned:

Through all this effort of porting systems, system specifications and competitive bidding, what has been learned? Several rules of thumb have been reinforced. Other intuitive beliefs have been confirmed.

Rule # 1: Do not buy vaporware. Over the time span from when Lever first wanted to move to the HP-UX environment there have been several instances when vendors wanted us to purchase serial number one of the product under consideration. When viable alternatives exist, it has always proved to be the prudent thing to resist purchasing products that do not exist. The link between the HP-UX system and the Fisher DCS is one such example. If we had decided too early to switch to HP-UX without this interface, the success of the project would have been questionable. In a similar vein, always ask for product demos. If the supplier cannot demonstrate a product on the desired platform, then it is unlikely that you will be able to take delivery of even a Beta version several months down the road.

Rule # 2: Satisfy user requirements. Open Systems, client/server, relational data base management systems. These are all nice technical ideas, concepts and in some cases products. But all of these put together will not sell a user. Meeting the needs of the users will sell a user on a technology. The job of the systems integrator is to use the appropriate technology to satisfy the user requirements. In several instances during our selection process, we found the technology being sold as the primary feature with little regard for the user needs. Needless to say, those suppliers did not in the end get our business.

Belief # 1: Relational Data Base Management System better satisfy user requirements than traditional Data Base Systems. Porting to HP-UX and a relational system was a major step. It has proven to have been the correct thing to do. The system is more powerful and more flexible especially in meeting the ongoing needs of the users. Changes and additions to the system that have taken place have been more manageable. In our first installed application we were able to use dynamic as well as standard SQL in the application. The users are pleased with the Ad-hoc reporting capabilities they have using standard forms and tables. The performance of the system has also been very satisfying. In the original system, with all of RTE-A's real time and multitasking capabilities, we notice performance problems with IMAGE. These problems; locking on deletes, serial reads of entire data set for certain reporting caused noticeable time delays in the old system. Although not engineered to remove these difficulties, the new relational systems do not have these problems. Developers now complain when a complete scan of a table lasts seconds. With IMAGE these same scans of a detail data set would have taken minutes.

Belief # 2: UNIX is a better environment than RTE-A. RTE does not have the relational data base management systems which are viewed as required today. Moreover, software suppliers are producing new products for the UNIX market at an incredible rate which yields strong competition and therefore competitive pricing. Additionally, HP-UX is provided on the HP-PA RISC architecture which is in a market segment that is fiercely competitive. By combining all of these aspects, a substantial price/performance increase has been realized by migrating to the HP-UX environment.

The one aspect that was certainly a question with the move to UNIX was performance. UNIX is a non-deterministic system. Would the performance provided by UNIX prove satisfactory for our applications? What is real-time and how important is real-time for the success of the application? HP-UX provides scheduling to a one second level although it is still non-deterministic. Our experiences have shown that the performance available today on the HP-UX/HP-PA architecture is responsive enough to handle applications that were previously believed to require strict real time capabilities.

Next Steps:

With two successful UNIX systems installed, what will the next steps be for Lever in our open systems plans. Initially we will add Oracle to out Relational Data Base capabilities. Additionally we will be updating our programs to work with the newer DCS systems from Fisher Controls (PROVUE consoles).

Looking forward, we see other new systems that would become part of our CIM architecture such as Quality or Warehouse Control being developed on a UNIX platform. These and other plantwide systems will begin to take advantage of the client-server model and the Distributed Computing Environment definition being developed by OSF.

Eventually, with the definition of SP-88 and the Flexible Batch Concept, we see the need to move to standardized batch data collection systems. We envision that this will be provided through the independent software supplier chain. In this way, we hope to realize the benefits of off the shelf products that we enjoy today with our RDBMS's and CIM/21 packages, namely well supported quality products.

Conclusions:

Through the efforts of migrating from the RTE-A environment to the HP-UX environment, Lever Brothers now enjoys the benefits of the improved price/performance provided by RISC, UNIX and Relational data base management systems. We have greatly reduced 3GL custom code, delivered systems with better integration, our applications are more robust and easier to modify. Most importantly, we had satisfied the user requirements and in some cases I believe we have exceeded them.

UNIX is a development environment providing both the internal software developer and external software supplier a target environment that was created by and for developers of software systems. It is in such a developers environment that new state of the art development is occurring. This is in part what is making the open systems movement as big as it is today. Our role as internal systems integrators is to harness the benefits of this technology and apply it within our organization in the most cost effective way.

UNIX	is a registered trademark of Unix System Laboratories, Inc.
HP-UX	is a trademark of the Hewlett Packard Company
X-Window System	is a trademark of M.I.T.
SCO-UNIX	is a trademark of the Sorta Cruz Operation
RTE-A	is a trademark of the Hewlett Packard Company
IMAGE	is a trademark of the Hewlett Packard Company
CIM/21	is a trademark of Industrial Systems, Inc.
INGRES	is a trademark of the Ingres Corporation
ORACLE	is a trademark of the Oracle Corporation
PROVUE	is a registered trademark of Fisher Controls International, Inc.
ALLBASE	is a trademark of the Hewlett Packard Company
PORT-RX	is a trademark of the Hewlett Packard Company

Defining HP-3000 printers in HP-UX spooler.

Paper 2008
Sören R. Olsen
Boeing Computer Services
P.O. Box 24346 M/S 6Y-09
Seattle WA, 98124-0346
(206) 477-1534

Background:

When our HP-9000 was acquired, there was a requirement for the existing distributed printers attached to the HP-3000 to be usable by the applications built on the new platform. My task was to solve the problem without the purchase of new software.

I was able to secure usage of an existing program acting as a remote spooler on an HP-3000 for other HP-3000's. This offered a potential solution. I had to develop a method to enable the HP-9000 to send a file to the HP-3000 that could be routed to a printer. The receiving HP-3000 would not care what environment instituted the print request, just simply provide the service. The spooling scenario would be built around what the solution provided.

The data was passed to the print server in two distinct files. First, the data to actually be printed was transmitted, and then a file containing control information is transmitted. At this time, the control file contains the fully qualified MPE filename with the location of the data to be printed. As the project progressed, we found that this was a perfect place to add routing information so multiple printers could be supported with a single script.

Three discrete pieces were identified that were required to make this all work. First, I had to interface with the native spooling system used by HP-UX, so the application developers would not have to be concerned about coding several work-arounds to provide print output for the user community. The next part of the puzzle was to provide a transport mechanism to get the data to be printed from the machine hosting the application to the machine with the printer. The final part was for the receiving machine to realize that something had been sent to it, and to understand what it was supposed to do once received.

The Early Stages:

Although there are volumes of information dealing with printers on your own system as well as other Unix systems, there is almost nothing available dealing with printers that are on any other type of system. Because of this, the search for proper information was not as straight forward as I had hoped. There were several paragraphs found in the *Installing and Administering Network Services Manual* discussing the use of NS to set up a remote printer on another Unix system. I believed if NS could be used to do this, it also could be used to setup a remote printer on the HP-3000.

Armed with this information, I decided to write a script to send the files across to the HP-3000 making use of the program that I had secured for that side to give us a work-around until I could set up a proper interface with the native HP-UX spooling system. At this point the *NS Cross-System NFT Reference Manual* became my best friend. This contained the information on the syntactic idiosyncrasies of the DSCOPY command. Getting DSCOPY to work became the first major milestone that provided the developers with a method for printing their work locally. They no longer had to rely on printouts delivered from the remotely located computer room.

HP-UX:

Once the work-around in place, it was time to study actual print driver scripts to see how HP-UX communicates with its printers. Since both shell scripts and 'C' programs are allowable as printer drivers, I was fortunate because I had Bourne Shell scripts exclusively. Quite a library of these scripts were found in the **'/usr/spool/lp/model'** directory. These range in complexity from one script named 'dumb' to quite sophisticated scripts that support every feature of the newest laserjet printers.

In addition, there is a model script for providing remote status as well as remote print request cancellation. These are found in the **'/usr/spool/lp/smodel'** and **'/usr/spool/lp/cmodel'** directories. The only scripts that I found here were the ones used for handling a remote Unix system. Following is the directory structure supporting the spooling system.

/usr				
	/spool			
		/lp		
			/cinterface	*Remote Cancel Scripts*
			/class	*Lists of devices in a class*
			/cmodel	*Remote cancel script models*
			/fonts	*Printer soft fonts.*
			/info	*Empty on my machine*
			/interface	*Print drivers in use*
			/member	*Device driver & class name*
			/model	*Print driver model scripts*
			/receive	
			/request	*request info*
			/sinterface	*Remote status scripts*
			/smodel	*Remote status script models*

HP-UX spooler directory structure

Each of these shell scripts in the **model** subdirectory is designed to accept a definitive set of parameters generated by the Unix 'lp' command. Understanding these parameters and how the 'lp' command and these scripts communicate is key to understanding the spooling system.

When a printer is installed on the system, the name of the script in the **interface** subdirectory is also the name of the device that it supports. Each of the scripts extracts this information (as: `basename $0`) to use in the standard banner that is printed. Next, the positional parameters $1 through $4 are used consistently for the same items of information.

$1	*print request id*
$2	*requesting user id*
$3	*title*
$4	*copies*

HP-UX spooler passed arguments

Following this, there will be localized script options optionally defined to 'lp' by '-o' options when 'lpadmin' was used to first define the printer. Several scripts check argument $5 for these optional parameters. The first run through did not support any 'printer specific' options. (This was left for a later enhancement.) Next will come the file or set of files to be printed. The scripts therefore loop the actual printing process for the number of files that it found on its command line. After performing 5 'shifts', the scripts set a variable 'files' to the contents of '$*', which are all remaining arguments.

Network:

With an MPE background, when the subject of transferring files from one machine to another came up, my first thoughts turn to using HP's Network Services (or more specific DSCOPY). HP's *Cross System NFT Manual* proved invaluable at this point.

HP's 'dscopy' utility remains functionally similar over both platforms. Both versions support the syntactic customs of their respective environments as well. Some decisions had to be made regarding limitations to impose for the sake of controlling the environment. The decision was made to limit output sent across to fixed length ASCII records of 134 characters which may or may not contain carriage control characters. As a result the following command was constructed to support these options. The HP-UX documentation is very specific about the syntactic conventions that must be followed.

```
dscopy -A -F -L134 $hp9k_file $hp3k_node#"$hp3k_logid"#$hp3k_file
```

Sample DSCOPY command

Note that the pound sign (#) separator is used to delimit the three major pieces of the MPE identifier. This identifier is composed of the 'node name', a valid logon including passwords all enclosed in quotes, and the fully qualified filename (that the logon will have write/save access to). Using this method of transmitting data between the two environments does require supplying the 'DSCOPY' program with all of the logon information for the other system, and then 'DSCOPY' creates its own session on the receiving system. When there is a transmission failure, 'dscopy' will return the information to you, however the spooling system on HP-UX will protect you from directly receiving the information. Although HP-UX spooler will send the information to you via the mail system when 'DSCOPY' returns a fatal error, I have found it useful to have a script that bypasses the spooler and sends the files directly to 'DSCOPY'. This is a handy tool to have around when you need to debug network abnormalities.

MPE:

As stated earlier, the MPE piece of this puzzle started with a program that already existed that had been helping multiple HP-3000's to share printers.

Our program is written in 'C'; however, your program can be written in whatever language you are comfortable with.

The basic flow of this program is as follows:

* List a directory with 'listf, 2'
* Programmatically scan the results.
* Open a file when one exists.
* Read the contents, searching for fully qualified file names.
* Attempt to open filename pointed to.
* If successful, direct to device requested.
* Purge both files.
* Repeat loop.

HP-3000 server process flow

Following these basic steps will allow you to duplicate this process with relative ease. We have done nothing out of the ordinary, nor have we used any special system capabilities in putting this together.

The account that is being scanned is designed strictly to be a 'spooler' account for this program. The 'normal' state of this account is totally empty. There are files in it only when something is awaiting printing. It's structure is as follows:

LASER9K:		
	CCTL:	*Files to be printed with CCTL.*
	FLAG:	*Files that are SCANNED.*
	PUB:	
	NOCCTL:	*Files to be printed WITHOUT CCTL.*

HP-3000 Account Structure

The 'PUB' group currently is not used by the spooling process. It is strictly the 'home' group for the account manager. The 'FLAG' group is where the 'control' file is deposited. The 'CCTL' and 'NOCCTL' groups are where we are going to place the data to be printed. Files deposited in the 'CCTL' group will be printed with carriage control directives, and those deposited in the 'NOCCTL' group will be printed without carriage control directives. The first working model that interfaced with the HP-UX spooler only supports the

'NOCCTL' group; however, the work-around script that bypasses the HP-UX spooler supports both.

In our implementation, this process operates as a 'son' process of our 'IDLEUTIL' utility. This could be modified from site to site depending on whatever job scheduling process you happen to have. This is a process that you will probably want to keep as simple as possible, with most printing features supported in the HP-UX printer driver script.

The Script:

The script begins with the setting of variables that make the script more maintainable and portable. These variables start out with ones to hold the information about the receiving HP-3000. These variables can then be easily filled with the proper data for your installation, and even changed to support printers on multiple HP-3000 nodes with relative ease.

```
hp3knode="node.company.com"
hp3kclog="user/password.laser9k,cctl"
hp3knlog="user/password.laser9k,nocctl"
hp3kflog="user/password.laser9k,flag"
cctl_grp=".cctl.laser9k"
nocctl_grp=".nocctl.laser9k"
flag_grp=".flag.laser9k"
```

Script Communication Variables

Next, the names of the work files used by the script are placed into variables, and the previous work files are deleted. I opted to use work files in putting this scenario together rather that use 'stdout' for everything for several reasons. The most important of these was my comfort with the logic involved with this methodology.

Leaving the work files on the system between print requests enables you to see all of the particulars about a print request in the event a failure occurs in the transmission of the file to the HP-3000 or causes a problem on the HP-3000 itself. This is not mandatory, and could be handled differently without any detrimental impact on your spooling system.

```
dev=`basename $0`
x="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
bannerf=".lj.print.banner"
temp_file=".hp3k.print"
rm -f $ctl_file
rm -f $bannerf
rm -f $temp_file
```

Script Variables continued

Next, we must create a valid but unique MPE filename. The chosen methodology was to select the first four characters of the HP-UX userid, and suffix this with the minutes and seconds from the system clock. This has proven to be a very reliable method, since we always have unique filenames for the HP-3000, and if something goes wrong, we can track down the owner of the print request from the HP-3000 side as well.

```
test_name=$2
name_length=`expr length $test_name`
if
    [ $name_length -gt 4 ]
  then
    base_file_name=`echo $2 | cut -c1-4`
  else
    base_file_name=$2
fi
file_suffix=`date +%M%S`
todays_date=`date`
hp3k_file_name=`echo $base_file_name$file_suffix`
ctl_name=`echo $hp3k_file_name$flag_grp`
hp3k_file=`echo $hp3k_file_name$nocctl_grp`
```

Script Work Filename Creation

Now it is time to create the 'standard' HP-UX print banner. This is done like every other printer interface script except we use a work file to hold the information for us until we are ready to assemble it with the file to be printed for transmission to the HP-3000.

```
echo "$x\n$x\n$x\n$x\n" > $bannerf
banner "$2" >> $bannerf
echo "\n" >> $bannerf
user=`pwget -n $2 | line | cut -d: -f5`
if [ -n "$user" ]
then
      echo "User: $user\n" >> $bannerf
else
      echo "\n" >> $bannerf
fi
echo "Request id: $1    Printer: `basename $0`" >> $bannerf
echo "Remote id:  $hp3k_file_name" >> $bannerf
echo "$todays_date" >> $bannerf
echo "\n" >> $bannerf
if [ -n "$3" ]
then
      banner $3 >> $bannerf
fi
copies=$4
echo "\014\c" >> $bannerf
```

Script Print Banner Creation

Finally, now is time to put the banner and data together along with the control information, and send the files across to the HP-3000. To build the control file, we first place the fully qualified MPE file name into the first record of the control file. Then we place the print device in the second record, and the number of copies in the third record. Then we use the 'cat' command to put the two components of the data together into the file that we transmit. Usage of the 'cat' command will also simplify the task of adding functionality such as compressed landscape to the printer interface, by inserting file names into the 'cat' command line that contain the proper escape sequences to turn these functions on and off at the appropriate times.

```
echo "$hp3k_file" > $ctl_file
echo "DEVICE=$dev" >> $ctl_file
cat $bannerf $file > $temp_file
dscopy -A -F -L134 $temp_file $hp3knode#"$hp3knlog"#$hp3k_file
dscopy -A -F -L72 $ctl_file $hp3knode#"$hp3kflog"#$ctl_name
exit 0
```

Script NS data transfer

Network alternatives:

Although I chose to use HP's Network Services for my initial network transport mechanism, that is not the only choice available. Any network transport that you have available on both of your HP platforms can perform this portion of the task for you.

If ARPA services is your network transport, then explore usage of FTP. You will have to do the research for the exact syntax to use, but the basic utility is similar.

If you are exploring the usage of Interprocess communications between your two platforms, this would offer you the most robust scenario of all. Here you also have a choice between Net IPC and ARPA/Berkeley services for your specific protocol. Either of these will work to accomplish the task.

The Next Phase:

As more features are added to this project, more features are being requested. This leads us to the 'where do we go from here' scenario. Well, we do have several plans in the works to make this a more robust system.

One of the items to be addressed is dynamic support for printer enhancements such as compressed portrait or compressed landscape. We are supporting this from within the application by embedding escape sequences in the print files, but this still provides a limited capability.

We also want to be able to provide accurate print status across the platforms that is currently beyond the technology that we are using. This is leading us to the usage of Interprocess Communications between the platforms.

Another requirement leading us toward the usage of Interprocess Communications is the requirement to provide print services for PC applications such as word processing, spreadsheets and business graphics. We feel that this technology may be what is needed for our users in current environment.

Conclusions:

In the final analysis, you can, with relative ease take control of your spooling problems when you add an HP-9000 platform to your existing HP-3000 platform without having to purchase a lot of extra software to get the job done. In times when the economy is not as robust as we would like it to be, quality service does not have to suffer unnecessarily. The spooling environment as described here can be as simple or as complex as your own needs dictate. If your budget is small and your needs relatively simple, it can be done for a relatively minor cost to your company. As your needs grow, you can add functionality a small piece at a time without destroying the budget. And, your users will appreciate your efforts.

Paper 2009

How Do You Know That Program Will Work?

by
Frederick W. Ochs, Jr.
Epic Consulting
615 Harbor Road
Alameda, CA 94501
(510) 865-6331

One answer to that question is by testing it. Testing does several things to promote program correctness: (1) It finds bugs. (2) It finds no bugs. Both of these results are valuable. If you find a bug, you can document the error and get it fixed. If you don't find a bug, you can note that a particular aspect of the program works, and you can use that fact to build user confidence in the program. You can't loose; that's one reason testing is a rewarding activity.

I have been developing a test system for Orbit Software and I would like to use it as an example of one approach to testing. Of course this is just one approach, but it produced good results and I hope it will is a useful source of ideas for your own testing. The software being tested is Orbit's native mode backup utility, Backup+/XL. So these tests are more extensive than those needed for a less critical piece of software. Nevertheless, many of the issues and problems I faced should apply to other software development as well.

From previous experience we knew we could never test enough to prove that the program was completely bug free. First, there are statements like this all over the place:

```
If Corrupted_Heap then
     Abort_Program (...
```

You know the type, the situation that is never supposed to happen. However, you need to code for it anyway because if it does happen and the program ignores it, the results are horrendous. Frequently you just can not test this type of logic.

Then there are structures like while loops.

```
While Next_Transaction_Number < Number_Transactions do
     Total := Total + Transaction_Amount;
```

How many tests do you need to "prove" this loop works? Should we run tests where the loop body does not get executed at all? How about ones where it gets executed once? What about twice? Is there a maximum value of Number_Of_Transactions that we should test? How about one less than that? How about one more? And what is a "typical" number of iterations. Maybe we should test that, too. Multiply this number of tests by the number of looping and other branching constructs and the result is an unmanageably large number of tests.

To make matters even worse, think about the more complex interactions where an IF being true in one part of the code makes a WHILE in another section behave differently. What about the good old divide by 0 that crops up now and then? How about table overflows? Integer overflows? YIKES!!!

Great! Now that I know that testing is either impossible or impossibly expensive, what do I do? Again, as learned from experience, I knew that testing is a statistical game and that I had to stack the odds in my favor. Basically, I needed to balance the effort of creating a particular test against the likelihood of an error in the code to be tested and against the consequences if there were an error in that code.

It's not as tough as it sounds. By taking a judiciously designed approach we were able to get excellent results from the test system.

Designing Tests

In designing the tests I wanted included in the suite, I used the principle that sheer numbers of tests do not mean the program is well tested. In developing this test system as well as in earlier ones, it was clear that a wide variety of tests, including ones checking boundary conditions, exposed far more problems than just a bunch of variations on the same basic test. So part of the test system design was a systematically constructed list of tests.

Figure 1 shows a portion of the test list. It is constructed in outline form. Tests are needed at the lowest (most detailed) levels of the outline. To read the entire meaning of a test, you need to read from the highest level and down to the lowest. For example test case 2.1.1.2.2 is a store where the file set is specified on the store command line. There is only one file in the file set and it is selected using wild card characters. I found that using an outlining technique made it easy for me to enumerate a large number of the ways a particular product feature could be tested.

```
2  File Sets
  2.1  STORE
   2.1.1  Command line file sets
    2.1.1.1  Files from a private volume.
             Job: TD0038J  Test:  954  Script: TP0122S
    2.1.1.2  One file in file set
     2.1.1.2.1  Actual file name
             Job: TD0012J  Test:   17  Script: TP0063S
     2.1.1.2.2  Wild card file name
             Job: TD0012J  Test:  246  Script: TP0063S
     2.1.1.2.3  With excluded files.
             Job: TD0012J  Test:  275  Script: TP0063S
     2.1.1.2.4  Empty file.
             Job: TD0012J  Test:  283  Script: TP0063S
```

Part of the List of Tests
Figure 1

As tests are written, they are entered into this document. For example you can see that test case 2.1.1.1 is run in test number 954 which is in job TD0038J. I have also documented the script used by each test. So this part of the test design becomes a living document that makes it easy to track the progress of test suite construction and to find the tests relating to a specific feature of the software I am testing.

System vs Unit Testing

Two traditional approaches to testing are system and unit testing. In unit testing, the entire product is not tested. Instead various portions are isolated and tested on their own. Since these portions are typically subroutines, "driver" programs are written to call the subroutines with canned data and then to record the results.

In system testing the entire product is tested. Here the construction of tests is simplified by using the regular user interface to enter commands for each test. In my case that meant the Backup+/XL store, restore, etc. commands.

Unit testing can have the advantage of more complete coverage of the functions a subroutine is expected to handle. On the other hand, system testing tests the actual product, and it does a better job of checking the interactions between the various modules making up the entire product.

We actually took the two-fold approach of using unit testing for the most critical modules and system testing for the bulk of the testing. This paper uses examples from the system tests, but the same principles were applied to both kinds of testing.

Regression Testing

Another design consideration was to use a regression testing approach. The idea here is that I would construct a suite of tests that would give me reproducible results each time they were run. In that way each new version of the product is run through the regression suite to make sure we didn't "break" anything that used to work.

The easiest way to create a regression test in MPE is to create jobs that run the tests. However, some tests require so much manual intervention that documented procedures are used to run them. A good example is the full system reload test. When the reload is being done, there is no operating system in place to even run a job. However, by using both jobs and procedures we do have a good solid regression approach to the tests.

Of course for each new version of the product there are new features and bug fixes. So I add tests to the suite to cover those new items as needed.

Command Files - Scripts

As I mentioned, jobs are great for regular regression testing. I just code each job once (or twice or three times depending on how good a programmer I am). Then whenever I need to test the system, I stream them and a bunch of tests are executed without further intervention. At the end of the job, I check the entire set of test results.

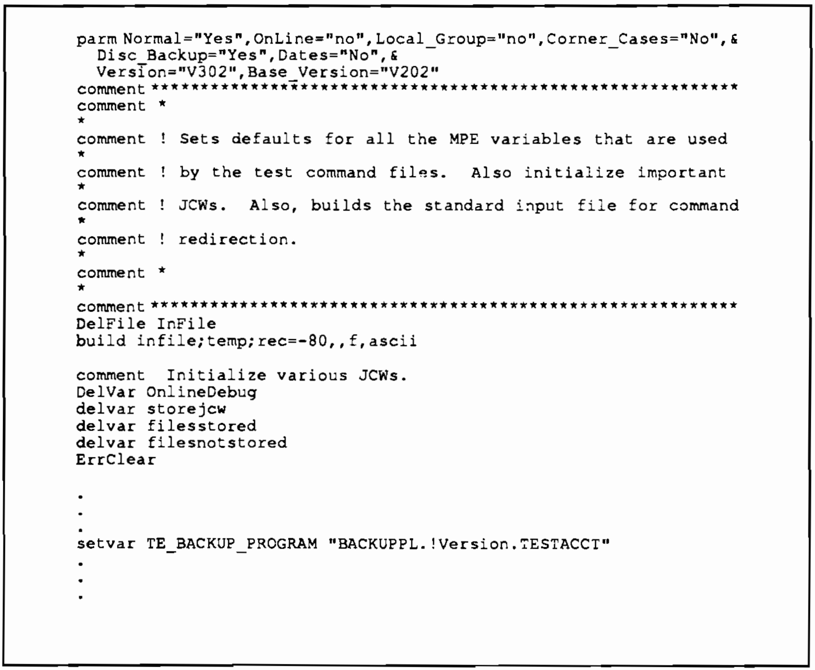
But what if one of the developers tells me he just fixed bug 127, and he wants to run a test to check the fix. In this situation, it is much more convenient to run just one or two tests that are particularly aimed at the bug 127 case. I could write a job to test the fix, but since it's only a couple of tests and they are already coded in some larger job in the test suite, I prefer to run them interactively. However, I also do not want to type in the 30 or so commands to run each test, because it is so inefficient.

So I code up command files (or scripts as they are more frequently called) for all tests. To run a test interactively, I just type in a one line command with some parameters and the script runs the test and checks the results for me. To run it in a job, I code the same line in a job file and stream the file.

Scripts have a number of other valuable uses.

A single script can be used to run a broad class of tests. For example, there are over 100 date selection tests in the QA system. Without using scripts (or UDC's) I would have had to code 100 individual tests at 95 lines each. Tedious to be sure. But also extremely error prone. Instead I have one command file that runs all of these tests. Each test simply has a different set of parameters.

Maintenance is another issue. Let's say that we rename the product, which in fact we did at one point. And let's say I have 1100 tests, which I do. Without script files, I would need to change the program name in 1100 places. Yech! If I had 200 test scripts, I only need to change the program name in 200 places. Still pretty yechy. However, in one of those rare moments of doing something right



```
parm Normal="Yes",OnLine="no",Local_Group="no",Corner_Cases="No",&
  Disc_Backup="Yes",Dates="No",&
  Version="V302",Base_Version="V202"
comment ************************************************************
comment *
*
comment ! Sets defaults for all the MPE variables that are used
*
comment ! by the test command files.  Also initialize important
*
comment ! JCWs.  Also, builds the standard input file for command
*
comment ! redirection.
*
comment *
*
comment ************************************************************
DelFile InFile
build infile;temp;rec=-80,,f,ascii

comment  Initialize various JCWs.
DelVar OnlineDebug
delvar storejcw
delvar filesstored
delvar filesnotstored
ErrClear

.
.
.
setvar TE_BACKUP_PROGRAM "BACKUPPL.!Version.TESTACCT"
.
.
.
```

Part of The Initialization Script
Figure 2

the first time, I developed a script (See Figure 2) that does nothing but initialize system variables with things like the name of the program. So when it came time to do the rename it took me about 2 minutes to change all 1100 tests. All I had to do was change the initialization script, and the new name was used everywhere the next time the suite was run. This flexibility has been useful too often to mention.

Another compelling reason to use scripts is the modularity they so nicely support. As you can imagine, most of the tests I wrote store and restore files. Indeed one of the key requirements of a backup utility is that it successfully restore files (Picky customers!). So in every one of the tests that do a restore, I compare the stored files to the restored files to make sure they are equal in all the right ways. Here again coding the file comparison into 200 or so scripts is maximum boredom and minimum productivity. Instead I wrote just one script having a suitable set of parameters that specializes in checking two sets of files for equality. My CPFiles script is then referenced in all 150 or so scripts with a simple one line command.

I call scripts like CPFiles utility scripts. Of the 400 or so scripts in the test system, probably half of them fall into this category. Because of this large toolbox of utilities, I can whip a new test script together very quickly compared to coding without them. They do all kinds of things including date calculations, find the amount of free space on the system, check stdlist test results, load tape drives, and much more.

I can not say enough for scripts. If you have not found the simple joys of script writing, you are missing out on some serious productivity gains and improvement in the quality of your life.

Efficiency

To make running the tests as efficient as possible, we sought a high level of automation. By automation I mean automatically checking test results, automatically running as many tests as possible, and automatically restarting the suite in the event of an interruption in the end to end execution of the entire suite. As mentioned above, not all tests could be automated. But Figure 3 illustrates the distribution I ended up with.

Level of Automation	**% of Tests**
Fully Automated	90%
Partially Automated	7%
Manual	3%

Figure 3

In addition to efficiency we also gain accuracy through automation. For example, it is just too tedious for a human being (me) to manually check the results of thousands of tests and expect accuracy. Here again scripts came to the rescue.

I have already mentioned the script used to compare stored and restored files. However, there are other result checking scripts that check JCW values, report output, etc. Usually some reference data are involved. In the case of the file comparison script, it uses the stored files as a reference. Similarly, report checking scripts usually compare the report gotten from the test run to a reference report from a previous run.

Since the automated suite is made up of a bunch of jobs, we needed some way to automatically run all of them in a way that would keep different tests that required the same resources from running at the same time. Two scripts were enlisted for this scheduling. A script named CallJob will stream a job and then wait for it to complete before returning. It will also optionally adjust the priority of each job to prevent system performance degradation. A second script is used to call calljob in a loop. Each time through the loop it will invoke calljob with a different job name to be streamed. See Figure 4 for an example of this technique.

```
!JOB ALLTT,USER.TESTACCT
!
!COMMENT  THIS JOB RUNS AUTOMATED TAPE TEST JOBS SEQUENTIALLY.
!
!COMMENT REDIRECT LASTJOB TO THE FILE THAT PROVIDES JOB
!COMMENT RESTART TRACKING FOR TAPE TEST JOBS.
!
!FILE LASTJOB=TTJOB.PUB,OLD;SHR
!
!COMMENT GET A LIST OF ALL TAPE TEST JOBS AND USE CALL JOB
!COMMENT TO STREAM THEM ONE AT A TIME.
!
!PatCommd TT@J.NMJob,CallJob %
!
!COMMENT RESET THE RESTART TRACKING FILE.
!
!ECHO 0 > *LASTJOB
!EOJ
```

A Job to Sequentially Stream Test Jobs
Figure 4

To provide a test restart facility I used (guess what) scripts that would run a particular test only if it had not previously been run in the current Q.A. cycle. If I need to rerun a particular test, this control is implemented in a way that is easily overridden. By editing a one line file, I can get the system to rerun a test that has been previously executed or, more simply, I just run the test script in a session. Figure 5 illustrates a job coded to take advantage of the RunTest scripts restart logic. As you can see, RunTest takes as one of its parameters the test script and its parms to be used in running the test. If the tests has already been run, RunTest does not execute the test script. If it has not been run RunTest simply passes the test script off to the command interpreter.

```
!JOB TT0002J,MGR.TESTACCT
!
!COMMENT  TAPE TESTS.  THESE TESTS USE THE DAT.  NO OPERATOR
!COMMENT  INTERVENTION REQUIRED IF TE_AUTO IS TRUE.
!
!SetVar Job_Dev "DAT"
!
!COMMENT   Autoreply tests - Both MPE and Product autoreply
!
!COMMENT   Test Case 51.3              Test 114
!
!RunTest 114,&
!TP0040S FILE_EQ_DEV=DAT,AUTOREPLY_DEV=14
.
.
.
!ENDJOB
!EOJ
```

Using The RunTest Script
Figure 5

Tracking Errors

Of course everything I have mentioned so far is useless if we don't give high visibility to an error that has occurred. In a suite of 1100 tests, you can imagine how easily one failure can get lost in a sea of success. (We should always be lucky enough to have only one failure.)

I took a two-fold approach to this. First the results of each test in a particular job are recorded in a results file as shown in Figure 6.

```
Job TD0001J started.  #J134 FRI, MAR 20, 1992 11:11 PM #O158
TEST 1 STARTED FRI, MAR 20, 1992 11:11 PM
TEST 1 SUCCESSFULLY COMPLETED
TEST 2 STARTED FRI, MAR 20, 1992 11:47 PM
TEST 2 SUCCESSFULLY COMPLETED
TEST 4 STARTED FRI, MAR 20, 1992 11:48 PM
TEST 4 SUCCESSFULLY COMPLETED
TEST 5 STARTED FRI, MAR 20, 1992 11:50 PM
TEST 5 SUCCESSFULLY COMPLETED
TEST 10 STARTED FRI, MAR 20, 1992 11:55 PM
TEST 10 SUCCESSFULLY COMPLETED
TEST 18 STARTED SAT, MAR 21, 1992 12:30 AM
TEST 18 SUCCESSFULLY COMPLETED
TEST 399 STARTED SAT, MAR 21, 1992 12:37 AM
TEST 399 FAILED COMPJCW=2010
Job TD0001J done. SAT, MAR 21, 1992 12:42 AM
```

A Job's Results File
Figure 6

Then as part of RunTest's processing, it checks the result of the test it just ran. If an error occurred, RunTest finds the spool file for the job it is running in, and then appends just that part of the spool file pertaining to the failing test to a file called FAILURES. So all I need to do is read through the FAILURES file to find the whole story on any failing tests. In the event of a single failure in the entire suite, this is 1100 time faster than checking each test. And it is a zillion times more accurate. I have to put in a plug for the native mode spooler here. In order to write the scripts that find and extract the stdlist data I want, it took me about 2 hours and 100 lines of command file programming. Try doing that with the old spooler!

Trapping Intermittent Errors

A final design consideration was driven by the intermittent errors we encounter from time to time. An error would occur as the automated suite was running and when the developers later tried to debug the failure, they were unable to get it to happen again. So to aid in debugging these flaky errors, we felt the need to preserve the environment in which a test had run as much as possible. What is the environment? It is things like the file set that was being stored and restored, and the back up file that was created.

Here again the RunTest script helps out. All the test resources needed by a particular test are reserved at the start of the test. These resources are things like groups of files to store, tape drives, disc backup file names, etc. When RunTest detects a successful test result, it returns these resources to the pool of available resources. However if it detects an error result, it will remove all resources used by the failing test from action so that other tests will not use or modify them. Then the developers can take their time in debugging, assured that the environment will not change under them.

But what happens if some other test is trying to run and needs one of the resources held for debugging? In some cases we have multiple copies of a particular resource. For example, we have three copies of a file set that is frequently used in tests. If multiple copies are available the test will search each of them until it finds one that is available. If none are it will wait until one becomes available.

In this way the suite provides pretty good trapping of intermittent failures while at the same time moving the suite forward as quickly as possible.

Results

We have had very gratifying results from developing the suite. It is difficult to put a number on them, but one reasonable measure is the percentage of bugs caught before the product goes into beta tests as compared to total number of bugs found before release. The total would include those found in beta testing as well. Using this approach a 10% bug catching would be pretty poor performance. However, 90% would be pretty darn good. I can hardly believe it myself but in the latest release cycle, we were batting 1000. I certainly am not going to play "you bet your contract" on 100% results every time we have a new release, but this ratio is a clear indication that the suite is very effective.

Please don't think this ratio indicates that there are not going to be any errors found after the product goes into full distribution. We know we can't test absolutely everything. But we know there are more than a bread box full of bugs that will never see a customer site and that is very rewarding.

As I have mentioned, the suite now consists of over 1100 individual tests which exercise over 2000 different test cases. The automated system runs for about 10 days, and semi-automated and manual tests for another 48 hours or so. Do not get the idea that all test programs need to go to these lengths. But, we feel that testing this extensive is critical to the success of software in the operating systems utility field.

To Be Continued

In addition to adding tests to the suite, I also plan to redistribute tests into more logically comprised jobs. Currently, the jobs are basically disorganized groups of tests that happen to all use a particular hardware device for their tests. For example all tests in one job would use a DAT drive. In the future each job will be organized both by hardware requirements and the feature they are testing. For example, one job might contain all data compression tests. Then when a change is being made to the data compression portion of the product, we can run just the data compression testing jobs as an aid to the development process itself.

Also I will (gulp) document the suite and how it is run. A task that is long overdue.

Dear HP

I tried to think of ways HP could contribute to better testing on HP/3000s. The most pressing item in my opinion is to modify their compilers to provide path flow instrumentation. The idea is that we would compile our programs with an option that would cause the compiler to insert code at the start of each possible code path through the program.

```
If Count = Max_Count then
     writeln ('Out of room in table.')
else
     Count := Count + 1;
```

For example, in the above code fragment there are two paths: one contains the writeln and the other the assignment.

Then when the program is run, some utilities provided by HP would count the number of times each path is run and report them. That would tell us how well the tests are exercising various parts of the program. Armed with that information we can design tests to hit areas not well covered.

There is already a facility of this sort on HP/UX machines so it should not be difficult to port it over to 3000's. Keep those SR's coming.

Paper 2010

AN ELECTRONIC DATA INTERCHANGE PRIMER

by
Frederick W. Ochs, Jr.
Epic Consulting
615 Harbor Road
Alameda, CA 94501
(510) 865-6331

INTRODUCTION

What's wrong with this picture?

Walt has almost wrapped up a long day at the computer terminal. He is a buyer at a large computer manufacturer and has just completed on-line approval for 7 purchase orders for key components needed for his company's assembly lines. "Now," he thinks with a sigh of exhaustion, "all I need to do is print the PO's out and get them in the mail." A half hour later the purchase orders are printed and ready to be entrusted to the care of the US Postal Service.

A couple of days later Marge opens her mail and finds Walt's purchase order among the dozen or so others she received. Marge turns to her terminal and fires up the order entry program to key in the batch of PO's sitting in front of her.

What is wrong is that, in today's technological world, we are taking data in one computer system, and putting it on paper only to reenter it into another computer system. At many large companies this useless rekeying of data is even going on within the same organization on the same system. Electronic Data Interchange or EDI is the term applied to eliminating just this sort of paper shuffling (Figure 1). It is the machine readable transfer of business documents from one company's computer to another.

The development of Electronic Funds Transfer Systems (EFTS) and intercompany order entry started in the 70's. At that time the underlying technology was already in place, because all EDI requires is a couple of computerized companies able to communicate with each to accomplish some business need.

BUSINESS BENEFITS

While the basic concept of EDI is simple, the company benefits are impressive. The reason companies are interested in EDI is for all the repercussions of eliminating the traditional document flow shown in Figure 1. At least two separate studies have pointed out that 70% of the data keyed into the computer systems in the US has already been keyed into another machine before. Think of the savings in eliminating all of that. I don't want to go into a full blown justification of EDI because it would take too long and the justification is most meaningful when tailored to a particular company. However, Table 1 presents a brief list of some of the cost saving and business enhancing benefits of EDI. You may enjoy verifying for yourself that all of them come from the elimination of printing some document on paper, getting it to your business partner, and they having to key it again.

PARTIAL LIST OF BENEFITS OF EDI
HANDLING AND POSTAGE EXPENSE
INVENTORY
IMPROVED CUSTOMER SERVICE
JUST IN TIME INVENTORY MANAGEMENT
IMPROVED ACCURACY
IMPROVED LOGISTICS
IMPROVED SPEED

Table 1

However, apart from these that are more or less obvious, there is an entire class of benefits that have to do with changing the basic nature of your business. They depend heavily on the kind of business you are in. For example, I once did a project for a company that sold janitorial supplies. In that business, there is always some percentage of items that are sold out. If a customer orders one of those items, it can not be shipped immediately but must be back ordered. Early in our EDI development we started receiving EDI purchase orders from a large bank with four or five hundred orders per transmission. As a result of processing the orders electronically, we saved almost a week in entering and processing them. With that extra week we realized that we had time enough to find out what items we were out of and replenish the stock for them before we needed to deliver the orders. Think of the implications of that. Because of EDI we could give the customer guaranteed 100% filled orders, save ourselves the expense of backorders, and receive the payments for items that would have been backordered several weeks earlier.

STANDARDS

What does the data we transmit look like in EDI. At first, two companies would just sit down and design a record format that work for both parties and use it. However, it soon became clear that this approach could not go on for long. There needed to be some kind of standards so that the amount of work needed to support the exchanges between a large number of trading partners would be more manageable. (A Trading Partner is the generally accepted term for the "other company" in an EDI relationship.) And so standards organizations were formed. Boy did they get formed!!!

This has led to new problems. First standards organizations move very slowly. They really have no choice in this because large committees must come to a consensus on the precise formats used in the standards for each document and on the meaning and valid data for every field in the documents. Second, rather than having a large number of private formats to deal with, we now have several standards to deal with (Table 2). In addition, many industry organizations, like EIDX for the electronics industry, add their own wrinkles to the standards. Nevertheless, the situation is far better than it would have been without standards, and the corporate world owes a lot of thanks to the folks who work very hard on developing them.

SOME EDI STANDARDS
ANSI X12
EDIFACT
DATACOMS
UCS
TDCC
WINS

Table 2

THE ANSI STANDARD

Most of the standards have many features in common. One of the most important ones is the ANSI X12 standard, and since it is fairly representative of many standards, understanding it will make it easier to understand many others. Table 3 is a list of some of the business documents currently defined by ANSI. A few years ago only five or six were defined completely, so you can see that the ANSI committees have been hard at work. Probably the most commonly used items on this list are the purchase order, purchase order acknowledgment, invoice, and functional acknowledgment. The numbers listed next to the descriptions are used in the ANSI format to identify the type of document being communicated.

SOME ANSI DOCUMENT TYPES
ANSI 3020

810	-	Invoice
820	-	Payment Order/Remittance Advice
823	-	Lockbox
830	-	Planning Schedule
832	-	Price Sales Catalog
840	-	Request for Quotation
843	-	Response to Request for Quotation
846	-	Inventory Advice
850	-	Purchase Order
855	-	Purchase Order Acknowldegment
856	-	Ship Notice/Mainfest
860	-	Purchase Order Change
862	-	Receiving Advice
862	-	Shipping Schedule
865	-	Purchase Order Change Acknowledgment
867	-	Product Transfer and Resale
869	-	Order Status Inquiry
870	-	Order Status Report
997	-	Functional Acknowledgment

Plus 86 Others

Table 3

WHAT EDI DATA LOOKS LIKE UNDER THE ANSI X12 STANDARD

Figure 2 shows various levels of detail of an ANSI X12 transmission between two trading partners. The concept of enveloping is heavily used in the standard. The entire transmission is called an Interchange.

Within an interchange there are groups of similar documents; these are called functional groups or FGs. You can look at a functional group as a batch of similar documents, for example several PO's might be included in one functional group. The whole idea of this 'functional' grouping is that the included documents are likely to be destined for the same application system at the receiving end.

Within each functional group are the individual documents which are called transaction sets. An example of a transaction set would be an invoice.

Within each transaction set are several segments. It is easiest for us as programmers to think of a segment as a record. These segments contain all the data about the business document. They have variable formats depending roughly corresponding to different areas on the document.

Segments in turn consist of elements. We would refer to them as fields in a data base or record. Each element is one piece of data, for example a part number or billing total. The final composition is that each element is made up of characters. This is the only data type allowed by X12.

Almost all of these levels have segments that indicate the start and end of the entity and provide some error checking for them. At the interchange level, these are contained in segments called the ISA and IEA. In functional groups, the enveloping is done by the GS and GE records, and by ST and SE records for transaction sets. Segments have a slightly different scheme. They begin with a two or three character ID element and end with a segment terminator character. Elements are separated from each other by an element terminator, which is a different character from the segment terminator.

Segments are variable length in ANSI and they are blocked up into fixed length blocks. It doesn't take long to figure out that variable length records (segments) will not in general fit evenly into a fixed length block, so as you will see one segment can be split between blocks. The idea of this is to keep transmission and storage costs as low as possible and to keep programmers employed.

Figure 3 illustrates an 80 byte block of ANSI data. I have highlighted each segment terminator, in this case exclamation marks, with a box. You will notice that there is no length field in the variable length segments. You only know the segment is over when you hit the segment terminator. Immediately after each terminator another segment begins. The segment terminator can be any character that does not appear in the data. Clearly, if it did, it could cause mayhem in downstream programs. (Some standards provide a way around this using an escape type of character.) The segment terminator is specified in the ISA segment of every interchange. Generally it is a non-printing character. I used the exclamation point here just so it would print in the figure.

THE ANATOMY OF A SEGMENT

In Figure 4, I have extracted just the BEG segment from Figure 3.
Each segment's contents and purpose is identified by the segment id which is always in the first element of the segment. The ANSI committees have tried to give some mnemonic value to segment IDs with varying levels of success. For example, a BEG segment id sounds like it labels a segment that ought to go at the start of a document, and so it does. However, another segment is called a PO1 and about the only thing you can guess from this is that it occurs somewhere early (as implied by the 1) in a PO. In fact it occurs in PO's and several other documents, and it occurs a considerable distance into the purchase order transaction set. There may also be additional qualification of the segment's purpose in a second field. For example the N1 segment is for a name. The second element in the segment clarifies whose name it is.

Even more interesting than the fact that the segments are variable length is the concept of the data elements (fields) being variable length as well. This means that two different PO1 segments, for example, in the same transaction set (remember what that is?) may and often will have different lengths. A similar scheme to the segment terminator is used to terminate elements. In this case the special character is called an element separator. It also should be a non-printing character, but seldom is. It also is defined for each interchange in the ISA segment. Missing elements in the middle of the segment simply have their place held by the element separator that would normally terminate the missing data. If the missing elements are at the end of the segment they and their element separators are simply omitted. There is never an element separator after the last element in a segment as the segment terminator fulfills both functions.

In ANSI documentation, elements are named by using the segment ID followed by a two digit number. For example, the second element in the PO1 segment is called PO102.

Figure 5 shows how to figure out what each data element by using the ANSI documentation. Here again the BEG segment is used as an example. Below it is the description of the BEG segment as defined in ANSI version 2002.

The first element in the BEG segment is BEG01. To find out what it means, you find the segment description in the ANSI documentation and look for the box labeled BEG01. As indicated in the figure, this is the transaction set purpose code. Looking back at the sample BEG segment, you can see that the value of BEG01 in this particular example is 06. This code further defines the purpose of the purchase order and is an example of an ANSI ID. Notice that the field number is 353. To look up the possible values of this field we refer to the definition of element 353 in the ANSI documentation. I've reproduced the pertinent section in the lower right part of the figure, where you can see that the 06 indicates this is a confirming order.

The BEG02 is the PO Type code and this example is a CF or confirming purchase order as defined by ANSI. BEG03 is 7214-29A and is the purchase order number. The optional (notice the O in the lower left corner of the BEG04 box) release number and Change Order Sequence Number are omitted in this example. The next field is the PO date of May 10, '89. The remaining fields in the BEG are omitted. All of the segments in ANSI interchanges are pretty much interpreted this way.

Also in the element boxes in the segment description is an indication of the type of data the element is. This is a code in the center of the bottom line of each box. For example, ID for and ANSI ID, AN for alphanumeric, and DT for date. In the lower right hand corner of the box are the minimum and maximum lengths of each field. For example, PO Number can be from 1 to 22 characters in length.

RELATING ANSI TRANSACTIONS SETS TO BUSINESS DOCUMENTS

Now that we have gotten down into the details of ANSI formats, let's back out a bit and try to relate them to business documents. Remember that a transaction set represents an electronic document. There is a pretty clear parallel between a paper purchase order and the PO transaction set specification (Figure 6). A paper PO has a heading area, detail area where there is a repeating format of information about the items being ordered, and some kind of footing area. The PO transaction set has the same kind of structure. There are addresses in the heading area of the paper document that identify the supplier and the shipping addresses for example. Looking at the electronic form you can see the same information present in its header area. In the line item area, we see things that identify what each item is, the quantity ordered, the price we plan to pay, and the expected delivery dates. These are represented in the several PO1 segments. In EDI repeating segments like the PO1 are called loops. You get the idea.

When you think in this kind of analogy, understanding standard formats becomes much easier and very relatable to business needs.

EDI SYSTEMS

Let's look at how you or I might implement an EDI system. From a high level point of view, a system design might look something like Figure 7. Note, that the data is flowing in a cycle from the send to the receiver and back again. As you can see both companies are represented here. The sender's system is at the top, some data communication medium in the middle, and the receiver's system is at the bottom. What I've got in the boxes are the logical functions that would probably have to be performed on any mini or mainframe system. First we have to pull the data to be sent out of the data base of the sender's application system. Then we need to translate that data into the ANSI format. This usually would include things like generating control numbers for the enveloping and logging the activity as well as producing the actual ANSI format. Next whatever mode of data communication will be used is invoked and the ANSI data sent to some data communication service provider.

On the receiving end of the document there is a mirror image of the processing going on. We have another data communication driver, something that translates from arcane ANSI into a format that your application can handle, and finally an update to the system.

The receiver is obligated to acknowledge the receipt of the sender's transmission. Usually this is done at the functional group level with a document called the Functional Acknowledgement (or FA) and may also be done at the document level with a document like the Purchase Order Acknowledgment.

The only paper analogy I've heard of to an FA is sending your PO's through the mail certified with return receipt requested. Basically the FA says, I got your document and it looked ok to me or if there were problems with it, here's what's wrong. This is at a technical level only. So an FA tells the sender that the data conformed to ANSI formats, that there were no missing records, etc. Usually, the FA is thought of as being generated out of the translation section of the receiver's system.

A PO acknowledgement on the other hand pertains directly to an individual transaction set and deals with acknowledging the business contents of the PO. It says I got your order and I will ship it. The processing of it is just like what happens to the PO, except for its direction.

You may be wondering why I drew the FA processing in on the top as well. That's so that the incoming PO acknowledgements are FA'ed. FA's themselves, however, are never FA'ed. Otherwise, we get a loop: You FA my PO. Then I FA your FA of my PO. Then, you FA my FA of your FA of my PO....

However, one thing this figure illustrates is that there are not really EDI senders and receivers. Both partners usually end up doing both.

SUMMARY

I hope I have conveyed a good introductory understanding of what EDI is and what the components of EDI systems are. There is little doubt that EDI is a very important factor in business today and will grow substantially in the coming few years. It is difficult to imagine any industry being untouched by EDI within the next few years and all but the smallest companies will need to support some form of EDI.

Though the formats and terminology seem alien, these matters are very easily conquered and the personal and corporate rewards of working in the field of EDI are soon realized.

A Case Study: Network Design And Implementation

Ken Bare
John Torro

CompuServe Collier-Jackson
3707 West Cherry Street
Tampa, Florida
33607

Synopsis

This paper discusses how Collier-Jackson, a division of CompuServe, designed and implemented a networking client server strategy and the significance of data communications, networking, and client server concepts as an integral part of Management Information Systems. It covers the various steps we followed during the design, installation, and management of a 150-node network. This discussion will focus on how we approached each step, the results, and what we learned.

Key steps discussed are how we:

- Developed a corporate strategy and goals.
- Identified inhouse resources such as employee expertise and available hardware.
- Identified outside expertise such as independent consultants, hardware, and software vendors.
- Designed a multi-phase plan with diagrams and hardware/software connection specifications.
- Acquired the hardware and software based on the network design plan.
- Installed the hardware and software based on the network design plan.
- Migrated from terminals to PCs.

- Managed all the network components and troubleshooting problems.

- Continue to plan for industry technical developments and the integration of new technology into the existing network.

Introduction

As stated in the synopsis, we will discuss how we introduced and implemented networking and client server concepts at Collier-Jackson, Inc. This paper is not meant to dwell on our company history or the chronological events which led up to our current data communication and client server position. We will, however, discuss how we reached our position with the intent that our experiences and endeavors will be educational.

When we proposed a local area network, our data communications environment was dedicated to dumb terminals connected to an HP 3000 via DB25 RS232 connectors and two pair four conductor shielded telephone cable. Our practical experience and expertise in sophisticated packet-based data communications was limited to what we had read in trade journals and newspapers. This position was not a bad place to be. After all, the "Year Of The LAN" was just around the corner, again and again. You invest in what works at the time. I venture to guess that anyone who had a minicomputer five years ago found themselves in a similar data communication environment.

Where Did We Start?

The first step was easy. We decided to remove all administrative and word processing functions from the HP 3000. The obvious answer was a local area network (LAN). Since we did not have prior experience with LANs, we really had much to learn before investing money and time.

When you take that first step into implementing sophisticated data communications technology, you really need a game plan. Develop a corporate data communication strategy and establish goals. This environment is very dynamic and constantly changing.

The corporate data communication plan is the vehicle that will keep your company positioned to take advantage of new technology at the least cost.

We assigned the LAN task to an employee who had a background in microcomputers and some character-based data communications. However, it was not realistic to expect someone to acquire the knowledge and skills in data communication required to design, purchase, and install a LAN. Therefore, we started looking for a consultant to assist us.

We are not going to tell you how to find and select a consultant; we will let a consultant do that. Our process started with a search through the yellow pages. At first we selected those companies with the largest ads, but we didn't stop there. We called friends and acquaintances from local micro computer stores and asked for recommendations, and also asked personal and professional associates for leads.

During this time, we also identified the major LAN players in the game. This turned out to be Novell and Banyan. After a series of phone calls and initial meetings, we had narrowed the consultant search to one company. Shortly, we selected Novell as the LAN vendor.

Obviously, our selection process methodology and logic did not consist of a coin flip or the size of their advertizing. The key factors were:

1. Education - We asked for credentials such as formal college education, professional technical training, and vendor training.

2. Experience and references - We asked for a list of professional and client references.

3. A gut reaction - We were looking for more than a consultant to make recommends and do the work. The relationship with the consultant had to be educational, challenging, rewarding, and fun. It was our intent to keep the responsibility for data communications inhouse and this was our chance to learn.

The Novell file server installation, which supported eight workstations, was successful. We developed a good business relationship with our consultant and both parties were very pleased.

The one networking consideration that posed the greatest problems was cabling. We chose to use coax ethernet cable. Our consultant recommended two cable contractors, but they turned out to be expensive (about $125 a drop). An independent telephone contractor who supported our building telephone cabling needs agreed to string ethernet cable for us. The only task that remained was the installation of BNC end connectors and cable mapping.

A local electrical supply store recommend ethernet cable type RG58 A/U, AMP installation tools, and crimp type connectors. The telephone contractor strung the cable and our network consultant assisted with cable testing. We installed the BNC crimp type connectors, performed testing, created cable maps, installed NIC cards, NIC drivers, and other procedures associated with a LAN installation.

Network Review: Novell file server (FS1)
8 workstations
Ethernet coax cable

Reference Exhibit A

The next phase was to install another Novell file server supporting sixteen workstations. The primary function of the network was to support a call log system which keeps track of support calls. We followed the same methodology as before, and again, it was successful. While installing the ethernet cable for sixteen drops or offices, we decided to install ethernet cable in all offices in the building. The two file servers were connected through an internal bridge, which gave us two LANs that can communicate.

Network Review: Novell file server (FS1)
8 workstations
Novell file server (FS2)
16 workstations
Ethernet coax cable
Internal bridge connecting FS1 and FS2

Reference Exhibit B

Three events or tasks placed us in a position to take advantage of new technology in data communications:

1. Ethernet cable was installed in our building.

2. Dumb terminals were replaced with micro computers.

3. Terminal emulation software was installed on each PC.

The corporate data communication strategy had been modified based on industry trends in connectivity and our interest in client server technology. Basic data communications support was still dependent on serial RS232 technology; however, we now had a good foundation for the future.

The next major phase involved the addition of a new building which is located across the street from the corporate office building. The project requirements included voice and data communications for 50 office locations and a data communications link between the two buildings. Even though the physical distance between the two buildings is a few hundred yards, we treated this connection as a wide area network.

The first consultant was committed to other projects and was not available, so we selected another consultant. Our consultant installed fiber optic and copper cable between the two buildings and a wire cabinet or rack in each building. A fiber optic multiplexer was rack mounted in each wire cabinet for servicing serial RS232 communications between the buildings. Ethertwist eight conductor cable was strung from the wire closet to each office. This cable is used for both voice and data. Each office was equipped with one voice RJ11 jack and two data RJ45 jacks. One RJ45 data jack supports serial RS232 and the other supports ethernet 10Base-T. Patch panels were used to manage connections between the office and multiplexer.

Network Review: Novell file server (FS1)
15 workstations
Novell file server (FS2)
22 workstations
Ethernet coax cable
Internal bridge connecting FS1 and FS2
Fiber multiplexer (RS323 support)
Ethertwist 10 Base-T cable

Reference Exhibit C

The next phase was to connect workstations to the Hosts via ethernet. Each workstation or PC-required terminal emulator software, network connection software, and a network interface card (NIC), all of which could provide a common interface over a wide range of networks and access to a variety of hosts and servers. Those PCs which connected to the Novell file servers had a NIC installed and were connected to the ethernet.

At the time we replaced dumb terminals with PCs we selected Reflection 1® by Walker, Richer & Quinn (WRQ) as a terminal emulator. This product served our PC serial communication needs; however, our communication requirements would become far more sophisticated if we planned on approaching such concepts as client server, Network File Systems (NFS), Posix, distributed CPUs, and distributed data base systems. Our needs now required a combination of RS232, TCP/IP, and LAT protocol access for each workstation and host. Walker, Richer & Quinn developed a product called Reflection Network Series which operates in conjunction with Refection 1 terminal emulation and in combination with an NIC. These products support multiple protocol access to our host devices over ethernet cable.

After taking care of the workstation requirements, we looked at the host devices (HP960, HP58D, HP922, HP9000, IBM/RS600, and a VAX8530). The HP960, HP58D, and HP922 support TCP/IP network services protocol, the HP9000 and IBM/RS6000 support tcp/ip telnet protocol, and the VAX hosts were supported by Local Area Transport (LAT) protocol.

All of these protocols are supported by workstation hardware and software. This makes it possible for a workstation or PC to communicate with the host machines only if a common communication link to transmit and receive packets of information exists.

We proceeded to connect all host devices to a network backbone segment. At first we connected all host nodes to a thick LAN cable segment; however, these thick lan taps proved to be cumbersome and hard to manage. We installed a parallel thin lan backbone and connected the HP and IBM host to it. An HP1010 bridge was used for connecting the thin lan backbone to the thick lan backbone. The HP1010 bridge prevents network traffic from crossing the bridge if that traffic does not have a receiving node on the other side of the bridge.

It quickly became apparent that Novell network traffic, which can coexist with TCP/IP or LAT, need not travel on the thick or thin lan backbone that supports host nodes. An HP1010 bridge was placed between the network segment with workstation nodes and the segment with host nodes.

A repeater was installed to support four workstation cable segments of one hundred and eighty five meters each. The repeater was selected for managing cable segments and eliminating internal Novell file server bridges. The repeater proved to be a valuable device for physically locating a network problem on a network segment. Once a problem was located, it was quite easy to disconnect the faulty segment, leaving three other segments to establish network communication once again.

Up to now I have dealt with hardware and the physical connections between that hardware. Obviously, hardware is not the only piece to a data communication puzzle. I have mentioned PCs or workstations running DOS and terminal emulation software. Each Host in turn must be running software that can communicate with terminal emulation software running on the PCs. The HP system manager configured the Host data communication program services.

Configuring the HP 3000 for LAN access

With the large number of NS/3000 Data Communications manuals, one may think that configuring an HP 3000 for LAN access would be overly complicated. As long as you're not dealing with Gateways and Routers, this is not the case. The physical path of the LANIC card is the most important piece of information. Only one card is necessary to support both an ETHERNET LAN and a DTC. This information, together with a local node name (*NODENAME.DOMAIN.ORGANIZATION*), an IP address and a Network Interface name, the *Guided Config* NMMGR screen is about all you'll need to configure.

Once this is configured, the following MPE/XL commands are all that is needed to make your LAN accessible:

```
:NETCONTROL START;NET=LOOP
:NETCONTROL START;NET=your_lan_name
:NSCONTROL START
```

Multiple HP 3000/XL Environment: Time for a decision

When the time came to add the HP922/XL to our LAN environment, transforming us to a multi XL environment, a decision had to be made to either buy an additional DTC to accommodate the necessary serial connections, limit the new machine to just ETHERNET-based connections (no dumb terminals), have all HP922 accessors *DSLINE* from the HP960, or migrate our Host-based DTC management to PC-based DTC management.

Migrating to PC-Based DTC Management

Our existing two DTC 48s connected to our HP960 were just staring to feel some relief due to the increasing number of ETHERNET accessors. Although we were no longer "maxing out" our DTC connections, we could not afford to dedicate one of them to just the HP922. After pricing an additional DTC, minimally configured, we decided the best alternative to provide for future growth would be to migrate to PC-based DTC management. This involved purchasing the *HP OpenView DTC Manager* software along with an HP Vectra *QS/165* (HP will only support a *Vectra, IBM or COMPAQ*).

The PC-based DTCMGR allowed us to share the existing two DTC 48s, as well as establishing the beginning of our Network Management workstation environment (we also purchased *HP OpenView HUB Manager* and *HP OpenView Bridge Manager*).

The actual conversion and *NMMGR* changes were easy and straightforward. Only *Nailed Devices* need to be defined on the host side (printers and/or terminals that will need to accessed programmatically by specific LDEV numbers — a must for some two screen debuggers). The PC DTC configuration involves creating a graphic representation of the backplane of each DTC - an exercise in *point and click.*

We have found the *HP OpenView DTC Manager* to be extremely reliable and flexible.

Network Review: Novell file server (FS1)
Novell file server (FS2)
HP, IBM, and VAX hosts
140 workstations
Ethernet coax cable
Internal bridge connecting FS1 and FS
Fiber multiplexer (RS323 support)
Fiber bridge
Ethertwist 10Base-T cable

Reference Exhibit D
Reference Exhibit E

Attaining our current position did not take place overnight or without planning. We started building our network in 1988. As technology trends continue to change we will adapt or modify our data communication corporate strategy to take advantage of new data communication technology. The benefits have certainly been with the effort and cost.

All product names are trademarks or registered trademarks of their respective companies.

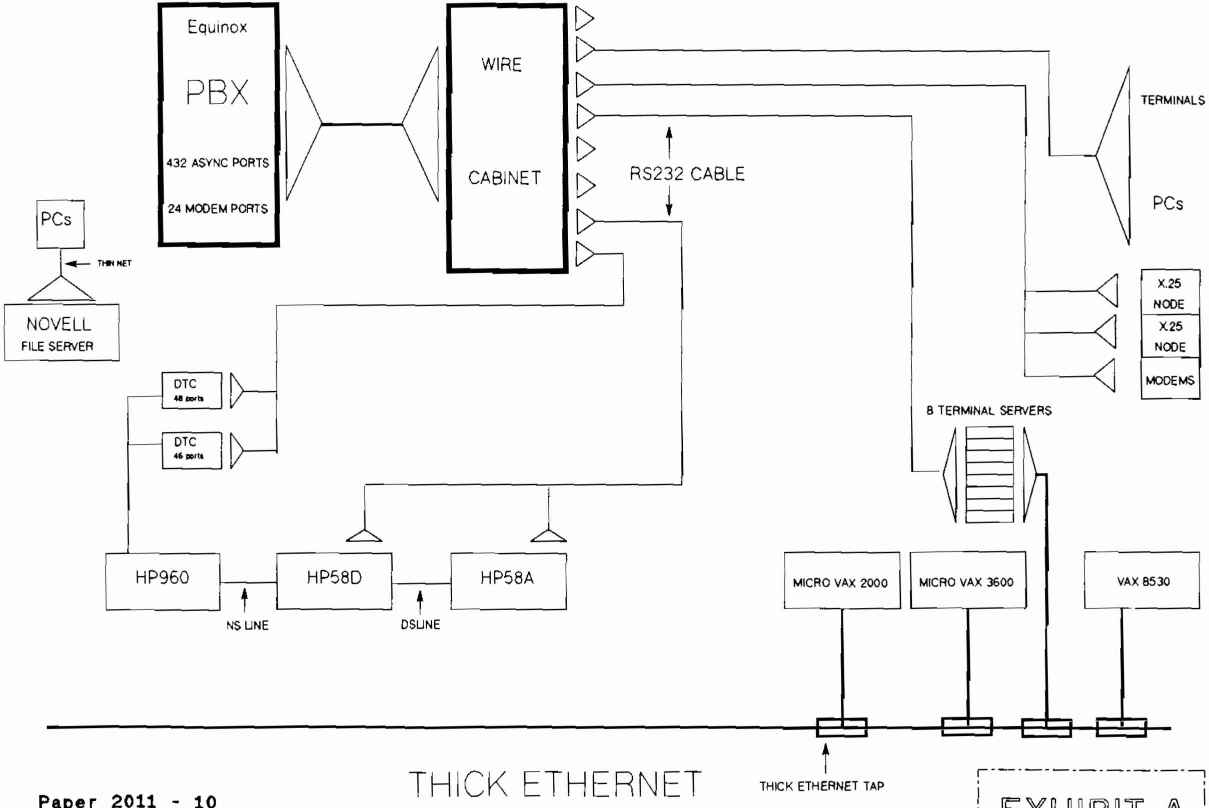
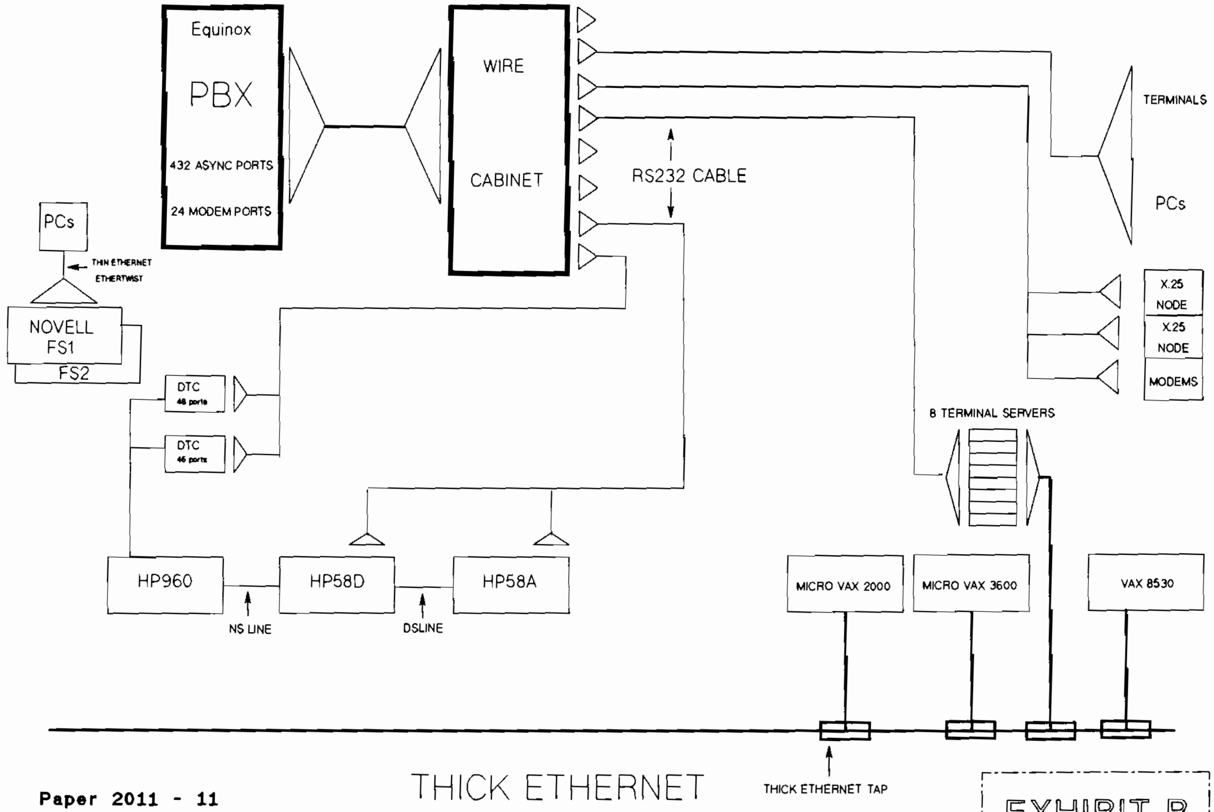

Equinox
PBX
432 ASYNC PORTS
24 MODEM PORTS
WIRE
CABINET
RS232 CABLE
TERMINALS
PCs
X.25 NODE
X.25 NODE
MODEMS
8 TERMINAL SERVERS
PCs
THIN NET
NOVELL
FILE SERVER
DTC
48 ports
DTC
46 ports
HP960
HP58D
HP58A
NS LINE
DSLINE
MICRO VAX 2000
MICRO VAX 3600
VAX 8530
THICK ETHERNET
THICK ETHERNET TAP

EXHIBIT A

EXHIBIT B


Equinox
PBX
432 ASYNC PORTS
24 MODEM PORTS
WIRE
CABINET
RS232 CABLE
TERMINALS
PCs
X.25
NODE
X.25
NODE
MODEMS
8 TERMINAL SERVERS
PCs
THIN ETHERNET
ETHERTWIST
NOVELL
FS1
FS2
DTC
48 ports
DTC
46 ports
HP960
HP58D
HP58A
NS LINE
DSLINE
MICRO VAX 2000
MICRO VAX 3600
VAX 8530
THICK ETHERNET
THICK ETHERNET TAP

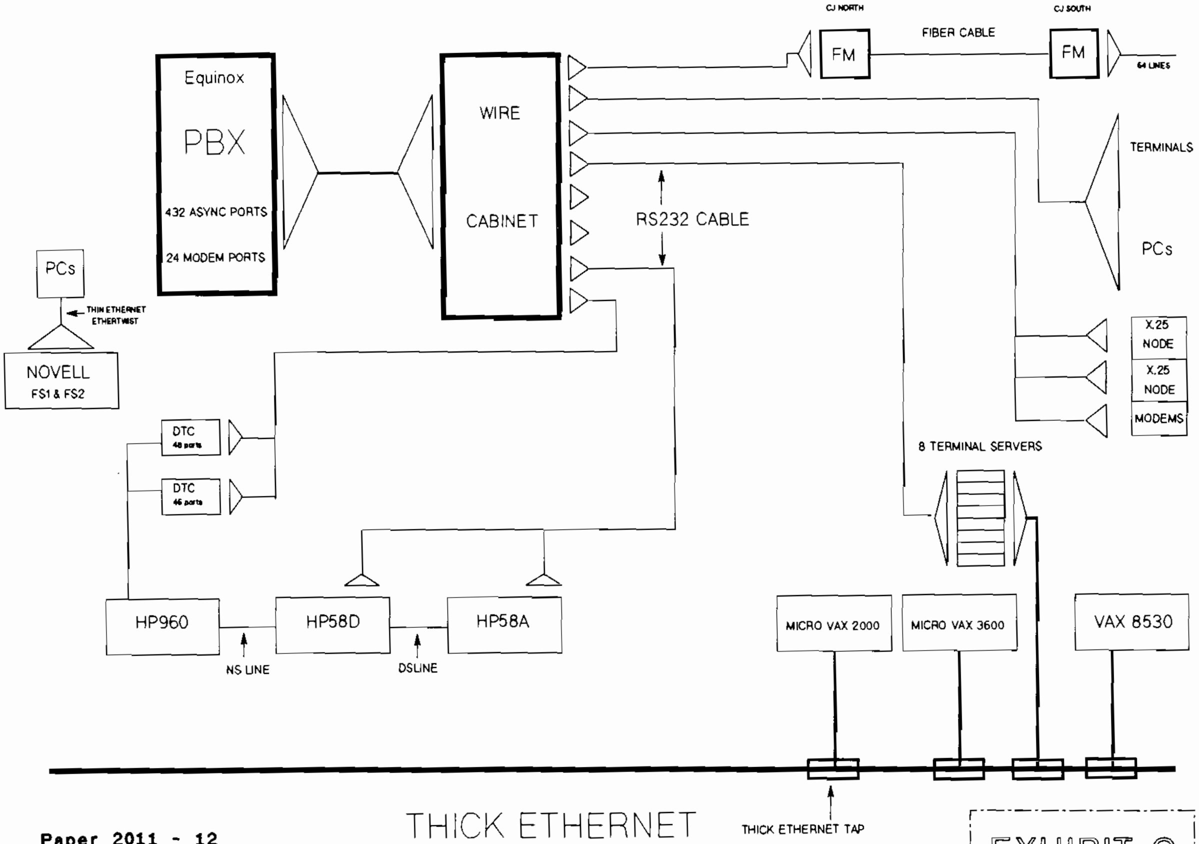

CJ NORTH
CJ SOUTH
FIBER CABLE
FM
FM
64 LINES
Equinox
PBX
432 ASYNC PORTS
24 MODEM PORTS
WIRE
CABINET
RS232 CABLE
TERMINALS
PCs
X.25 NODE
X.25 NODE
MODEMS
8 TERMINAL SERVERS
PCs
THIN ETHERNET ETHERTWIST
NOVELL
FS1 & FS2
DTC
48 ports
DTC
46 ports
HP960
HP58D
HP58A
NS LINE
DSLINE
MICRO VAX 2000
MICRO VAX 3600
VAX 8530
THICK ETHERNET
THICK ETHERNET TAP

EXHIBIT C

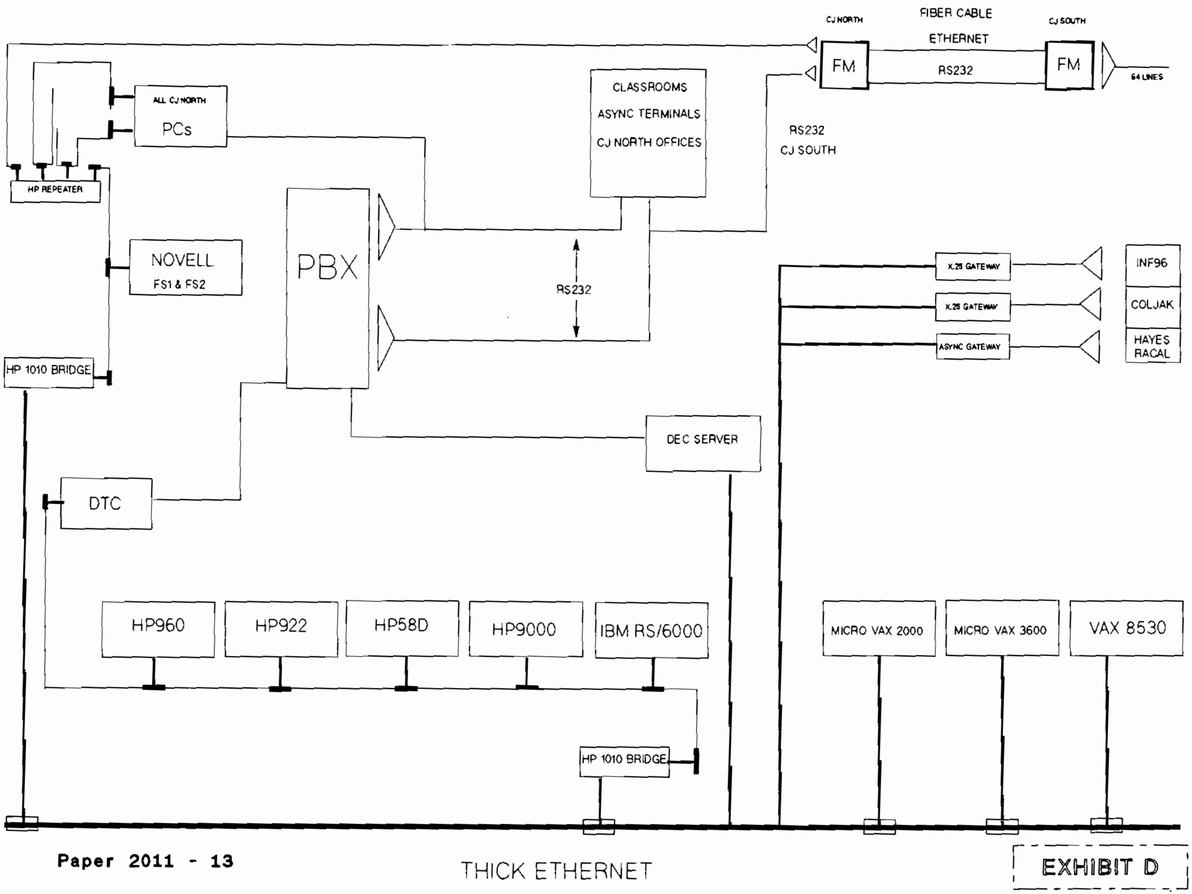
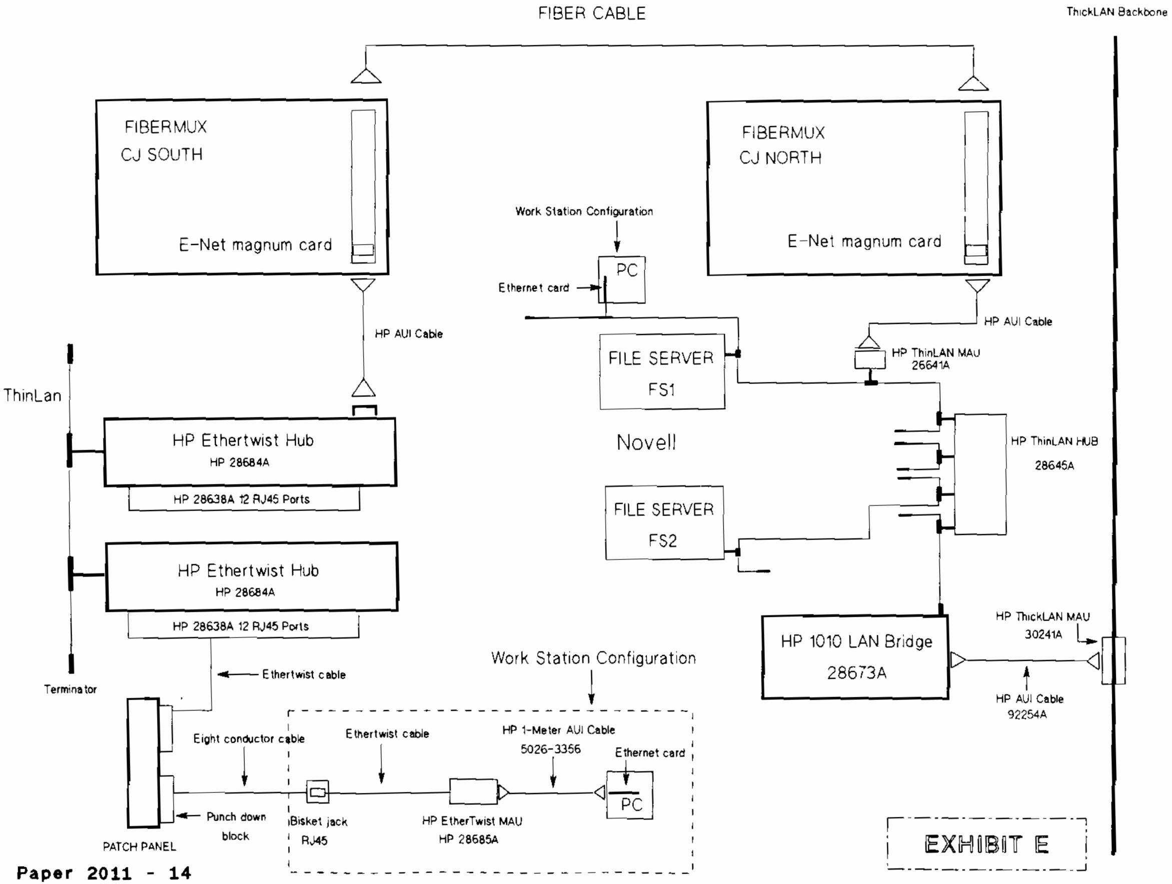

CJ NORTH
FIBER CABLE
ETHERNET
CJ SOUTH
FM
RS232
FM
64 LINES
ALL CJ NORTH
PCs
CLASSROOMS
ASYNC TERMINALS
CJ NORTH OFFICES
RS232
CJ SOUTH
HP REPEATER
NOVELL
FS1 & FS2
PBX
RS232
X.25 GATEWAY
X.25 GATEWAY
ASYNC GATEWAY
INF96
COLJAK
HAYES
RACAL
HP 1010 BRIDGE
DEC SERVER
DTC
HP960
HP922
HP58D
HP9000
IBM RS/6000
MICRO VAX 2000
MICRO VAX 3600
VAX 8530
HP 1010 BRIDGE
THICK ETHERNET

EXHIBIT D

EXHIBIT E


FIBER CABLE
ThickLAN Backbone
FIBERMUX
CJ SOUTH
E-Net magnum card
FIBERMUX
CJ NORTH
E-Net magnum card
Work Station Configuration
PC
Ethernet card
HP AUI Cable
HP AUI Cable
HP ThinLAN MAU
26641A
FILE SERVER
FS1
Novell
FILE SERVER
FS2
HP ThinLAN HUB
28645A
ThinLan
HP Ethertwist Hub
HP 28684A
HP 28638A 12 RJ45 Ports
HP Ethertwist Hub
HP 28684A
HP 28638A 12 RJ45 Ports
Terminator
Ethertwist cable
Work Station Configuration
HP ThickLAN MAU
30241A
HP 1010 LAN Bridge
28673A
HP AUI Cable
92254A
Eight conductor cable
Ethertwist cable
HP 1-Meter AUI Cable
5026-3356
Ethernet card
PC
Punch down
block
Bisket jack
RJ45
HP EtherTwist MAU
HP 28685A
PATCH PANEL

PAPER NO.: 2012

TITLE: Writing UNIX-like Commands in the Bourne Shell

AUTHOR: Craig P. Albrecht
Cray Research, Inc.
1440 Northland Drive #3224
Mendora Heights, MN 55120
(612) 683-7299

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

PAPER NO.:	2013
TITLE:	An Inside Look at the UNIX Command NAWK-- An Interactive Investigation
AUTHOR:	Craig P. Albrecht Cray Research, Inc. 1440 Northland Drive #3224 Mendora Heights, MN 55120 (612) 683-7281

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

Paper #2015
HP Client-Server Configurations

Gregory Proulx
Oracle Corporation
400 Oracle Parkway Redwood Shores, CA 94065
(415) 506-6162, (206) 637-3214

Introduction

The concept of the decentralization of computing power into a client-server configuration has been gaining in popularity in recent years, and it is not difficult to see why. Some compelling reasons for moving to a client-server solution include the realization of true distributed processing to match today's decentralized workplace, the leveraging of existing investment by taking advantage of idle PC-desktop CPU cycles while simultaneously offloading host systems (which translates into more users), and the over-riding cost of ownership benefits of the client-server solution in most cases when compared to centralized one. An added benefit is the ability to utilize some of the many graphical user interfaces (GUI's) available on PC's and workstations.

As with anything, however, there are some drawbacks to a client-server implementation which also must be considered. These become most obvious when viewed from an operational point-of-view. For example, the introduction of client-server into environment could potentially cause applications to appear more complex from a user's perspective, the reduction in processing on the host system may in turn translate into a greater load on the network, and other operational issues such as security, software distribution, performance, and reliability will necessitate more planning on the part of the MIS staff.

Fortunately, these drawbacks can be addressed and overcome with the proper planning, design, and implementation. This paper will attempt to assist in the planning effort by focusing on the various issues associated with a client-server configuration, and discuss some possible solutions to help ensure a successful implementation. For purposes of illustration, Oracle software will be utilized as the example for some of the concepts discussed in this paper.

Client-Server Architecture Overview

There are three basic components of a client-server architecture: clients, servers, and networks. Each of these will be discussed in greater detail, but at this point it is sufficient to state that all three components must be present in any client-server configuration. Figure 1 provides a high-level diagram of a client-server configuration. Note that for each component, there is both a hardware and software requirement.

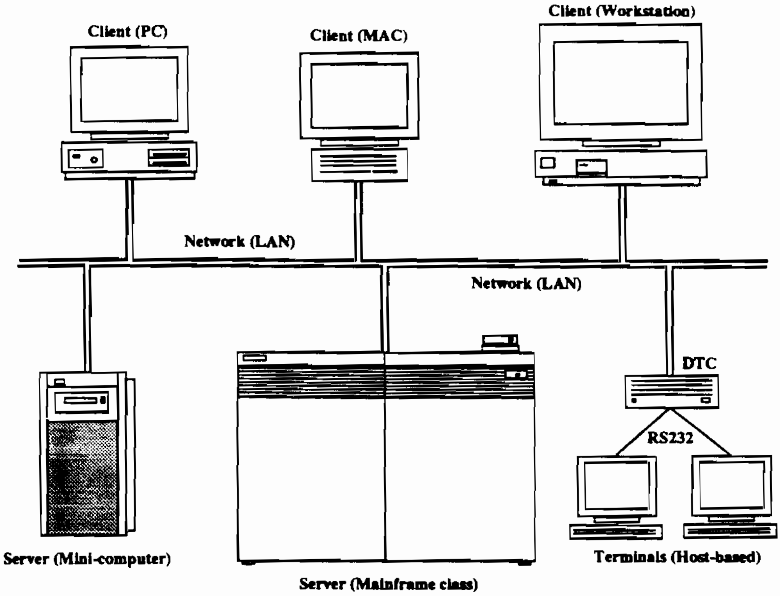


Figure 1. Client-Server Environment

Clients have traditionally been viewed as desktop PC's or Macintosh's, although this does not preclude the use of workstations, mini-computers, or even mainframes. Similarly, although servers are usually considered to be of mini-computer or mainframe class, workstations and PC servers are also filling this role today. The network portion of the client-server architecture can seem the most complex, with a wide array of choices for networking protocols and application programmatic interfaces (API's), various network topologies, and gateways. Like the client and server components, networks consist of both hardware (the physical connection), and software (the protocol communications driver), with common networking software typically residing on both the client and the server.

It is important to note that in planning for a flexible and open client-server implementation, the choice of software is playing an increasingly dominant role, as more and more hardware vendors are providing platforms that adhere to standards and the generally recognized guidelines for 'open systems'. One such example of this move towards 'open systems' is HP's support for the Ethernet standard on the HP3000, as well as the POSIX interface in MPE/XL, which has been renamed MPE/iX.

One should also consider that with almost any move towards a client-server configuration, there will be a transitional period where some users will remain in a host-based operating mode. This 'mixed-mode' may consist of users with terminals, PC's acting as terminals through emulation software, and PC's or Macintosh's operating in true client-server mode. During this transitional period, which may take place over a period of weeks or even years, any combination of the above scenarios may be occurring. This underscores the importance of choosing a software solution that will smooth the transition between the three operating modes, and make it appear as transparent as possible to the users, while leveraging the time and effort of the MIS development staff.

Client-Server Architecture Overview

As previously stated, each component of a client-server architecture has both a hardware and software element. Figure 2 illustrates these corresponding elements, using Oracle's client-server architecture as an example.

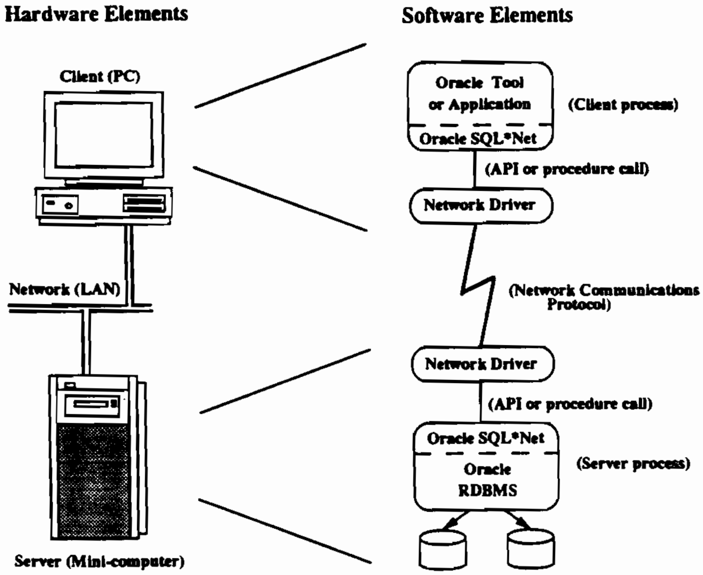


Figure 2. Client-Server Architecture

On the client side, the end-user is represented by the client process, which in this case is either an Oracle tool or application, together with the SQL*Net layer. The SQL*Net software is an API-like component which abstracts out the underlying physical connection and communications protocol used. The benefit of this approach is that multiple networking protocols can be utilized with SQL*Net without affecting the application program or tool. SQL*Net will pass requests for data to the client's network protocol driver, usually via a procedure call, which in turn will send the request across the physical network to the server to be processed.

On the server side, a listener process will intercept the client's request for data, by way of the corresponding network protocol driver residing on the server. This software driver will in turn pass the request through to the SQL*Net software, which will route the request to the Oracle RDBMS to actually retrieve the data from the database files residing on the server disks.

Client-Server Configurations

There are many possible client-server configurations, and a discussion of all permutations is beyond the scope of this paper. Consequently, the following examples will focus on common configurations utilizing Hewlett-Packard clients and servers, together with an industry-standard network consisting of Ethernet and the TCP/IP protocol.

TCP/IP is a publicly-developed, non-proprietary communication protocol, and a de-facto standard that is widely used for linking computers supplied by different vendors. TCP/IP is also the most extensively used networking protocol in the HP systems environment today. The TCP/IP protocol operates in a process-to-process mode. This means that when a client system requests a remote connection to a server system, the server responds by initiating a process to establish a connection and handle the pending and subsequent requests. In doing so, the server process remains independent of the communication tasks, and exists for the duration of the remote connection. Utilizing TCP/IP and Oracle together is just one of the possible solutions for implementing distributed database processing in a client-server configuration.

In order to be able to process requests for remote connections, the server first must be initialized in the following manner: a) the SERVICES file must exist and contain the TCP socket address (Oracle recommends 1525), b) the ORATAB file must exist and contain a list of valid Oracle SID's and their corresponding locations (group.account), c) the TCP/IP network driver (Network Services) must be up and running, and d) the Oracle SQL*Net listener process (ORASRV) must be up and running.

Figure 3 details the steps involved in the initiation of a client-server connection using the Oracle tools with the TCP/IP networking protocol, once the above initialization has been done. In step (1), the client side makes a remote request for a connection to the server. This request is intercepted by the ORASRV listener process in step (2), which verifies that the database system identified (SID) and location are valid, and then establishes a connection. The ORASRV process then creates the server (or shadow) process in step (3), and then passes the previously established connection from the client to the server in step (4), at which point they begin communicating directly. Finally, in step (5), the ORASRV process is free to await the next request for a remote connection.

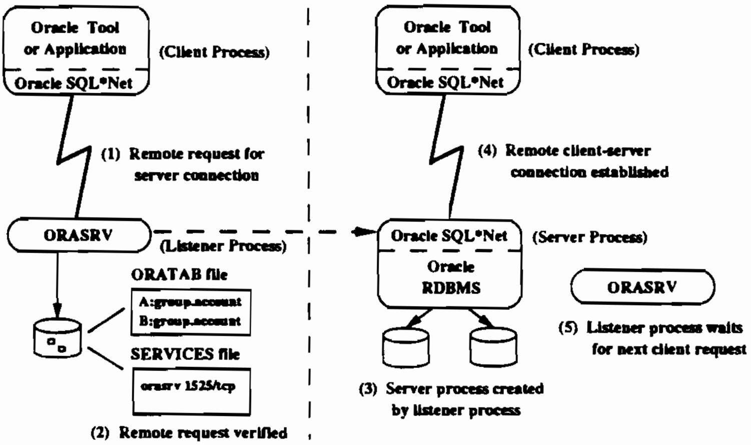
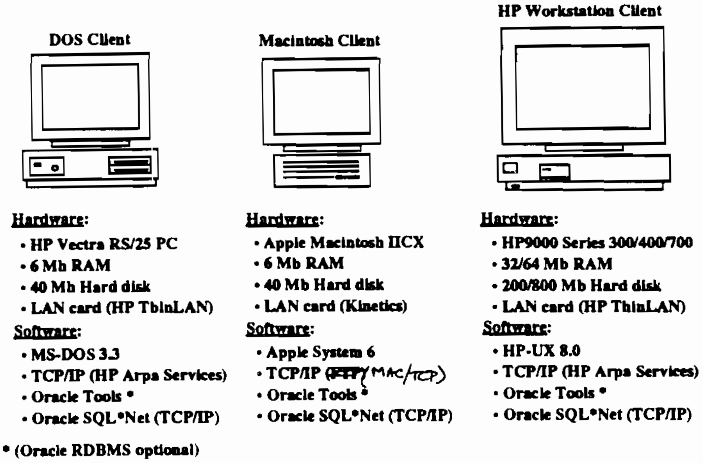


Figure 3. Client-Server Initialization

As mentioned previously, each client component consists of both a hardware and software element. Figure 4-1 shows some sample client configurations, including an HP Vectra PC and an HP-UX workstation. Notice that there are both hardware (the LAN card) and software components (TCP/IP, Oracle SQL*Net) of the network present on the client side. Although the applications, in this case the Oracle tools, are required on the client, the data (and it's repository) are not, although this is an option. Finally, please note these configurations are provided for example purposes only, and are not meant to be all-inclusive.



DOS Client

Hardware:
- HP Vectra RS/25 PC
- 6 Mb RAM
- 40 Mb Hard disk
- LAN card (HP ThinLAN)

Software:
- MS-DOS 3.3
- TCP/IP (HP Arpa Services)
- Oracle Tools *
- Oracle SQL*Net (TCP/IP)

Macintosh Client

Hardware:
- Apple Macintosh IICX
- 6 Mb RAM
- 40 Mb Hard disk
- LAN card (Kinetics)

Software:
- Apple System 6
- TCP/IP ~~(FTP)~~ (MAC/TCP)
- Oracle Tools *
- Oracle SQL*Net (TCP/IP)

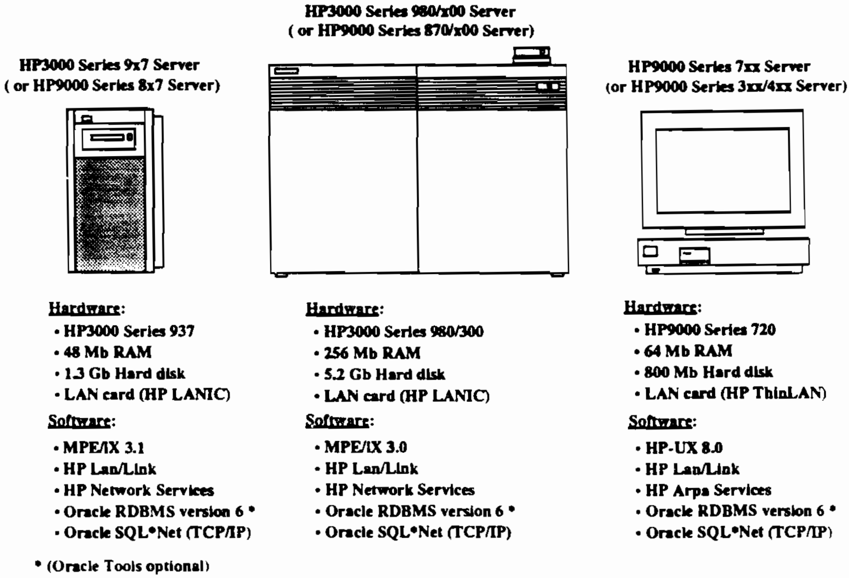
HP Workstation Client

Hardware:
- HP9000 Series 300/400/700
- 32/64 Mb RAM
- 200/800 Mb Hard disk
- LAN card (HP ThinLAN)

Software:
- HP-UX 8.0
- TCP/IP (HP Arpa Services)
- Oracle Tools *
- Oracle SQL*Net (TCP/IP)

* (Oracle RDBMS optional)

Figure 4-1. Sample Client Configurations



HP3000 Series 9x7 Server
(or HP9000 Series 8x7 Server)
HP3000 Series 980/x00 Server
(or HP9000 Series 870/x00 Server)
HP9000 Series 7xx Server
(or HP9000 Series 3xx/4xx Server)

HP3000 Series 9x7 Server (or HP9000 Series 8x7 Server)

Hardware:
- HP3000 Series 937
- 48 Mb RAM
- 1.3 Gb Hard disk
- LAN card (HP LANIC)

Software:
- MPE/iX 3.1
- HP Lan/Link
- HP Network Services
- Oracle RDBMS version 6 *
- Oracle SQL*Net (TCP/IP)

HP3000 Series 980/x00 Server (or HP9000 Series 870/x00 Server)

Hardware:
- HP3000 Series 980/300
- 256 Mb RAM
- 5.2 Gb Hard disk
- LAN card (HP LANIC)

Software:
- MPE/iX 3.0
- HP Lan/Link
- HP Network Services
- Oracle RDBMS version 6 *
- Oracle SQL*Net (TCP/IP)

HP9000 Series 7xx Server (or HP9000 Series 3xx/4xx Server)

Hardware:
- HP9000 Series 720
- 64 Mb RAM
- 800 Mb Hard disk
- LAN card (HP ThinLAN)

Software:
- HP-UX 8.0
- HP Lan/Link
- HP Arpa Services
- Oracle RDBMS version 6 *
- Oracle SQL*Net (TCP/IP)

* (Oracle Tools optional)

Figure 4-2. Sample Server Configurations

On the server side, we have a similar model. Figure 4-2 highlights some sample server configurations again utilizing HP systems, in this case a low-end HP3000 mini-computer, a high-end HP3000 SMP system, and a mid-range HP-UX workstation. Like the client side, we see that there are both hardware (the LANIC card) and software (HP Network Services, Oracle SQL*Net) components of the network on the server side. The physical connection between the clients and servers in this example would be ThinLAN Ethernet cable.

Since the data will reside on the server, the repository, in this case the Oracle RDBMS, must also reside on the server. However, here the applications or Oracle tools are optional, and would only be required if it were desirable to run them in a host-based mode using terminals or PC's with terminal emulation software. This might be the case during the transition phase or 'mixed-mode' environment mentioned earlier.

Integrating other Relational and Non-Relational DBMS's

One major benefit of a client-server implementation is the ability to perform distributed database operations, such as queries across multiple server nodes. In the example of Oracle, this is accomplished through the use of a 'database link', or logical reference to the remote database(s). An example of the creation and use of a database link is the following:

Example 1 - Accessing remote Oracle databases.

```
create public database link boston using 'T:BOST3000:A';

select * from employees@boston;
```

where 'boston' is the name of the link, and in the *using* clause, 'T' represents the TCP/IP protocol, 'BOST3000' the node name of the server, and 'A' is the Oracle SID. Since no username or password is supplied, the user's local login will be passed to the remote database. To reference the data on the remote machine, the user would specify the database link name following the '@'. Location transparency can be provided through the use of a database synonym.

In addition, it may be desirable to access existing data on a server node that is stored in a non-relational repository, such as TurboIMAGE. In Oracle this can be accomplished through the SQL*Connect gateway, which provides read access into TurboIMAGE databases. This allows the heterogeneous integration of the TurboIMAGE DBMS as though it were simply one more Oracle node on the network. Figure 5 shows a high-level diagram of the SQL*Connect to TurboIMAGE architecture in a client-server configuration. In this case, the server process is the SQL*Connect to TurboIMAGE server, which communicates with both the Oracle RDBMS and TurboIMAGE DBMS simultaneously, via procedure calls. The Oracle RDBMS instance on the server is referred to as the 'supporting node', and is utilized to enforce security and access to the TurboIMAGE databases, and to store information about user-customized translation rules.

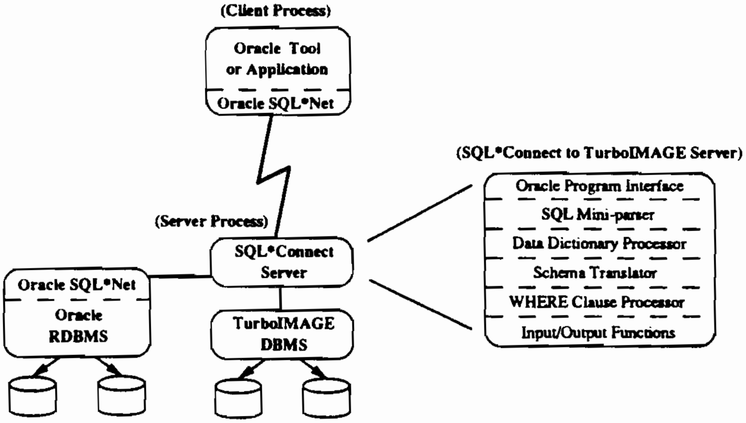


Figure 5. SQL*Connect to TurboIMAGE in Client-Server

The method for accessing a TurboIMAGE database is similar to that of a remote Oracle node, namely, through the use of a database link. An example of the creation and reference of a SQL*Connect database link is as follows:

*Example 2 - Accessing remote TurboIMAGE databases via SQL*Connect.*

```
create database link turbo_boston
connect to scott identified by tiger
using 'T:BOST3000:A(EMPDB.TURBO.DBASE/MANAGER)';
```

where 'turbo_boston' is the name of the link, and in the *using* clause, 'T' represents the TCP/IP protocol, 'BOST3000' the node name of the server, and 'A' is the Oracle SID. The remainder of the connect string refers to the TurboIMAGE database, where 'EMPDB' is the name of the root file, followed by it's group and account, and 'MANAGER' is a password defined in the TurboIMAGE database. The *connect to* clause specifies the username and password to connect to on the Oracle supporting node, where 'SCOTT' is the username, and 'TIGER' is the password. It is through the combination of the Oracle username/password together with the TurboIMAGE password that security is enforced against the TurboIMAGE database. As in the previous example, to reference the TurboIMAGE data on the remote machine, the user would specify the database link name following the '@'. Location transparency can again be provided through the use of a database synonym.

Operational Issues

Although there may be tremendous benefits in moving to a client-server configuration, these must be weighed against the ability of the MIS staff to manage and administer such an implementation. One of the first considerations will be the decision of where to locate the applications and data. In a distributed processing environment, the location of the software tools and applications is independent of where the data is stored. Some important factors in making this decision include: what is the level of utilization (and response time) of the current hardware choices, both now and in the future; what is the availability of network hardware and software to connect together the various systems being considered; what systems are most convenient for the primary users of the application to access; and what is the ease of developing or maintaining application software on a given system, and distributing it to the client systems. For example, how easy (or difficult) is it to distribute a newer version of an application to all of the client systems? Is there a method for automatically distributing these revisions/upgrades across the network rather than relying on the client end-users in the remote sites to perform this task? A final consideration is the ability to effectively manage a distributed network, and there are various tools available on the market (from HP and others) for handling this task.

In a distributed database configuration, DBA's have flexibility in determining where to store data, and this decision in turn affects data security and access, application performance, and reliability. The issue of *security* can be addressed in several ways, including through the definition of database links and the ORATAB file. Referring back to example 1, database links can be created as either *public* or *private*, and can optionally include the *connect to* string. The DBA can choose to control access through the definition of multiple links, with a variety of username/passwords each having different levels of Oracle capabilities. Recall also that the ORATAB file will contain a list of valid Oracle databases on a server that can be accessed remotely. Inclusion of a database instance in this list precludes the capability for remote access. These security issues and methods apply equally when accessing TurboIMAGE databases, but take on an added dimension since security must be planned for and enforced in both Oracle and TurboIMAGE. For example, in addition to specifying the TurboIMAGE password in the database link statement, it is also possible to alter the TurboIMAGE database open mode (default = 5) for the user specified in the *connect to* clause to also affect access.

When considering *performance*, there are many aspects which may have an affect. In designing an optimal distributed database and processing scheme, DBA's should consider at least the following questions: where is the data located currently; where will new data be inserted from primarily; are the applications of a query-based or update-based nature; do the applications perform a lot of screen-handling, or are they more of a block-mode nature. For example, an application written in SQL*Forms may be more suited to a client-server configuration than one written in a 3GL using block-mode, since all of the screen processing overhead will be off-loaded to the front-end. Similarly, an application that is of an update nature will benefit more from a client-server configuration than one that is query-based, since all of the processing of the user's input will be handled by the client. In addition, the MIS staff should consider what the current load is on the network; and what is the expected mix of application operating modes on the system during the transitional phase.

Reliability is another important issue that takes on increased importance in a distributed processing environment. Some of the issues and questions to be asked include: how reliable is the networking hardware and software; how reliable are the client and server systems to be utilized; what transaction models will be defined for distributed transactions (e.g. short, long, across multiple DBMS's); how will backup and recovery be handled on both the client and systems. For example, in the case of a distributed, multi-node transaction that performs an update, each node must be up and running in order to complete the transaction. What happens if one of the nodes goes down in the middle -- are there adequate automated recovery mechanisms on all of the nodes to properly handle this situation? In a similar vein, is there an ability to perform an on-line backup on the server systems to avoid down-time for all the clients? From the support perspective, will end-users be responsible for backing up their own client systems (e.g. PC's)?

The final topic to be discussed in the area of operations is the planning for MIS systems development. In planning for a client-server implementation, this will most likely involve a transitional period. The ability of the MIS development staff to effectively manage this transition will be based in large part on the portability of the solution utilized. For example, how difficult is it to create an application on any of the systems in the network (e.g. the one with the least system utilization at a particular point in time), and not be concerned about which systems it will ultimately be deployed on? Will an application that is developed on an HP3000, for example, be portable to other systems in the network, for example PC's or Macintosh's? Is it possible for the same identical application (e.g. binary) to execute in all of the transitional or mixed-mode environments? For example, could you run the same application in client-server using a PC, in terminal emulation using a Macinstosh, and in a host-based mode utilizing terminals?

Another concern in this mixed-mode environment is that of end-user training and portable user interfaces. Is it possible to deploy the same application user interface and keyboard layouts across all of the platforms? Does the software solution utilized provide the ability to have data transparency as well as location transparency, to shield the end-users from unnecessary complexity in the client-server configuration? If the answer is no to these questions, then the MIS staff will have an additional burden of retraining and supporting the end-users as they move from environment to environment during the transition phase.

Conclusions

Although there are some obstacles to implementing a client-server environment that are not present in the traditional centralized, host-based environment, these can be overcome with the proper planning up front. In deploying a client-server configuration, the burden will be upon the MIS staff to design and implement applications that address all the issues associated with performance, reliability, security, application portability, and ease of use. For the present time and foreseeable future, the most dominant factor in a successful implementation will be the proper choice of a software solution. With the proper planning, the added benefits of GUIs, distributed processing, flexibility, performance, and the cost effectiveness of a client-server implementation can be realized.

PAPER NO.: 2017

TITLE: Understanding IP Addressing

AUTHOR: J. W. Swearingen
The Apex Group, Inc.
7151 Columbia Gateway Drive
Columbia, MD 21046
(301) 290-1606

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

PAPER NO.:	2018
TITLE:	Convert That Old Application to X

AUTHOR:

James Langan
J.B. Langan & Associates, Inc.
2800 Fish Hatchery Road
Madison WI 53711
(608) 836-8008

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

PAPER NO.: 2019

TITLE: Connect Those Computers

AUTHOR: James Langan
J.B. Langan & Associates, Inc.
2800 Fish Hatchery Road
Madison, WI 53711
(608) 273-0428

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

Paper # 2020

How to Migrate Forms to X11/Motif

Bob Combs
Combs International, Inc.
886 Belmont Ave. Suite #3
North Haledon, NJ 07508
201-427-9292

Abstract

Full screen forms have been an accepted, even expected, method of user interaction for some years now. It is commonly accepted that forms and windows are the way to provide user friendly features. X11 with Motif widgets are currently the most commonly accepted manner of designing these forms under UNIX systems. But the learning curve for X and Motif is steep. Therefore, let's step through the migration process to convert a form from a serial terminal to an X/Motif environment.

1. Introduction

Migrating a form application from an ASCII terminal to an X terminal requires a fair amount of X knowledge. Since the learning curve on X is steep, this paper will only cover enough of X11R4 and Motif 1.1 to migrate a basic form. It really isn't possible to cover in one paper what normally takes at least six months to learn.

The approach will be to describe a form, identifying its component parts, introduce X concepts, describe the X elements our form will need, and to put them all together. Throughout this process we'll review what code will be needed to drive the form.

2. Form Description

Form applications have been in use for over a decade, closer to two decades. An RS-232C ASCII "intelligent" terminal which can handle forms has two main features over "dumb" terminals; protected mode and block reads. The protected mode lets an application set areas that can be typed into (i.e. fields) and areas that are protected from the user. A block read provides the mechanism that allows the computer to request or user to send all of the unprotected data in one transmit block. The unprotected fields only are sent, and are typically separated by some special character so that the program receiving the data can distinguish one field from the next.

We'll refer to these RS-232C ASCII forms as just ASCII forms to differentiate from X forms.

2.1. Screen

The entire screen serves as the "window" for an ASCII form. Fields and labels may be placed anywhere on the screen.

2.2. Fields

A field is a contiguous unprotected text area on the screen. It can be anywhere on the screen and cannot be longer than one line. That is, a field is terminated at the end of a line if not otherwise terminated. The field can be described as the unprotected area between two protected areas. Generally, we think of the entire screen as protected, and the unprotected islands as specific fields.

2.3. Labels

Most forms have a label before or after a field to identify its contents. A label is just text in a protected area of the screen. The computer can output the labels, but the user cannot access this area of the screen. Labels are usually part of the form's background template.

2.4. Controls

A form application must provide a way for the user to signal when the form is complete and should be read into the computer. This control may be a special character such as control-U, or special character sequence. Many form applications use a softkey. The softkey generates special escape sequences which the computer can detect. For example, on an HP terminal F1 sends <esc> <p> <cr>. Since softkeys can have and often do have labels, the softkey escape sequence is predefined to have a particular meaning such as "Enter", "Next", "Previous", "Help", or "Abort". This allows the user to request various actions on the screen of data they are viewing.

3. Form Routines

There are several basic functions which must be performed in any forms application to make the form truly useful. These functions provide the ability to display the form, write the field values, read the field values, erase the form, and wait for an operator request. These functions must be accomplished whether the form is ASCII or X.

Displaying the form is probably the most complex function, since it must build the field areas, labels, and any additional screen geometry, colors, etc. Screen background templates may be built previously by some editor, created on-the-fly at execution time, or some hybrid combination of these two. Either way, the display form function will place the form template onto the screen.

A user request routine usually waits for the user to press some particular key sequence, usually a softkey, and returns a value indicating which key or command was requested. In the case of X, event handling will take place in this routine, making it more complex than its ASCII counterpart.

Writing and reading field values is generally performed against specific fields. Due to the nature of ASCII terminals, the entire screen of fields are generally read or written in one block. This requires write screen and read screen routines which transfer the screen fields into or out of a local field buffer. The individual write field and read field routines would then only operate against this in memory buffer. Note that if the terminal can perform field specific addressing, the buffer and buffer routines are not needed. Each field under X is a separate widget, so field values may only be accessed one at a time.

Erasing the screen implies that the terminal will be placed back in character mode after the screen is erased.

Therefore, the following set of screen functions can be defined:

`display`	`display the form`
`fieldwrite`	`write a field value`
`fieldread`	`read a field value`
`getrequest`	`get a user operation request`
`erase`	`erase form`

4. X Introduction

There are several layers to X which will be used for our form. X library is the protocol control layer and closest connection to the system. The X layer routines usually begin with 'X', X Toolkit intrinsics usually begin with 'Xt', and Motif routines usually begin with 'Xm'. Figure 1 shows the general relationship between an X/Motif program and the various layers of the X system. Most of the calls are to Motif routines, with some calls to the Xt toolkit. Most of the calls to X are made by the Motif and Xt routines. X routines are fairly low level, and usually called only to perform functions which neither Motif nor Xt can perform.

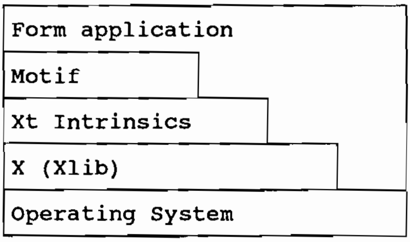


figure 1.

4.1. Client-Server Model

X programs come in pairs that communicate across TCP/IP (or possibly some other communications medium). There is an X protocol that defines the type and format of transfers that can occur between the pair. The user terminal, since it is capable of handling multiple programs (windows), is called the X server. The driving programs which run on the UNIX machine are referred to as the clients. Refer to figure 2. Initially this naming seems backwards because of our exposure to file servers, but when viewed in the X model it makes sense. An X server (terminal) may be running multiple X clients, which may be executing from the same or different UNIX machines.

There is a window manager to generally regulate the placement of the windows on the screen, control the focus, and give the windows the manager's look by controlling the frame's look. The Motif window manager is called "mwm". The look mwm gives to the frame is the 3-D look which is associated with Motif. Other window managers such as Open Look give the frames and windows a different look. The window which is currently accepting input is said to have the focus. There is only one window and one widget which are actively focused upon at any time. Usually the active window has its frame highlighted.

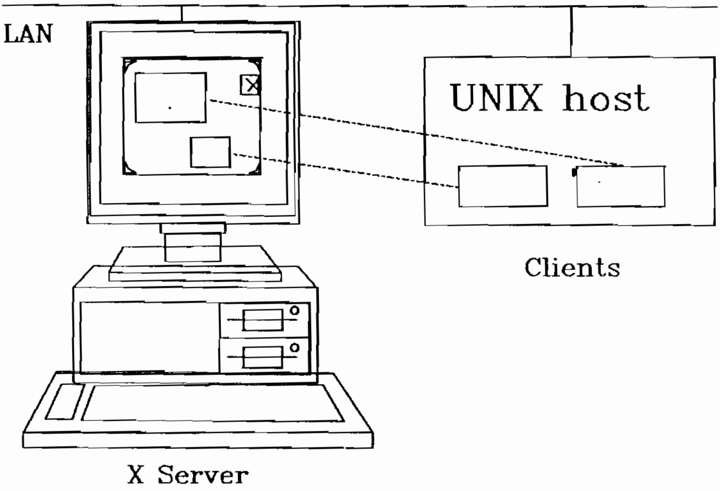


figure 2.

4.2. Widgets

The word *widget* has historically been a slang word meaning "thing". Generally used when one doesn't know what to call an object. Well, the X folks didn't quite know what to call the function elemental pieces of X windows, so they used the word widget. Its somewhat difficult to describe a widget. If it were easy they'd have a better name!

Each widget is a window or subwindow. Widgets may be attached to other widgets. For example, labels are widgets that are attached to a background widget window. Each field is also a widget. So are buttons, menu bars, pull-down panels, and option selections. The frames, colors, shadow thickness, etc. are not widgets; they are resources.

4.3. Gadgets

If we have to have widgets, we have to have gadgets too! A gadget is a special reduced functionality form of a widget. Gadgets consume less resources and provide somewhat less functionality. Labels, for example, can be either widgets or gadgets. When a label is a gadget, it cannot set its own colors as it can if used as a widget. Motif supplies both widget and gadget calls where useful. Background windows or text fields don't have gadget versions; there isn't any use for a gadget version.

4.4. Resources

Resources are the characteristics of a window or widget. They include items such as foreground color, background color, x & y position, active or inactive shadow colors, frame line thicknesses, etc. including even the text of a field, button, etc. Resources are inherited down the window tree. That is, the window manager (mwm) passes its settings to the application window, and the application passes its settings to the widgets spawned under it.

The default resources may come from an applications default file (like /usr/lib/X11/app-defaults/Example), from the command line runstring ($ bb -bg Yellow), or be set directly by the application program. Those resources which are not set are either inherited or null. Figure 3 shows a partial inheritance tree for the widgets used in this paper.

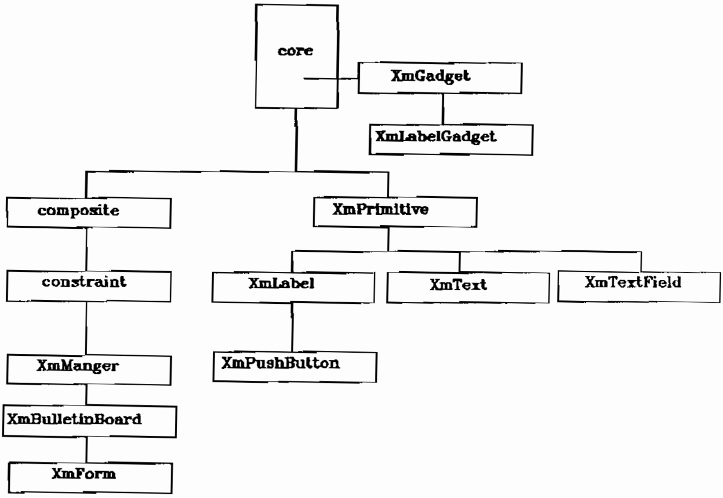


figure 3.

5. X Programming

X programming implicitly assumes a firm knowledge of C, X usage, and X terminology. The documentation which must be read to master X/Motif programming is staggering. However, we will cover just enough of the basics to get a form working. There are many embellishments which one may wish to add to their own form application. Those will be left to each individual to learn later.

5.1. Window

A form will need a window on the X server's display. It is on this background window that all the additional form widgets will be attached. There are two basic widgets that may serve for the form background; Form and BulletinBoard widgets. Of the two, I have chosen the BulletinBoard widget. The reason I did so has to do with the attachment rules. A Form widget allows attaching additional children widgets to its sides, top, or bottom. A BulletinBoard widget allows attaching of children widgets directly to its background using an XY coordinate.

```
#ident "@(#)bb.c - bulletin board example"

#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <Xm/Xm.h>
#include <Xm/BulletinB.h>


/**********************************************************************
 *      main logic
 **********************************************************************/
void main (argc, argv)
unsigned int argc;
char **argv;
{
        XtAppContext app_context;
        Arg args [20];
        Widget bb;
        int n;
        XmString string;
        Widget toplevel;


/*      Initialize toolkit and create bulletin board */
        toplevel = XtVaAppInitialize (&app_context, "Example", NULL, 0,
                                        &argc, argv, NULL,NULL);

        string = XmStringCreate("bulletin board example",
                        XmSTRING_DEFAULT_CHARSET);
        n = 0;
        XtSetArg(args[n], XmNdialogTitle, string);              n++;
        XtSetArg(args[n], XmNwidth,  200);                      n++;
        XtSetArg(args[n], XmNheight, 150);                      n++;
        XtSetArg(args[n], XmNresizePolicy, XmRESIZE_GROW);      n++;
        bb = XmCreateBulletinBoard(toplevel, "bboard", (ArgList) args, n);
        XmStringFree(string);
        XtManageChild(bb);


/* Get and dispatch events */

        XtRealizeWidget(toplevel);

        XtAppMainLoop(app_context);
}
```

figure 4.

Figure 4 is our sample program to create and display a solitary BulletinBoard widget window. The include files are required for X intrinsics, X strings, Motif, and the BulletinBoard widget. XtVaAppInitialize performs several X functions; connect to the X display, parse any command line arguments, load resources from defaults file etc., and create the top level parent window which will handle all interaction with mwm.

The XtSetArg calls define resource values which are used for the BulletinBoard widget when it is created by the XmCreateBulletinBoard call. The XmCreateString is used to define a string which is set as the title (by XtSetArg).

The memory which XmCreateString allocates is released by XtFree after the widget was created.

When a widget is created, it is said to be "instantiated". This is another word which the X folks made up. It means that the widget structures have been defined, but not necessarily displayed. Displaying is referred to as "realizing".

After the widget is created, it must be given over to management by X. This is performed by the XtManageChild call.

Once all the widgets have been created and managed, they may be "realized" using the XtRealizeWidget call. This is called for the toplevel shell widget only. All children of the toplevel are then realized.

Finally, XtAppMainLoop is called. This is the event processing loop. The program waits in this call for events such as mouse movement, button clicks, and keyboard key presses. Some events are related to actions on other windows, such as exposures.

This program is a full, be it small, operational X/Motif program.

5.2. Labels

Labels are created using the Label widget. Figure 5 shows the code required to add a label.

```
#include <Xm/Label.h>

        . . .

        Widget lbl1;

        . . .

/*      Add a label */
        string = XmStringCreate("Label-1", XmSTRING_DEFAULT_CHARSET);
        n = 0;
        XtSetArg(args[n], XmNx, 10);                              n++;
        XtSetArg(args[n], XmNy, 17);                              n++;
        XtSetArg(args[n], XmNlabelString, string);                n++;
        lbl1 = XmCreateLabel(bb, "label1", args, n);
        XmStringFree(string);
        XtManageChild(lbl1);

        . . .
```

figure 5.

The text string which will be the label is converted to X usable format with the XmStringCreate call, and then its memory released by the XtFree call later. The resources XmNx and XmNy are set to the xy location of the label, as the XmNlabelString is set to the X label text. XmCreateLabel instantiates the label widget.

Widget resources may be set by passing the arguments into a widget creation call or invoked separately using the XtSetValues call. Figure 6. shows an alternative call sequence to figure 5. Both work equally well, although figure 5. may be more efficient.

```
/*      Add a label */
        lbl1 = XmCreateLabel(bb, "label1", args, n);

        string = XmStringCreate("Label-1", XmSTRING_DEFAULT_CHARSET);
        n = 0;
        XtSetArg(args[n], XmNx, 10);                                         n++;
        XtSetArg(args[n], XmNy, 17);                                         n++;
        XtSetArg(args[n], XmNlabelString, string);                           n++;
        XtSetValues(args, n);

        XmStringFree(string);
        XtManageChild(lbl1);
```

figure 6.

5.3. Fields

Field widgets are created just like Label widgets. See figure 7.

```
#include <Xm/TextF.h>

        . . .

        Widget fld1;

        . . .

/*      Add a field */
        n = 0;
        XtSetArg(args[n], XmNx, 60);                                         n++;
        XtSetArg(args[n], XmNy, 5);                                          n++;
        XtSetArg(args[n], XmNcolumns, 12);                                   n++;
        fld1 = XmCreateTextField(bb, "field1", args, n);
        XtManageChild(fld1);

        . . .
```

figure 7.

The resource XmNcolumns defines how many characters long the field is when displayed.

5.4. Buttons

A button must be created similarly to the Label and Field widgets, and must also have its action routine defined. Figure 8 shows the code to define Button widgets.

```
#include <Xm/PushB.h>

        . . .

        Widget wa[7];
        int wc=0;

        . . .

/*      Add push buttons */
        string = XmStringCreate("Enter", XmSTRING_DEFAULT_CHARSET);
        n = 0;
        XtSetArg(args[n], XmNx, 10);                                         n++;
        XtSetArg(args[n], XmNy, 100);                                        n++;
        XtSetArg(args[n], XmNlabelString, string);                           n++;
        wa[wc] = XmCreatePushButton(bb, "btn1", args, n);
```

```
XmStringFree(string);

wc++;

string = XmStringCreate("Init", XmSTRING_DEFAULT_CHARSET);
n = 0;
XtSetArg(args[n], XmNx, 70);                                   n++;
XtSetArg(args[n], XmNy, 100);                                  n++;
XtSetArg(args[n], XmNlabelString, string);                     n++;
wa[wc] = XmCreatePushButton(bb, "btn2", args, n);
XmStringFree(string);

wc++;


string = XmStringCreate("Exit", XmSTRING_DEFAULT_CHARSET);
n = 0;
XtSetArg(args[n], XmNx, 140);                                  n++;
XtSetArg(args[n], XmNy, 100);                                  n++;
XtSetArg(args[n], XmNlabelString, string);                     n++;
wa[wc] = XmCreatePushButton(bb, "btn3", args, n);
XmStringFree(string);

wc++;

XtManageChildren(wa,wc);

. . .
```

figure 8.

Three buttons are created in figure 8, each with a different label and at a different position. The buttons will appear on the screen with the labels "Enter", "Init", and "Exit". Another programing concept introduced here is the usage of widget arrays (wa). It is more efficient under X to create multiple widgets, placing them into a widget array, and then managing them all at once with the XtManageChildren call.

5.6. Callbacks

What good are the Buttons created in figure 8 if no event action is defined for each? An action can be defined for an event by defining a CallBack. A CallBack is a routine that will be executed by X when the described event occurs. The routine performs the action desired for that event.

There are many different types of events, and each widget has a set of events it can invoke through callbacks. However, for tutorial purposes, we will only define a button callback since that is the minimum required to make the form work.

```
/*       Add callbacks for each button press */
         XtAddCallback(wa[4], XmNactivateCallback, enterCB, "Enter button");
         XtAddCallback(wa[5], XmNactivateCallback, initCB, fld[0]);
         XtAddCallback(wa[6], XmNactivateCallback, exitCB, NULL);
```

figure 9.

After the Button widgets have been created, as in figure 9, callbacks are defined for each. In each case the event type is XmNactivateCallback. This means the event is activated by clicking the mouse while pointing to that button. There is a separate routine defined for each callback. The fourth parameter is a parameter which is passed to the event when called. The keyword NULL means zero;

nothing is passed. The string "Enter button" is passed to the enterCB routine. The field widget pointer fld[0] is passed to the initCB routine. Nothing is passed to the exitCB routine. Refer to figure 10.

```
/***************************************************************
 * Enter CallBack
 ***************************************************************/
void enterCB(w,str,ca_d)
Widget w;
char *str;
caddr_t ca_d;
{
        char *s;
        int i;

        printf("%s was pressed\n",str);

        for (i=0; i<2; i++) {
                s = XmTextFieldGetString(fld[i]);
                printf("  Field %d = %s\n",i,s);
                XtFree(s);
        }
}

/***************************************************************
 * Init CallBack
 ***************************************************************/
void initCB{w, wf, ca_d)
Widget w;
Widget wf;
caddr_t ca_d;
{
        char *s;

        s = XmTextFieldGetString(wf);
        printf("Field-1 was \"%s\", setting field to \"default\".\n",s);
        XmTextFieldSetString(wf, "default");
        XtFree(s);
}

/***************************************************************
 * Exit CallBack
 ***************************************************************/
void exitCB(w,str,ca_d)
Widget w;
char *str;
caddr_t ca_d;
{
        exit(1);
}
```

figure 10.

In the enterCB routine, the passed in string is printed and then the value of each field is fetched and printed too. Items which are printed to standard output are displayed in the window which executed the X program. Since many X programs are invoked in the background it isn't a good idea to use the routine printf as we have here.

The InitCB routine fetches the field text from the passed in widget, displays what it was, and then initialized it to a value of "default".

The exitCB routine terminates the X program. Note that without a button to trigger termination, the window would have to be closed from the X frame function which is available on each mwm window frame. Not providing an exit function is crude and sloppy.

5.7. Putting It All Together

Figure 11 puts all the pieces of our simple form example together.

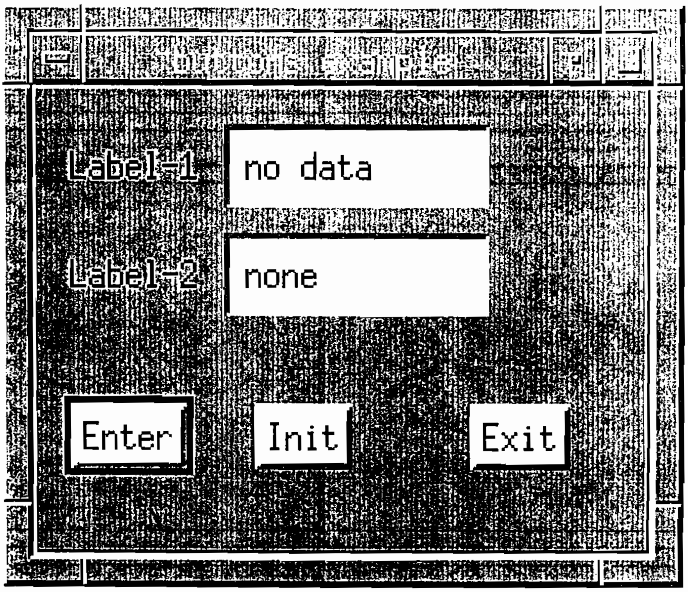
```
#ident "@(#)form.c - form example"

#include <stdio.h>
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <Xm/Xm.h>
#include <Xm/BulletinB.h>
#include <Xm/Label.h>
#include <Xm/TextF.h>
#include <Xm/PushB.h>

Widget fld[2];           /* global field widget array */

/****************************************************************
 * Enter CallBack
 ****************************************************************/
void enterCB(w,str,ca_d)
Widget w;
char *str;
caddr_t ca_d;
{
        char *s;
        int i;

        printf("%s was pressed\n",str);

        for (i=0; i<2; i++) {
                s = XmTextFieldGetString(fld[i]);
                printf("  Field %d = %s\n",i,s);
                XtFree(s);
        }
}

/****************************************************************
 * Init CallBack
 ****************************************************************/
void initCB(w, wf, ca_d)
Widget w;
Widget wf;
caddr_t ca_d;
{
        char *s;

        s = XmTextFieldGetString(wf);
        printf("Field-1 was \"%s\", setting field to \"default\".\n",s);
        XmTextFieldSetString(wf, "default");
        XtFree(s);
}

/****************************************************************
 * Exit CallBack
 ****************************************************************/
void exitCB(w,str,ca_d)
Widget w;
char *str;
caddr_t ca_d;
{
        exit(1);
}

/*********************************************************************
 *      main logic
 *********************************************************************/
void main (argc, argv)
```

```
unsigned int argc;
char **argv;
{
        XtAppContext app_context;
        Arg args [20];
        Widget bb, wa[7];
        int n;
        XmString string;
        Widget toplevel;
        int wc;


/*      Initialize toolkit and create bulletin board */
        toplevel = XtVaAppInitialize (&app_context, "Example", NULL, 0,
                                          &argc, argv, NULL,NULL);

        string = XmStringCreate("form example",
                                 XmSTRING_DEFAULT_CHARSET);
        n = 0;
        XtSetArg(args[n], XmNdialogTitle, string);                 n++;
        XtSetArg(args[n], XmNwidth,  200);                         n++;
        XtSetArg(args[n], XmNheight, 150);                         n++;
        XtSetArg(args[n], XmNresizePolicy, XmRESIZE_GROW);         n++;
        bb = XmCreateBulletinBoard(toplevel, "bboard", (ArgList) args, n);
        XmStringFree(string);
        XtManageChild(bb);

/*      Add labels */
        wc = 0;

        string = XmStringCreate("Label-1", XmSTRING_DEFAULT_CHARSET);
        n = 0;
        XtSetArg(args[n], XmNx, 10);                               n++;
        XtSetArg(args[n], XmNy, 17);                               n++;
        XtSetArg(args[n], XmNlabelString, string);                 n++;
        wa[wc] = XmCreateLabel(bb, "label1", args, n);
        XmStringFree(string);

        wc++;

        string = XmStringCreate("Label-2", XmSTRING_DEFAULT_CHARSET);
        n = 0;
        XtSetArg(args[n], XmNx, 1D);                               n++;
        XtSetArg(args[n], XmNy, 52);                               n++;
        XtSetArg(args[n], XmNlabelString, string);                 n++;
        wa[wc] = XmCreateLabel(bb, "label2", args, n);
        XmStringFree(string);

        wc++;

/*      Add fields */
        n = 0;
        XtSetArg(args[n], XmNx, 60);                               n++;
        XtSetArg(args[n], XmNy, 5);                                n++;
        XtSetArg(args[n], XmNcolumns, 12);                         n++;
        XtSetArg(args[n], XmNvalue, "no data");                    n++;
        wa[wc] = XmCreateTextField(bb, "field1", args, n);
        fld[0] = wa[wc];

        wc++;

        n = 0;
        XtSetArg(args[n], XmNx, 60);                               n++;
        XtSetArg(args[n], XmNy, 45);                               n++;
        XtSetArg(args[n], XmNcolumns, 12);                         n++;
        XtSetArg(args[n], XmNvalue, "none");                       n++;
        wa[wc] = XmCreateTextField(bb, "field2", args, n);
        fld[1] = wa[wc];

        wc++;

/*      Add push buttons */
```

```
        string = XmStringCreate("Enter", XmSTRING_DEFAULT_CHARSET);
        n = 0;
        XtSetArg(args[n], XmNx, 10);                                n++;
        XtSetArg(args[n], XmNy, 100);                               n++;
        XtSetArg(args[n], XmNlabelString, string);                  n++;
        wa[wc] = XmCreatePushButton(bb, "btn1", args, n);
        XmStringFree(string);

        wc++;

        string = XmStringCreate("Init", XmSTRING_DEFAULT_CHARSET);
        n = 0;
        XtSetArg(args[n], XmNx, 70);                                n++;
        XtSetArg(args[n], XmNy, 100);                               n++;
        XtSetArg(args[n], XmNlabelString, string);                  n++;
        wa[wc] = XmCreatePushButton(bb, "btn2", args, n);
        XmStringFree(string);

        wc++;


        string = XmStringCreate("Exit", XmSTRING_DEFAULT_CHARSET);
        n = 0;
        XtSetArg(args[n], XmNx, 140);                               n++;
        XtSetArg(args[n], XmNy, 100);                               n++;
        XtSetArg(args[n], XmNlabelString, string);                  n++;
        wa[wc] = XmCreatePushButton(bb, "btn3", args, n);
        XmStringFree(string);

        wc++;


        XtManageChildren(wa,wc);

/*      Add callbacks for each button press */
        XtAddCallback(wa[4], XmNactivateCallback, enterCB, "Enter button");
        XtAddCallback(wa[5], XmNactivateCallback, initCB, fld[0]);
        XtAddCallback(wa[6], XmNactivateCallback, exitCB, NULL);

/*      Get and dispatch events */

        XtRealizeWidget(toplevel);

        XtAppMainLoop(app_context);
}
```

figure 11.

The window that figure 11 generates is figure 12.



Label-1 no data

Label-2 none

figure 12.

6. X form routine functionality

Coding each form in X is still quite a bit of work. Our company solved the problem by writing an X forms generator that takes care of all the X calls. The form calls were made compatible with the ASCII form calls which everyone had been using. Our X forms generator is currently sold as the product **Xfrm**. Referring to the X programming we have used above, it can be divided into the original routines defined for our simple forms manipulation.

6.1. display

The **display** function must define the background screen as it does in ASCII forms. Under X, **display** should initialize the display, create the bulletin board window, create each of the widgets, define any callbacks which are needed, call to manage the widgets, and realize the widgets. This is most of the X program, but it isn't all of it.

6.2. getrequest

The **getrequest** function contains the event and dispatch processing (XtAppMainLoop). The buttons must set a value which can be returned to the caller to indicate which button was pressed.

This can be accomplished several different ways. One way is to use XtAppNextEvent and XtDispatchEvent instead of the XtAppMainLoop call. This expansion allows testing the external button_pressed variable. Each button's callback must set a different value into the button_pressed variable which **getrequest** will see and exit back to the caller, returning the button number pressed. Figure 13 contains the **getrequest** code.

```
int getrequest()
{
        XEvent event;

        while (!button_pressed) {
                XtAppNextEvent (app_context, &event);
                XtDispatchEvent (&event);
        }
        return (button_pressed);
}
```

figure 13.

6.3. fieldwrite & fieldread

Text field values are read or written using the **fieldwrite** and **fieldread** functions as access to XmTextSetString and XmTextGetString calls. Any numeric to text conversion can be performed by the field routines before or after calling the respective XmText routine. Don't forget to XtFree any XmString the routines create.

6.4. erase

The **erase** function would usually just exit the program. But it may be required to only end the window and allow the program to continue executing, possibly to create another window. In this case, **erase** will need to call XtDestroyWidget for the toplevel widget. This recursively destroys all children of toplevel. Note that callbacks for the destroyed widgets are also destroyed, but any allocated memory is not returned. This must be done explicitly with XtFree calls.

References:

X Window System (2nd Edition) by Robert W. Scheifler & James Gettys; Digital Press; 1990

OSF/Motif Programmer's Reference (Release 1.1) by Open Software Foundation; Prentice Hall; 1991

OSF/Motif Programmer's Guide (Release 1.1) by Open Software Foundation; Prentice Hall; 1991

Motif Programming Manual (Volume Six) by Dan Heller; O'Reilly & Associates, Inc; 1991

Is there a Workstation in your Future?

Interex '92 Paper #2021

by

Brian Nogar

Hewlett-Packard Company
540 Officenter Place
Columbus, Ohio 43230-5321
(614) 478-6282
bnogar@a2614uxa.msr.hp.com

Power! Speed! Elegance! The allure of the UNIX workstation abounds in the industry today. You can not pick up a trade magazine without reading about some new advance in workstation hardware or software technology. As you watch these advances in technology continuing to accelerate, it is hard not to wonder what impact the workstation could have in your HP3000 environment.

The most important question to ask before rushing out and ordering a bunch of HP9000 Series 700's is,

> *"What benefits will my organization realize with the addition of these workstations?"*

Understanding the capabilities of the workstation and how it can integrate into your existing environment is a necessary step before the expected benefits can be determined. This paper will help build this understanding by examining the capabilities of the UNIX workstation as a Client in the HP3000 environment.

Evolution of Computing Architectures

The applicability of the UNIX workstation in an existing environment is going to be directly related to the type of computing architecture currently in place. These architectures range from batch orientated mainframes to the emerging Client/Server architecture of the 90's. Outlined below is a brief discussion concerning the evolution of computing architectures.

- In the 1960's the most prevalent computing architecture available was very large batch oriented systems. These machines were designed primarily to perform manipulations on large amounts of data, with little or no on-line activity. Typically, these systems were room sized machines with considerable environmental and staffing requirements.
- The 1970's saw the evolution of on-line interactive computing. Led by the development of small 'mini-computers', computing power was moved closer to the users by providing on-line access to smaller distributed systems. These machines typically had much lower staffing and environmental requirements than mainframes and created the era of distributed computing.
- The 1980's saw the personal computer revolutionize the definition of computing, by putting computer technology on millions of desks throughout the world. The PC brought the average user a closeness to the computer never before achieved, allowing them to satisfy many of their information requirements while never leaving their desks. The PC was used both as a personal productivity tool and as a window of access to the information housed on the corporate computing systems.
- With the 1990's comes the era of Client/Server computing. Client/Server computing consists of an information architecture in which multiple processes work together to satisfy the information requirements. The application workload is typically shared between two or more machines with each component performing the piece of the application they are most capably able to supply. Typically the client system provides and manages the user interface and requests services from one or more server systems attached to the network.

Of the computing architectures defined above, the UNIX workstation is going to find its largest degree of acceptance in the emerging Client/Server arena. In fact, the advancement of the workstation is one of the technologies driving the advent of Client/Server computing.

What's a Client to do?

In order to determine how the workstation will best fit into and existing environment, it is necessary to set adequate expectations concerning what a client system is expected to do. The list below discusses some of the key capabilities which a Client platform should possess.

1. The client must support a Graphical User Interface (GUI). Although the importance of GUI's was open for some debate in the late 1980's, there seems little disagreement that they are the mandatory interface for the 1990's and beyond. A consistent GUI provides significant benefits in the areas of ease of use, user training, data accuracy, consistency among applications, and user productivity and is a must requirement for any client platform being evaluated.
2. The client must support local and/or wide area communication capability. The chosen client architecture must be able to seamlessly communicate with a variety of host systems providing information services. This communication should be accomplished in a method that is transparent to the end user.
3. The client should support multi-tasking. The ability to work on more than one task at a time is a definite requirement in order to provide true client/server functionality.
4. The client ideally will possess local processing power. The ability to run applications or pieces of applications can provide significant benefits in the areas of response time and application functionality.
5. The client should have a wide range of application software readily available. As the chosen client will most likely be the user's sole window into the information infrastructure, it should be capable of meeting personal productivity computing needs.

Before anointing the UNIX workstation as the client platform of choice, it should be noted that their are other contenders for the crown. Currently the MS-DOS and OS/2 Operating systems on the Intel platform provide very capable alternatives, while down the road Microsoft's Windows NT may well provide the most significant competition. All of these offerings provide technology leadership in one area or another, and there is no single best solution. By understanding the strengths and weaknesses of each offering and how to integrate them with existing architectures, the best overall solution to address a particular business need can be determined.

MS-DOS, OS/2, or UNIX? Windows 3.1, Presentation Manager/Workplace, Motif, or OpenLook? 80X86, PA-RISC, SPARC, MIPS, RIOS, or ALPHA?

One thing is certain, lack of choice is not a problem facing the information systems architect of today. In fact, with technology advances exploding at an accelerating rate, the contrary is often true. It is difficult to make an architectural platform decision due to the wealth of choices available today.

The discussion below provides a brief description of the capabilities of each particular client operating environment, along with an evaluation of how each environment matches up against the list of desired client capabilities.

MS-DOS

With millions of units installed and hundreds of thousands of software applications available, this platform has been a phenomenal success in both the business and home environment. It is hard to find a personal application need that doesn't have some kind of answer in the MS-DOS world. For personal productivity tools such as Word Processing and Spreadsheets, MS-DOS is a platform that is hard to beat.

GUI	Yes	MS-Windows
Network	Yes	LanManager, Novell, Arpa Svcs Not integrated in O.S.
Application Availability	Yes	Very wide range of choice
Multi-tasking	No	Only non-preemptive multi-tasking
Local Processing	Yes	Limited by multi-tasking capability of MS-DOS.

Let's now examine how MS-DOS meets our requirements for a client platform. The Microsoft Windows environment has become widely accepted and is close to becoming the standard GUI on the Intel platform. MS-Windows provides a wide range of available applications with a strong industry acceptance. From a communications point of view, the two most popular LAN connections available are LanManager and Novell. Although, both of these options provide connectivity to the HP3000, it should be noted that neither are tightly integrated into the operating system.

A drawback that has emerged in the client/server environment is that the hardware capabilities of the personal computer have begun to outstrip the capabilities of the MS-DOS operating system. The 640K memory limitation of MS-DOS can be very restrictive when running multiple applications in a GUI environment or across a Local Area Network. Unfortunately, the design of

MS-DOS is not optimized to take advantage of the turbocharged hardware on the market today. These limitations have spurred the development of alternative operating systems such as OS/2 and Microsoft Windows NT, which were both developed to eliminate the current deficiencies found in MS-DOS. The development of client/server applications which require both a GUI and a Local Area Network connection will be impeded by these MS-DOS limitations.

In short MS-DOS provides a client platform with some significant strengths. It is very widely installed, it is very cost effective, there is a tremendous availability of applications, and there is a large amount of MS-DOS expertise available in the industry. If you are choosing a personal productivity platform which only requires terminal based communication with a host, the MS-DOS platform is your best choice.

OS/2

The OS/2 Operating System was designed primarily to provide solutions to the shortcomings found in MS-DOS. The goal of OS/2 was to relieve the memory limitations of MS-DOS, to provide a true multi-tasking environment, and to provide integrated GUI and LAN services, while at the same time providing upward compatibility with MS-DOS applications.

The success or failure of any product can best be measured by its acceptance in the marketplace, and by these standards the early releases of OS/2 in the late 1980's were met with a resounding lack of success. The lack of application availability coupled with the cost of the engine required to run OS/2 resulted in an offering that the market avoided in droves. These objections are beginning to be erased by both the dramatic fall in the cost of hardware as well as the recent release of OS/2 version 2.0 from IBM. Time and the market will once again be the ultimate determining factor in the final measure of success which OS/2 receives.

GUI	Yes	Presentation Manager/Workplace
Network	Yes	LanManager Integrated with O.S.
Application Availability	Yes	Not as extensive as MS-DOS, but OS/2 2.0 is purported to run all MS-DOS and Windows 3.0 Apps
Multi-tasking	Yes	Provides True pre-emtive multi-tasking
Local Processing	Yes	Better than MS-DOS due to multi-tasking and memory availability.

Where OS/2 has achieved its greatest acceptance and where it exceeds the capabilities of MS-DOS are in custom written client/server applications developed by large 'Fortune 500' types of users. Typically these applications are entire systems developed to solve a specific business problem and consist of custom code on both the client and the server side. It is very common for these types of applications to be very critical to the business needs of the organization and require a significant investment in development time and effort.

UNIX

The UNIX Operating System has grown out of the engineering environment into the commercial data processing realm. UNIX supports both multi-tasking and multi-user capabilities, providing a range of installation sizes from single user workstations all the way to mainframes capable of supporting hundreds of users.

A look at how a UNIX client fares against our evaluation criteria is provided below:

GUI	Yes	Motif, OpenLook (X-Window-System based)
Network	Yes	TCP/IP ARPA/Berkeley Services integrated with O.S.
Application Availability	Yes	Low - Personal Productivity High - Application development
Multi-tasking	Yes	Provides True pre-emtive multi-tasking
Local Processing	Yes	Strong Local and Network processing capabilities

The graphical environment on the UNIX workstation is based on the X-Window System interface, and currently the two most prevalent GUI's available are OpenLook from Unix System Laboratories and Motif from the Open Software Foundation. One of the key benefits that X based interfaces have to offer over the MS-Windows and Presentation Manager environments is that networking is integrated in the design of the interface. This provides great flexibility in the ability to create and implement network based applications.

The overwhelming choice of connectivity in the UNIX environment is ARPA/Berkeley services based on the TCP/IP communication specification. As a defacto communication standard in the industry, this provides almost guaranteed connection to any UNIX server on the network. Additionally, many non UNIX systems (including the HP3000) also support these standards providing a very strong connectivity capability. An added bonus is that the hardware and software necessary for TCP/IP based communication is integrated up front with almost every workstation available.

In the area of application availability, UNIX has a mixed story to tell. From a personal productivity point of view, application availability does not come close to what MS-DOS has to offer. Although applications such as Lotus-123 and WordPerfect have found their way to the UNIX workstation, the overall choice is much more limited than in the MS-DOS environment. However, from a development environment perspective, the UNIX workstation offers advantages in the wide range of development tools available, the majority of which cross platform boundaries. This allows the workstation to be the development platform for both the client and the server components of the application, thereby freeing processing resources on the server while simplifying the testing and debugging phases of development.

Some of the key development tools available on the workstation platform include CASE tools such as Softbench, Software through Pictures, and Texas Instruments' IEF; and database development environments such as ORACLE, INGRES, INFORMIX, SYBASE, PROGRESS, COGNOS, and ALLBASE. These tools, along with other offerings to numerous to mention, provide a very strong cross platform development environment.

Which Client Should I Choose?

The final choice of the client platform ultimately revolves around three key areas:

1. The application need being addressed.
2. Personal preference of the end user.
3. The cost of the solution

While all three factors figure in the final decision, the ability of the client to solve the application need being addressed is probably the most important factor. Before personal preference, or cost can be evaluated you must insure that the selected client provides a solution to the business need.

It is very common today for an off the shelf application to be chosen independent of a particular hardware platform. This is a perfectly acceptable way to do business. The chosen application provides the technology necessary to answer the information processing requirements, while the hardware platform in which it runs is almost an afterthought. If the application is only available on one particular client platform, then the platform choice is made by default.

A tactic many companies are employing today is to standardize on a particular development platform. These tools, which include relational database management systems, fourth generation development tools, and CASE tools; define the base on which all applications will be built. The choice of development tools is a very important factor to consider when building an information architecture, as it can control your ability to take advantage of emerging hardware and software technologies.

Assuming an application has been selected which is able to run across multiple platforms, the decision now boils down to a tradeoff between client technologies. The preference of the end user probably has more impact in the client/server arena than in any other computing architecture. Since the advent of the personal computer, users have come to expect a great deal of choice in the way in which the applications function. The vast array of software has allowed users to find interfaces and applications with which they are familiar, comfortable, and productive. Once a user has a strong preference for a specific client architecture they probably will exhibit reluctance to move to another architecture.

The historical goal of information systems is to provide the user with the information they need to do their jobs. Client/server computing carries this goal a step further by providing the information in a presentation form with which the user is most comfortable, thereby increasing the users efficiency at the task. When one considers the importance that user acceptance has in the overall efficiency of the application, it is easy to see that user preference is a requirement which cannot be discounted.

The final area to be examined is cost. Unfortunately, it is hard to make a definitive statement about cost, outside the framework of a specific configuration. However, it can be said that at the current state of technology the personal computer platform will provide a lower cost entry point for most client systems. This lower cost is based on average configurations and does not take price/performance or functionality into consideration. The only way to make an accurate comparison is to price out the competitive configurations and evaluate the relative cost of each solution.

So, What do I choose?

By now it is clear that there is no single best solution which answers every alternative. All three platforms have areas where they exceed over the other two.

If the client technology is being evaluated to run personal productivity tools and provide terminal based access to a number of systems, then MS-DOS is the obvious solution. However, if a custom client/server application is being developed, the current memory and multi-tasking limitations of MS-DOS tend to limit its' long term effectiveness. In this scenario, it makes most sense to choose either OS/2, UNIX, or to wait for Microsoft Windows NT.

The choice for OS/2 is a scary one, because of the current level of acceptance it has received. Early indications about OS/2 2.0 indicate it is a strong offering with much better application support than previous OS/2 versions. The case for OS/2 will probably be decided in the next 12 months as the market either accepts or rejects this new offering. At this point, the safest approach would be to wait for OS/2 to receive wider acceptance before embarking on any long term OS/2 based client projects.

So, is there a workstation in your future? The answer is yes, if your future is one in which client/server computing can be found. The UNIX workstation can bring significant benefits to applications designed to take advantage of the performance, user interface, and networking capabilities that it provides. Applications built on fourth generation tools and designed to take advantage of the local processing power of the workstation are the best choices for this environment. As there is not a wide range of these types of applications currently available in the market, the workstation will typically find its' highest degree of acceptance in environments where new application development is taking place.

Integrating the workstation with your HP3000

There are two areas of integration with which the application developer should be concerned; network integration and application integration. The network integration defines the base level of communication in which the client uses to talk to the server, while the application integration deals with how application specific information is passed between the systems. A brief discussion of each area is provided below.

The base level of communication will be provided over a local area network running the TCP/IP protocol. This connection provides the framework for the various higher level communication protocols and services available.

- Network Services (NS): Network Services is a proprietary network product from Hewlett-Packard designed to provide basic communication services between the HP9000 and the HP3000. A virtual terminal connection (VT3K) is available to provide the workstation with VT connectivity to the HP3000, and file transfer operations are supported in either direction using DSCOPY. As the industry and Hewlett-Packard move toward defined network standards, the NS solution probably is not the best long term alternative.

- ARPA Services: This defacto industry standard defines virtual terminal (telnet), file transfer (FTP), and electronic mail communication methodologies (SMTP). At this time, both telnet and FTP are supported on the HP3000. The FTP solution simply requires a local area network interface card (LANIC), and the FTP software, to provide file transfer capability to any FTP supported system. Telnet on the other hand was implemented using the Distributed Terminal Controller (DTC). For a workstation user to be able to telnet to the HP3000 a "Telnet card" needs to be installed in the DTC. This card essentially converts from telnet protocol to AFCP protocol which is the standard HP3000-terminal communication protocol. Therefore, any telnet communication from the workstation to the HP3000 is routed through the DTC. Telnet support from the HP3000 to the workstation can also be accomplished via the DTC. However in the scenario no additional card is required, the DTC has built in communication capability to talk to any host that supports in bound telnet (which the workstation does).

- Network File System (NFS): NFS is an industry standard networking service that provides transparent access to remote file systems. Currently NFS support on the HP3000 allows workstation users (or users from any system supporting NFS) to transparently access files on the HP3000. NFS on the HP3000 is available from Quest software.

The services described above provide a base level of connectivity between the workstation and the HP3000 server. Services such as virtual terminal and file transfer provide relatively simple access to HP3000 based information, but supply nowhere near the capability necessary for true client/server applications.

The services described below provide connectivity options available for client/server applications being developed with third generation languages such as C and COBOL.

- NetIPC: NetIPC is the network inter-process control subset of Network Services. NetIPC provides the ability for processes on either system to communicate via the network. Remote procedure execution, passing of information, and process control are all features available in NetIPC.

- Berkeley Sockets: Berkeley sockets are the industry standard answer to NetIPC. Berkeley sockets provide the ability for processes on different machines to communicate and pass data between each other. At this point it is anticipated that Berkeley Sockets will be available on the HP3000 in the late 1992 time frame.

- HP Sockets: HP Sockets is a high level application interface which runs across a number of different communication protocols. It is designed to provide the application with a high level program to program communication method which is independent of the network. HP Sockets are currently supported on both the HP3000 and the HP9000, as well as a number of other hardware vendors equipment.
- OSF DCE: The Open Software Foundation's Distributed Computing Environment is an integrated set of services that supports the development, use and maintenance of distributed applications. One of the base features of OSF DCE is the Network Computing System from Hewlett-Packard. The Remote Procedure Call (RPC) capabilities of NCS make the resources of the network transparently available to the user via remote execution of client procedures on servers throughout the network. OSF DCE is targeted to be the premier client/server development environment of the 1990's and will be fully supported on the HP3000 in 1993.

These components provide the framework for developing client/server applications using program-to-program communication in the third generation development environment. Although still a popular alternative third generation language development is being augmented and even replaced by fourth generation tools based on relational data base management systems.

The greatest degree of client/server effectiveness will be realized by developing applications based on a relational data base management system. The workstation provides a very powerful engine in which to place CPU extensive data manipulation and presentation resources. The HP3000 on the other hand utilizes intelligent server processes to select the requested data from the corporate database and send it to the workstation. Relational database management systems provide significant capabilities in the area of data selection and presentation as well as providing transparent communication to the host system(s) servicing the data requirements. Current relational database systems and fourth generation tools available on both platforms include: ORACLE, INGRES, COGNOS, FOCUS, and ALLBASE. With the addition of the POSIX interface on MPE iX this list is expected to grow in the future providing the HP3000 with strong multi-vendor database connectivity.

Paper 2022

Using Structured Analysis for Application Migration

Phil A. Beetley

Hewlett-Packard Company
201 West 103rd Street, Suite 100
Indianapolis, IN 46290
Tel: (317) 844-4100

Abstract

Planning and implementing the migration of applications from one hardware/software platform to another can pose many serious problems. For those instances in which a given application is complex, widely-used and evolutionary (i.e., modified over time to meet changing user requirements), there may not be a single, coherent description of what the application does. If some form of description does exist, it may be unofficial, out-of-date or not written down. Is there a methodology which can organize and facilitate the migration process in these situations?

Experience with such a situation, involving an application originally developed for an HP1000 computer, has shown that the tools and techniques of structured analysis provide a sound framework for application migration. Structured analysis identifies mandatory and desirable functional requirements, differentiates types of users and their respective concerns, and addresses organizational, technical and economic feasibility. This paper explains how structured analysis methods were used to identify, evaluate and recommend alternatives for migrating a computer-integrated manufacturing application for a major chemical/pharmaceutical firm.

Characteristics of Evolutionary Applications

Few applications remain unchanged once installed and running. As user requirements change, applications are revised accordingly. If an application is designed and coded in a modular fashion, with its modules able to interface easily to new software, modules may be reused by programs which have little in common with the original application. In certain cases, modules that can operate in isolation from the application as a whole come to be employed as standalone utilities, with users having little knowledge of the functionality the modules provide to some larger program.

One goal of application developers is to write software that will have a long operational life, long enough to recover the development costs. But a long application lifetime can cause problems. First, there can be very significant turnover among the users. Such turnover, combined with evolutionary changes to the application, means that it is unlikely that there can be a concensus on the true purpose of the application. If the purpose of an application is not clear, it is difficult to make sound decisions regarding migration to a new hardware/software platform.

Second, the quality of the changes made over time to the original application may vary widely. Some code may work well, but exhibit poor programming technique and style. Other code may be well-written, but not work. And, of course, there is always the chance that code has been added which serves no useful purpose because it fails to satisfy user requirements. It has been estimated that in mature software-producing organizations, up to 80% of their total effort is spent on software maintenance. But is is also true that few such organizations assign their most experienced and skilled personnel to maintenance tasks.

Third, evolutionary changes can be highly localized. This is especially true if a) changes are not coordinated, b) there is poor communication among the people making the changes, or c) the changes affect a small, or relatively isolated, portion of the user community. Localized changes can be a serious problem to migration, especially when they satisfy conflicting user needs or cannot be reconciled into a coherent application design.

Fourth, the state of the art may advance a different rates. As an application evolves, certain components may be state of the art while others lag well behind. This uneven advance may obscure the true importance of the application components. It is critical to look deeper than implementation in order to identify the core - the heart - of an application. It is also important to remember that the state of the art at any time may be based on proprietary technology, and that proprietary technology can restrict migration alternatives.

To summarize, any complex application can pose problems to migration. The successful migration of evolutionary applications is often made more difficult by turnover in the user community, varying quality, localization of changes and uneven technical advances. To be considered as suitable for use in migration, a methodology must be able to resolve these problems.

Structured Analysis - A Brief Background

Structured analysis was developed to address deficiencies in classical systems analysis, deficiencies which caused major problems as application software and systems became increasingly complex throughout the 1970s. In brief, the primary drawback of classical systems analysis was its reliance on narrative description. Narrative descriptions of application purpose and process are inherently ambiguous, often quite long, difficult to break down, and hard to maintain. In fact, narrative descriptions were so difficult to write that some were never written and many that were written were incomplete and soon out of date.

Structured analysis is a set of tools and techniques designed to bring order and rigor to the analytical process. Its purpose is to enable the analyst to develop, in a step-by-step fashion, a new type of document, the structured specification. In contrast to the narrative description, a structured specification is primarily graphic, made up largely of diagrams. Because it is graphic, a structured specification is concise and unambiguous. Furthermore, a structured specification is developed in a hierarchical (top-down) manner, proceeding from abstract to detailed levels. Because it is hierarchical, a structured specification can be decomposed, or broken down, into simpler component specifications, each of which can be refined independently. With independent refinement, division of labor is possible. Lastly, a structured specification is easy to maintain. Changes can be identified and associated with the appropriate component specifications.

Structured analysis and the complementary methodology of structured design were proven effective in industry practice from the late-1970s through the 1980s. While acceptance and implementation of structured methods has not been universal, research has shown that structured methods, when properly applied, improve application quality, reduce rework, and help to keep development on schedule and at budget.

The Structured Specification

The product of structured analysis is the structured specification, which has three main parts. The first part is a set of data flow diagrams (DFD). The DFD set is a collection of network diagrams which depict the application from the perspective of the data which "moves" through it. DFDs identify four types of constructs: data sources, data sinks, data flows and processes. The second part is the data dictionary. In the data dictionary, the composition of each data flow is described in fine detail. Data dictionary entries incorporate special symbols to denote repeating entries, optional entries and so on. The third part is the process specification. The process specification is the set of rules which define how a process transforms input data flows into output data flows. This set of rules may be expressed in graphic (e.g., decision table, decision tree) or restricted narrative (e.g., pseudocode) form.

Because a structured specification is graphic, concise and unambiguous, it is an ideal tool for identifying and reconciling different users' views of application functionality.

Structured analysis can be used to create a structured specification for an existing application program or an existing manual procedure which is to be automated. It can also be used to create a paper model of a target application, i.e., an application to be developed. In the case of migration, a structured specification can be used to describe the functionality of an application on its current hardware/software platform and what changes in functionality, if any, are to be accomplished during the migration to its new platform.

Mandatory and Desirable Requirements

The purpose of an application program is to satisfy the needs of its users. Structured analysis divides user needs into two distinct categories: mandatory and desirable. Mandatory requirements are defined as the requirements which an application must satisfy fully. It is not enough to satisfy a subset of the mandatory requirements; all mandatory requirements must be met.

If mandatory requirements are "must haves," desirable requirements are "nice to haves" or "like to haves." Desirable requirements are used as "tie breakers" to differentiate between alternative approaches which meet all mandatory requirements. In order to carry out a successful application migration, one must identify the mandatory requirements and ensure that all of them are met by the implementation on the new hardware/software platform.

(Note: in theory, once requirements are categorized as mandatory or desirable, no changes are permitted: the requirements are fixed. In practice, there can, and should be, changes to reflect critical changes in users' needs. An analogy is the changing of water into ice. When water is frozen, it becomes solid and can be used as a foundation. Ice can be turned back into water, but it takes energy to do so. Requirements can be "frozen" at some point in the analysis process, and revised at some later time. Making such revisions, however, <u>always</u> has a cost associated with it.

For application migration, it is very important to clearly identify all requirements and then carefully differentiate them as mandatory or desirable. The more mandatory requirements that an application must meet, the larger and more complex that application is likely to be. As a rule, larger and more complex applications are more difficult to migrate. When dealing with existing applications, requirements are reflected by functionality: if an application does something, the reasonable assumption is that the functionality is there to partially or fully satisfy some requirement(s). In those situations where there is no record of original requirements, functionality is an important source of insight.

When categorizing requirements, it is important to realize that different users may have widely varying opinions about any given requirement. This is likely if users interact with relatively independent portions of a large, multifaceted application. Each user may only know the functionality that he or she uses on a regular basis. To that user, the known functionality is mandatory while all other functionality is at best desirable and perhaps unnecessary.

Because a structured specification identifies all basic constructs (data sources and sinks, data flows and transforms), describes all data flows in detail by means of a data dictionary and explains all process rules, there is no lack of information for categorizing requirements. Furthermore, a structured specification makes it easy to locate redundant functionality, which is common in evolutionary applications that have been changed in a localized or uncoordinated manner. Removing redundant functionality is a painless way to reduce application complexity.

Feasibility Issues

Feasibility is the characteristic of being capable of being done, effected or accomplished. Feasibility assessment is not an activity of structured analysis *per se*, but feasibility can, and very often should be, considered once mandatory and desirable requirements are identified. Three types of feasibility should be considered: organizational, economic and technical.

Organizational feasibility refers to how well any given product or service, e.g., the results of an application migration effort, agrees with organizational culture, structure, goals and strategic direction. A migration effort, no matter what its technical merits, must be judged a failure if it produces an implementation which does not fit organizational constraints or further organizational objectives. For example, consider the computing strategy of adopting open systems. If an organization has this strategy, it makes no sense to plan a migration to proprietary hardware/operating system platforms.

Economic feasibility refers to how much an effort or activity costs. The most common means of expressing economic feasibility is in terms of a budget, measured in dollars or some other currency. But economic feasibility in its widest context takes into account all resources - money, time, personnel, compute cycles, megabytes of mass storage, and so on. Economic constraints are often very specific quantities: a budget of X dollars, a schedule of Y months, Z hours of staff overtime, etc. Once mandatory and desirable requirements are fixed, economic feasibility should assessed. If meeting all mandatory requirements is economically feasible, one can rank the desirable requirements and estimate the cost of each. If meeting all mandatory requirements is economically infeasible, either the mandatory requirements must be reduced or additional resources must be allocated.

Technical feasibility is concerned with the current state of the art - for application migration, the state of computing - and if the technology exists to meet the functional requirements. In those situations where no additional mandatory and/or desirable requirements are to be satisfied by the migration, technical feasibility is usually assured due to technical advances that have occurred during an application's lifetime. Conversely, if new requirements must be met by an implementation on a new hardware/software platform, the implementation may become technically infeasible.

Finally, it should be understood that organizational, economic and technical feasibility are never completely separate. Each can have a profound effect on the other two. To continue from a previous example, suppose that an organization is moving to open systems and has a fixed budget for application migration. It may not be possible to migrate a complex application under such constraints. There may not be an open systems platform with adequate power available at or under the budgeted amount. An application migration effort, therefore, which proceeds under organizational, economic and technical constraints is an exercise in optimization, aiming to deliver the best possible results under a range of constraints.

Types of Users and Their Concerns

There is not always an average or representative user for an application. There are usually three different types of users. Each type of user has unique concerns, and those concerns can be directly reflected in constraints which impact feasibility.

The first user type is the system owner, usually upper management. The system owner has final responsibility for the success or failure of an application. The system owner is charged with ensuring that a given application is in agreement with organizational policy, adheres to organizational standards and contributes to the meeting of organizational objectives. The system owner usually sets or communicates relevant constraints and is the final judge of organizational feasibility.

The second type is the responsible user, the person who is directly responsible for the functionality of a given application. In business organizations, this person has profit/loss or cost-center responsibilities. In production facilities, this person is likely to be responsible for production goals and quality. The responsible user is likely to be most concerned with economic and technical feasibility.

The third type of user is the hands-on user, the person who uses the application in his or her daily work. It is hands-on users who often feel most strongly about what requirements an application must meet, while at the same time having a limited understanding of overall application functionality. The hands-on user is often concerned with what an application does (functionality) and how fast it does it (practical performance), and is less concerned with how (design/implementation). In cases where there is no written description of current application functionality, information from the hands-on users, organized into a structured specification, is invaluable.

One of the critical activities for anyone responsible for application migration is to identify key people of each user type, collect information from them, and act as a liaison between the various users. It is vitally important that input be solicited from all areas of the user community.

The Central Transform

All application programs can be described in abstract terms as being made up of only three types of activities: input, process and output. Input activities bring data into the application from "outside" and cast that data into a form suitable for processing. The process activity transforms the input data according to the purpose of the application. Put another way, the process activity transforms the input data in order to satisfy user requirements. One can think of the process as "adding value" to the input data; the process converts data into information. Output activities take output data from the process and put it in a form is appropriate for communication with the "outside." The complexity of each type of activity can vary greatly from application to application.

Structured analysis and design have specific terms for the three abstract activities. The input data is made up of *afferent* data data flows, the output data is made up of *efferent* data flows and the critical value-adding process is referred to as the *central transform.* The central transform is the mission-critical portion of an application. If the central transform is correct, the application can meet user requirements. But if the central transform is flawed, the application cannot succeed.

When analyzing the migration of an evolutionary application, one can use the concept of the central transform to help identify core functionality, and, by extension, the mandatory requirements of the application. Identifying the central transform is a way to get back to basics, to focus on what's really important. Many applications are developed or evolve with complicated afferent and efferent activities - interactive user/computer dialogue managers, data validation/edit routines, graphical report generators and so on. In some cases, the software to accomplish these activities exists independently of the central transform. In these situations, certain users may not know, or care, what the central transform does.

Summary

To successfully migrate an evolutionary application, one needs to know several things. Structured analysis provides tools and techniques well-suited to collecting, organizing, communicating and validating information about an application. Structured analysis can be augmented by feasibility assessment to account for organizational, economic and technical issues. It is also useful to identify and categorize users and to understand how user roles affect feasibility. Finally, the central transform concept can be employed to determine and focus on mission-critical functionality.

The remainder of this paper will describe how these methods and concepts were applied during a specific consulting assignment to identify and evaluate migration strategies for a data historian application.

Customer Profile

The customer was a major pharmaceutical/chemical manufacturer which employs a wide range of Hewlett-Packard and other computer systems for computer integrated manufacturing (CIM). At one particular manufacturing facility, a data historian application was running on an HP1000 A900 computer under the RTE-A V5.2 operating system. Work was underway to bring additional vats into production, with an attendant increase in the volume of process data to be collected, analyzed and stored.

In addition, the plant was in the early stages of a major quality-control effort responsible for improving all aspects of production. Statistical process control would form the basis of this effort. The process automation department was concerned that the HP1000 system could not provide enough computing power or mass storage to meet these new demands. Hewlett-Packard was contracted to analyze the data historian application, identify and evaluate alternative migration strategies and make a recommendation. The recommendation was to be vendor neutral, containing no assumptions that Hewlett-Packard would implement any portion of the migration.

Analyzing the Data Historian

Initial information about the design and functionality of the data historian came from meetings with the senior systems analyst from the process control department and a quality analyst who had been a key member of the original development team. From these meetings, it was clear that additional interviews with other process automation staff and production workers would be necessary. It was also clear that it would be helpful to have an analytical framework to guide the consulting assignment.

The description of the data historian produced by the interviews was as follows: some of the software for data entry and operator control was written by the process automation department, some was purchased from a commercial software developer; a memory-resident, real-time database was procured from a major vendor in the factory automation market; some of the software for producing graphs and charts was from the same factory automation vendor, while other output software had been purchased, then customized by process automation staff, and the complex data compression routines had been written by the man who was now the quality analyst. The documentation was a collection of vendor manuals, training guides, class notes, descriptions of module design and calling conventions, on-line help text, and comments in source code files.

The next step was to study the available documentation and source code to understand how these different pieces of software worked together to carry out the various data historian functions. One major concern was the degree of interdependence (coupling) among the programs and their respective modules. Loose coupling would mean that the overall design was modular, and modular software is always easier to maintain, change and migrate. Fortunately, the design proved to be modular: the in-memory data structures maintained by the real-time database were well documented and used consistently by various programs; otherwise, programs were written to access the process data once stored on disk.

Identifying the Central Transform

Identifying the central transform proved fairly easy. From talking to the process automation staff and analyzing the function of each component program, it was clear that the real-time database was the heart of the data historian. The other programs either prepared data for inclusion in the database or took data from the database, stored it, or presented it to users. By tracking afferent (input) and efferent (output) data flows, finding the central transform was a straightforward task.

Organizational feasibility also played a role in determining the central transform. The customer had implemented this particular real-time database at several plants, was very pleased with its performance and was committed to using it as the basis for the quality-control effort. (The factory automation vendor also sold a statistical process control product which interfaced to the real-time database.) For these reasons, the customer had decided that the current real-time database, which was available on a range of hardware/software platforms, would be the incorporated into all data historians.

This decision also resolved two other aspects of technical feasibility. The customer had already determined that the real-time database versions for HP9000 HP-UX systems and Digital Equipment Corporation (DEC) Vax/VMS systems would provide the capacity required by expanding production. The customer was also confident that both the HP9000 and the DEC Vax offered configurations that could provide the needed computing power and mass storage. It is important to note here that the organizational constraint of using this particular real-time database served to limit the number of suitable hardware/software platforms.

Reporting Preliminary Results

When the user interviews and source code inspection were done, a preliminary report was presented to the senior systems analyst and the process automation department head. Data flow diagram sets, program structure charts, data dictionary entries and process specifications were included. The senior analyst and department head agreed that structured analysis provided an effective framework for the consulting assignment. They also agreed that all major software components in the data historian had been identified. Approval was given to proceed to the next phase of the consulting assignment.

Alternative Migration Strategies

One key deliverable of the consulting assignment was to identify alternative migration strategies, evaluate them and recommend one. Eventually, four alternative strategies were developed:

- Migrate the current implementation to a new hardware/software platform, making no changes in functionality and employing software that worked as closely as possible to the current data historian. This alternative was referred to as the "straightforward migration." For example, customer-developed programs written in Fortran or Pascal would be transferred to the new platform and recompiled. Modifications to source code would be limited to those required for successful compilation. This alternative was based on the assumption that there would be no significant technical problems to overcome.

- Migrate the current implementation to a new hardware/software platform, making no changes in functionality, and employing software that worked as much as possible as the current data historian, but with the assumption that there would be significant technical problems to overcome. This alternative was referred to as the "difficult migration." Due to routine use of system calls in the Fortran and Pascal code, among other reasons, it was considered likely that there would be major technical problems.

- Redevelop the data historian on a new platform, taking full advantage of whatever new facilities are provided by the new hardware and software. This was referred to as the "conversion alternative." All current functionality would be retained and, where feasible, the design would be modified to allow application features to be added at some later date. If, for example, the decision was to migrate to HP-UX, Fortran and Pascal programs would be rewritten in C and RTE-A command files would be rewritten as shell scripts.

- Provide functionality equal to or greater than the current data historian implementation by the specification, installation and integration of commercially-available software products. The integration could be carried out by customer staff, the vendor of the new platform or by a third-party. This was referred to as the "tool integration alternative."

Evaluation and Recommendation

From discussions with the process automation department head, it was clear that the evaluation process must consider both organizational and economic feasibility. (Recall that technical feasibility was in large part determined by the decision to use a specific real-time database. This limited the range of new platforms to those computer systems for which the database was available.) To do so, the concept of "ownership" was applied. The ownership characteristic of each migration alternative was determined by five factors:

1) The ability of the process automation staff at the plant to maintain the data historian after migration.

2) The ability of the process automation staff at the plant to enhance the data historian after migration.

3) Training costs for the data historian.

4) New staff costs for the data historian.

5) The length of the operational life of the data historian.

Ownership was a means of comparing what resources it would take to support the migrated application over its operational lifetime.

It was also necessary to consider the initial cost of each migration alternative. Two types of costs were considered - personnel costs (labor) and technology costs (tools). When the four migration alternatives were evaluated according to ownership and costs, the following ranking resulted:

MIGRATION ALTERNATIVE	OWNERSHIP	COSTS
straightforward migration	low	low
difficult migration	low	high
conversion	high	high
tool integration	high	low

Based on this ranking, Hewlett-Packard recommended that the customer select the tool integration strategy for the migration of the data historian. This alternative also seemed to be the most compatible with the customer's plans for the wider adoption of open systems.

The customer accepted this recommendation with the final report, which was presented to a select group of customer personnel as part of the consulting assignment deliverables.

Final Outcome

With respect to producing all required deliverables, the consulting assignment was a success. Hewlett-Packard produced a final report which described the current data historian in detail, identified four alternative approaches to migration, evaluated each alternative and made a specific recommendation. The final report was vendor neutral as the customer had specified.

In the end, a DEC Vax/VMS system was selected as the new platform. This decision was strongly influenced by an organizational constraint which was introduced after the consulting assignment was complete, i.e., that the data historians at different plants be standardized on a specific commercial package. The package was only available on DEC Vax/VMS systems; the developer was considering porting a version to HP-UX, but the port would not be completed in time. The data historian had to be up and running when the higher volume of process data started coming in from the expanded production areas.

In short, the migration of the data historian was accomplished by means of tool integration, just as the consulting assignment had recommended. But the new platform was a proprietary system, not an open one.

Conclusion

The last ten years have produced great advances in hardware design and performance. Processors have increased in power to the point that 100 MIPS is available in desktop systems. Memory, mass storage and display technology have also made significant gains. There are a number of *de facto* and *de jura* standards which, if followed, make open systems a reality. But for all this, it remains the application software that does the work of computing. Organizations with proven applications, applications which have cost a great deal to implement, need an effective method for application migration.

Structured analysis is an effective method for application migration. The reasons for its effectiveness are as follows:

* The tools and techniques were developed to address many issues that are central to application migration.

* The methodology is well-proven by fifteen years of industry experience, especially for larger, more complex applications.

* The basic aspects of structured analysis are easily learned and there are numerous good good books and seminars available.

* It is compatible with various design methods, including structured design and prototyping.

- * Structured analysis is appropriate for internal (i.e., in-house) software development and for generating specifications to guide procurement.

- * Many CASE products are based on structured analysis, making software process automation tools available for application migration.

The decision to migrate a complex application to another hardware/software platform is never trivial. Organizations which do not have an effective methodology for migration may perceive the risks of migration as being too high. After all, migration is predicated on the fact that there is a current, functional application doing important work. But if applications are not migrated, the benefits of current technology, such as open systems, cannot be realized.

Structured analysis is an effective methodology for application migration. It does not make application migration easy, but it does make it a rational and manageable process, a process that organizations can undertake with confidence.

References

Allen, C. Paul. *Effective Structured Techniques: From Strategy to CASE.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1991.

DeMarco, Tom. *Structured Analysis and System Specification.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1979.

Fornier, Roger. *The Practical Guide to Structured System Development and Maintenance.* Yourdon Press, New York, 1991.

Gane, Chris and Sarson, Trish. *Structured Systems Analysis: Tools and Techniques.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1979.

Page-Jones, Meilor. T*he Practical Guide to Structured Systems Design.* Yourdon Press, New York, 1980.

Weaver, Audrey M. *Using the Structured Techniques, A Case Study.* Yourdon Press, New York, 1987.

Weinberg, Victor. *Structured Analysis*. Yourdon Press, New York, 1978.

Yourdon, Edward. *Managing the Structured Techniques.* Yourdon Press, Englewood Cliffs, New Jersey, 1989.

Yourdon, Edward. *Modern Structured Analysis*. Yourdon Press, Englewood Cliffs, New Jersey, 1989.

Comprehensive Automated

Storage Mangement

for

Networked Environments

Gottfried B. Bertram

R&D Manager

System Management Division

Hewlett Packard GmbH
Herrenberger Str. 130
W-7030 Boeblingen, Germany
Phone: +49-7031-142644

Abstract

Knowing that "Network is the Computer" implies distribution of data on many storage devices throughout a network, the system. People responsible for the information processing architecture, i.e., the distributed data center, must be capable for managing these stored information elements and storage locations. This requires complex designs and operations. The advantage of open distributed systems must not be lost because of problems associated with managing them.

To resolve these problems, highly automated solutions for unattended operations, easy of use and setup, hiding of unnecessary complexity, etc., is currently under development by HP's System Management Division (SMD). The Automated Storage Management (ASM) project will address the full complexity of networked environments we encounter today.

A comprehensive Approach

The networked environments we see today, and those we expect and envision to serve with an automated storage management solution consist of several servers linked together with a variety of desktop clients. These servers, typically HP 9000 (Unix), or HP 3000 (MPE) mini computers serve Terminals, PC's, and Workstations. The network being the system, to manage it is characterized by a possible co-existence of threeimportant network operating systems (NOS):

- Unix networks on TCP/IP offering OSF's Distributed Computing Environment (DCE) services

- Novel's Netware

- Lan Manager

Since these are portable versions of Netware and Lan Manager, we expect all three operating environments to coexist in the same network. Such an environment implies a large number and variety of storage locations and organizations to serve, hence increasing the environment complexity.

The source of information objects are from:

- Personal Computer Clients (DOS)
- Workstation clients (UX)
- Personal Computer Servers (Netware, LM)
- Mini Computer Servers (UX and MPE)

These information objects, for storage management purposes, may be organised as follows:

- Disks, volumes, raw device contents
- Files, directories of files
- Data bases

To implement Storage Management, the following operations are required:

- Backup and recovery / restore
- Archival and retrieval
- Storage space management

From the above sources, the kinds of information objects and the kinds of operations, there are three dozen combinations that a comprehensive storage management solution must address. Although, not all of these combinations make sense, e.g., to archive disks for historical or legal purposes is certainly not meaningful, and data bases will almost always be backed up from servers and not from desktop clients, a considerably long list of storage management solutions need to be developed for distributed environments:

Disk Backup / Recovery for:

- DOS PC's, MAC's and Workstations

- UN*X Servers with portable Netware and portable LAN Manager (LM/X)

- MPE Servers with portable Netware and portable LAN Manager (LM/iX)

- Native Netware (INTEL-PC) Servers

- LAN Manager Server on OS/2 PC's

Data Base Backup / Recovery for:

- UN*X Servers
- MPE Servers
- INTEL PC Servers

If data bases are to be backed up from servers, the different Data Base Management System (DBMS) suppliers must be taken into account as they require different support for their special flavour of backup processes.

File Backup / Restore from and to:

- DOS PC's, MAC's and Workstations

- UN*X File Servers

- Novell Netware File Servers

- LAN Manager File Servers

- MPE Servers

Assuming that all three major network operating environments coexist on the same network, file backup and restore must be able to handle the different file name conventions of DOS, MAC, Netware, OS/2, UN*X, NFS, DCE/DFS and some proprietary environments.

Archival and Retrieval

For some environments, both archival and retrieval may be required. Archival differs from backup as it's purpose is not to safeguard a copy, but to retain historical copies of files for legal and other purposes, e.g., version maintenance.

Data base archival is a required storage management task, and can be offered as an integrated solution consisting file archival and procedures offered by the major DBMS suppliers, e.g., ORACLE, INFORMIX, INGRES, SYBASE, DB2, ALLBASE, ADABAS, etc. (Most DBMS have utilities to unload / export data base tables to the file system of their host server and to reload / import them). Archiving is then achieved by moving these files into archive together with appropriate attributes to retrieve them later.

Storage Management

The purposes of storage space management is to monitor usage levels of disks and to free up space, by rolling out stored information objects that are infrequently accessed or are already old and obsolete. Storage space management maybe considered for all disks in the system:

-DOS PC's, MAC's workstations (i.e., all clients)
- UN*X, MPE Servers
- PC Servers (Netware, OS/2)

Some scientific and technical applications use huge amounts of data, organised in file systems too big to be always locally present. Storage space management may serve these environments offering high performance, automatic, roll out / roll in processes to integrate data from fast local disks to networked storage servers, and to removable high density media in autochangers, and to migrate them back transparently if accessed.

The Storage Management Process Model

Managing systems/networks must focus on people who perform the management role, the methods and tools they use to perform that role, and the processes that describe how people perform their management role using the methods and tools at hand. This applies to system management in general and to storage management in particular.

Technology changes like distributed or client-server computing not only introduce more services, possibilities, capabilities and advantages over traditional centralized mainframes, but also introduce more complexity. Processes to manage these distributed computing environments must reduce this complexity through automation.

In the case of storage management, the key trend identified is to shift the focus from different stored information objects and their management tasks, e.g., backup, recover, archive and retrieve, to the people who manage them. Only the automation of tasks and entire processes people perform, will cope with the challenge to keep pace with the changes in information processing environments.

To manage the backup, recovery, archival and retrieval of such different objects as disks files and databases, together with control over storage space available across an entire network, cannot be done any more by the selective and individual usage of single tools and methods. It needs a bundle of processes to assure safeguarding and archiving of valuable information. Ideally, we think of finding a way to free up the people from carrying out the processes to let them only manage the process of storage management.

The solution is not to build tools people can use to manage stored information objects efficiently, but to actually remove the people from the process as much as possible. Human reaction and interaction speed is limited, as are human abilities to cope with situations of even increasing complexity like data bases, file systems and storage locations scattered throughout the network. What needs to be automated are processes that are large complex collections of tasks performed by people today.

Backup a data base from one server in the network to a special backup and storage server that offers all suitable peripheral devices and media, requires a long sequence of task to be carried out, to be understood and mastered. It requires the knowledge of specific DBMS's on line backup procedures, the way and format of the data to be backed up, how they need to be transported to the backup up server, and where they will be placed on what media.

Recovery is even more complicated. Such management processes need improvements in the people, in the tools and methods and the processes. Most solutions today only enable storage management for some of the different purposes, and fall short of what customers need as they do not deal with the processes that people actually have to perform.

For the definition of advanced storage management solutions, we therefore need a solution maturity framework, and a target level of maturity we want to achieve for the next generation of storage management solutions. Todays solutions are characterized by the fact that all knowledge on the processes has to be in the heads of the people managing the system. These solutions only offer tools to automate single steps and tasks. To explain, we may call this an "initial" level of maturity, followed by a level called "repeatable", and another level described as "definable"

- Initial

A variety of single step solutions address partial needs only. An absence of consistent, comprehensive and integrated set of methods and tools impede the creation of reliable policies, procedures and processes, to safeguard and archive valuable information.

Methods and tools for storage management are known and tools for all specific purposes may be available, but none of them are integrated or based on a consistent architecture or framework. Operations are carried out on an individual, adhoc basis. There is no common or centrally available knowledge in case there is loss of information and requires recovery.

- Repeatable

Solutions offer a set of fairly comprehensive tools and methods. These tools can be put together and configured to carry out a variety or sequence of tasks.

But the configuration and operation of such sequences still requires an expert to design and set up the processes desired. Very often these tasks are performed by a specialized person or a guru who knows about the 'what', 'when' and 'how'of the processes. He is the person everybody knows they need to call just in case a complicated recovery has to be carried out. Processes depending on persons bear the risk of loosing the experience and expertise with changes of either environment / technology or with personnel / organizational changes. A shift from centralizing computing environments already implies such a shift in paradigm, resulting in demand for new processes to manage the backup, archival and storage space available in networked systems. Of course, this change does introduce greater complexity together with tremendous advantages of having computing power everywhere on a network. But these complexities aggravate the problems of management of such systems. SMD's ASM project is therefore targeted to achieve the next level of management solution maturity.

- Definable

As size and complexity of distributed data processing environments grow continuously, so does an increase in the risks of managing them by repeatable processes only. Too rapid a change, complexity and variety have to be learned and mastered by too many experts. One can always increase management resources and improve processes, but beyond a certain level of complexity, help can only be expected by another level of automation. The "definable" level of solution maturity will allow policies to be carried out automatically and independently from personally owned knowledge.

If solutions are policy controlled, the policies will be available, documented and independent of specific experts. They may be inspected, changed and adapted. The defined solution is people independent, consist of documented policies that record process characteristics, steps and tasks that can be passed over to people, and either are executed manually or by pre-programmed reactions, operations and administrative measures. Hence, it achieves a higher degree of automation and decreases the complexity seen by people managing the system. The definable policy level of management solutions enhance management by managing the processes.

For design of storage management solutions we need a general management process architecture that enables definable level of solutions. We see a four stage architecture for storage management purposes and processes: operation, execution, administration, change of policy and planning.

Architecture of Storage Management

An architecture for the definable solutions of self-managed systems consists of three process loops on three operational levels. The lowest level for execution of storage management operations offer storage services together with transportation and transformation of stored information objects from and to storage management clients. These services operate on the distributed system and record results from the operations. The results gathered are measurements fed into the next higher level, i.e., operational and then administraional levels. For the operational level, the operations executed and events recorded from a loop to control the correct execution of the operations. The record loop is associated with the administrational level to keep track of the state changes and the history of operations. The third loop is between the administrational and operational levels. Application strategies get policies and schedules set by administrators and put them into the administrational services. From these administrational services, the application strategies are feedback records, to be offered as reports. The administrational services execute the policies regarding what to do when, through schedules applied to the application operations.

Both the application strategies and application operations, come in different flavours, e.g. database backup and archival or file backup and archival, and their inverse operations, restore and retrieval. The application operations may monitor and report events and certain states to local operators, if there is no way to continue operations without human intervention. But usually, the application operations execute automatically, and only report results to the administrational services and application strategies, thus executing all processes automatically, controlled by policies and schedules that are set up as application strategies by administration. This is the criterion that applies to the definable level. One might argue that in the case of restore/recover or retrieval operations, there will always be a need for an operator, but such need depends on the nature of the event that requires a restore/recover/retrieval. In many cases these operations may be carried out and triggered automatically.

Besides the strategic and operational components of the storage management architecture, services for storage, transportation, transformation and administration have been mentioned. All these services are common to the case set of storage management functions.

Backup, archival and administration of available storage space use copying or moving mechanisms to get stored information objects from storage clients to storage servers and to place them on the available hierarchy of storage devices. Regardless of application, such transportation can be achieved on LAN or local BUS data channels, by generally available methods for file transfer and networked file management (e.g., NFS, DFS, Netware Lan Manager, FIP, FTAM). The administrational components will have to record the names, locations (sources and destinations) owners, etc., and provide these information attributes to storage management applications. To execute copying and moving, a good architecture should provide separate command and data channels.

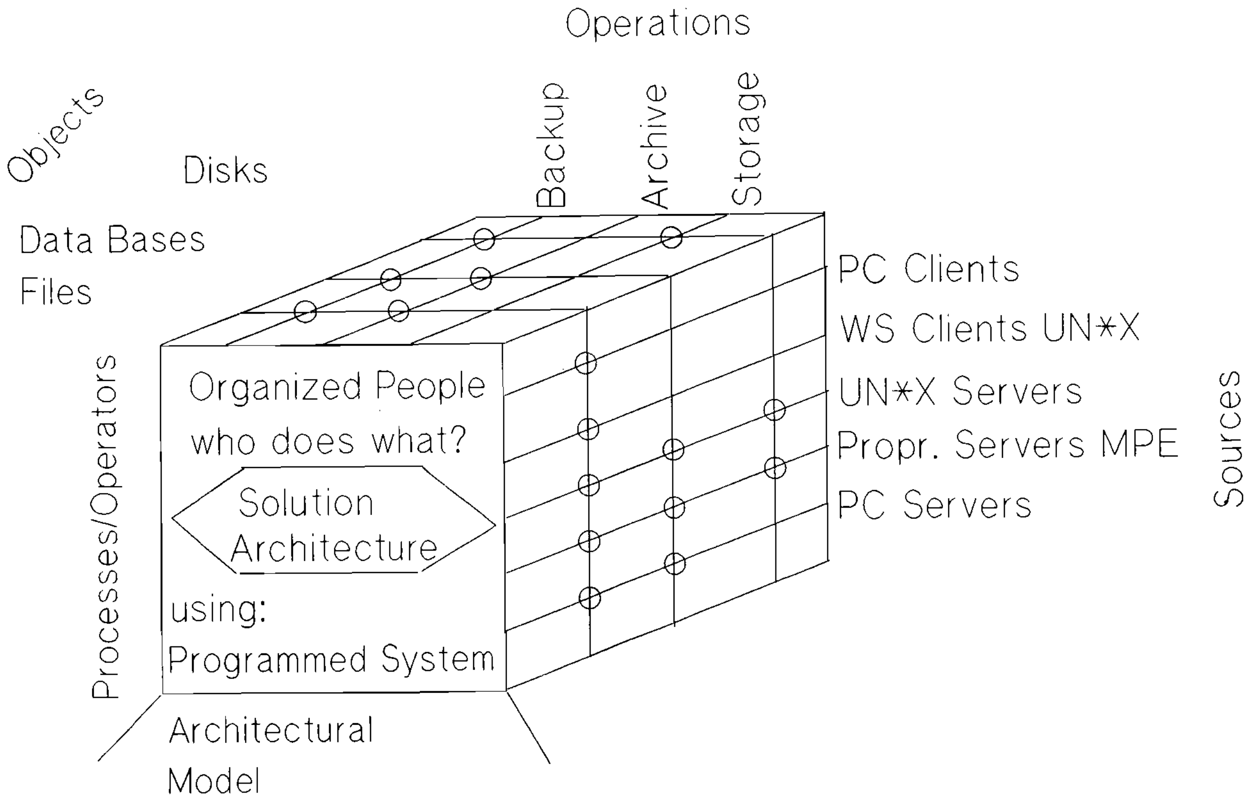
On the command channel, management applications would talk to local agent processes for the transport execution. A remote management application may thus activate local managers, that in turn, use local agents. The communication of remote activation of operations, and the feedback of events signalling correct or unsuccessful operations, would be carried out using standard management protocols.

Transportation is not the only class of function required for central operational core. Usually all information objects that will be entered into storage management undergo some transformation or translation. Desk files and database tables will be transformed into a standard object, consisting of a header and a bit stream to be stored on disk or other devices. One may choose some standard formats for that, and conversions between them, e.g., tar, cpio, etc. Very often, encryptions, compressions or import/export transformation for databases may be asked for. Since there will be different file systems that coexisting on the netware, it will be required to translate between the different file name specifications of UN*X, DFS, Novell, Mac, DOS, HPFS, etc. The administration service mentioned before, is a database driven application, using a relational data base service either centrally or distributed. This database will serve as the repository for:

- clients and users, their contracts, rights and obligations.

- contracts on what to do when, i.e. policies, worklists and schedules.

- histories of executed operations (e.g. backup sessions) to provide information on what was executed and the results achieved.



Operations

Objects

Disks

Data Bases

Files

Backup

Archive

Storage

PC Clients

WS Clients UN*X

UN*X Servers

Propr. Servers MPE

PC Servers

Sources

Processes/Operators

Organized People

who does what?

Solution

Architecture

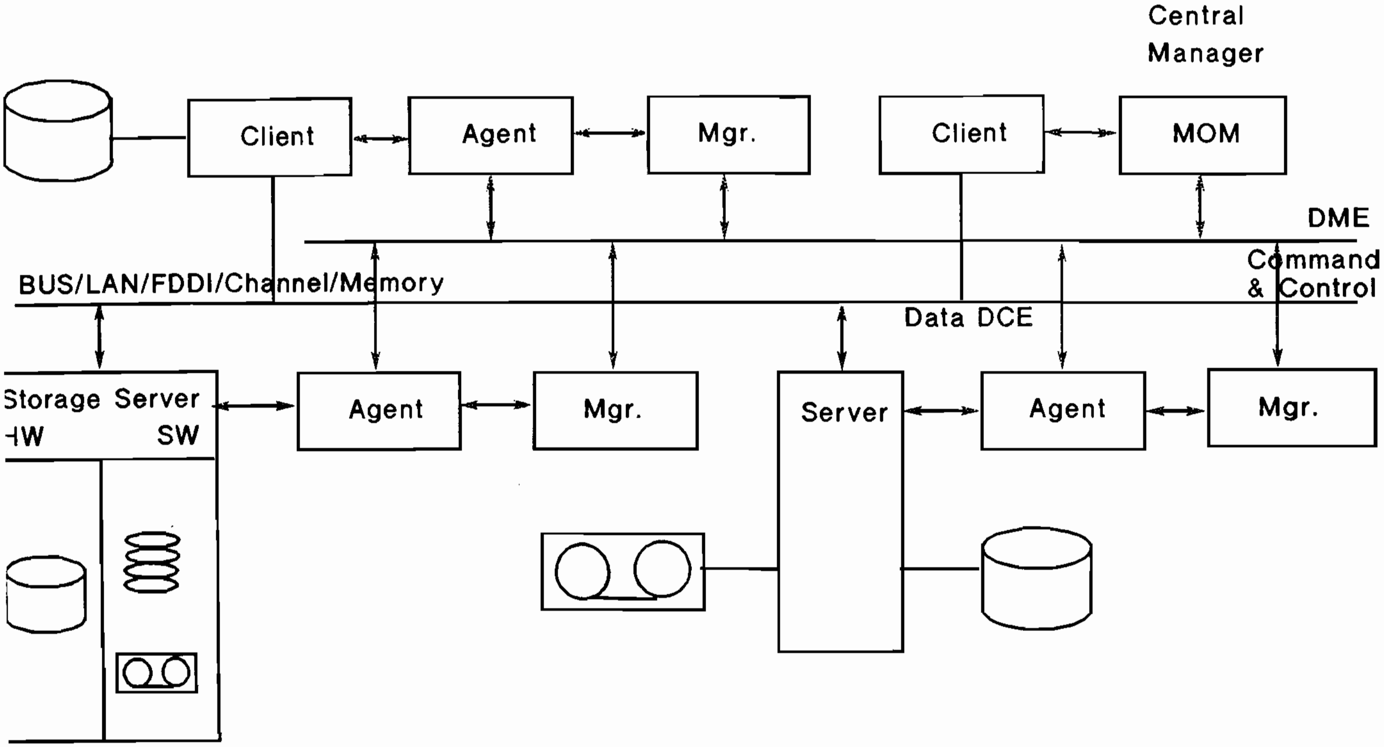
using:

Programmed System

Architectural

Model

ASM Problem Space



Storage Servers and Clients

Paper Number: 2024

Integrating Information With Object-Oriented Databases

Douglas Dedo

Hewlett-Packard
19111 Pruneridge Avenue 44MP
Cupertino, CA 95014
408-447-7726

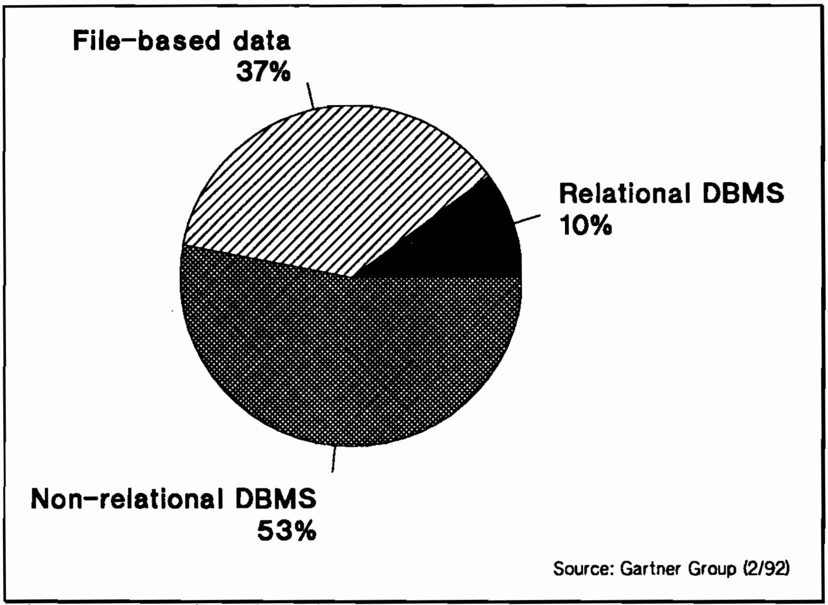
ABSTRACT

For businesses to stay competitive, their entire organization needs to have access to the right information at the right time regardless of where it is located. To do this, many different data sources need to be transparently integrated together creating useful business information. This information then needs to be presented back to a person in a business model that fits and supports their day to day activities. This presentation explores how an evolutionary object-oriented database approach can assist in this process.

Your visit to a hospital generates a very real need to pull together your medical information. It is not always easy to get quick access to your medical records, lab test results, X-Rays, and research literature related to your specific case. In fact, eight out of ten times your doctor will not have access to the right information at all. The experience of the physician then plays a major role in the quality of your health care. It would be a significant step forward for the entire hospital staff to be able to get quick access to the information they need to do their job.

Getting timely access to information is no less important for companies trying to deliver their products or services to their customers than for hospitals saving lives. The use of computers has enabled companies to build large sources of valuable data that are spread throughout the enterprise. This information is typically stored in many different formats. We hear customers describing the need for a new type of computer-based information system. One that gets the right information to the right people. Some people have changed the acronym EIS to stand for Everyone's Information System (where the "E" initially stood for Executive) reflecting this new line of thinking.

This type of system requires both the need to model information with an enterprise-wide view as well as a mechanism to get at all types of data formats. The enterprise-wide view is necessary if applications that support different functional areas within a company are to work together. The ability to access any type of data reflects the demographics of data today. Gartner Group, an industry consulting company, estimates that 92% of today's commercial business data is stored in non-relational formats (see Figure 1). If the data was in one or two types of systems, there would not be a problem. However, the data is in heterogeneous databases including network-model and relational as well as the file system. This complexity is what makes these systems so costly today.



Source: Gartner Group (2/92)

Figure 1. Distribution of enterprise data in 1992.

This paper will focus on how this kind of enterprise-wide, data integration system can be developed in a cost-conscience way using new type of core technology called an object-oriented database management system (ODBMS).

How is data accessed, integrated, and used today?

Many business decisions are made with limited or no data at all. For certain types of information like future results, neither computers nor crystal balls provide much help. For data that does exist, sometimes the cost to get at it exceeds the value in having it. Lowering the cost and time to get at data is the objective.

There are three common techniques that companies use today to pull together heterogeneous data:

1) The parachute approach;
2) Hand-crafted applications; and
3) General purpose tools

When information is required quickly for a significant business decision, a common approach is to send a scout parachuting into an organization to get the data by hand. This is time consuming and people intensive work which caries a high price tag. Once the raw data is available, the scout may do the work to integrate it together to create usable business information. This allows the decision makers to focus more on analysis of the information than interpretation of the data. This approach makes sense when the decisions being made are of a magnitude to merit this level of investment.

When the process to get at the data is going to be repeated by many people, special applications are written to make this task easier. These applications tend to fit one type of user in one part of an organization. Since they are hand-crafted in computer software to access a specific set of data, they do not lend themselves well to environments that change rapidly.

There are data access tools today designed to help both software developers and end users get at heterogeneous data in a more general purpose way. Hewlett-Packard has a data access tool called Info Access. Independent software companies like Information Builders, Inc. and Uniface have data access tools too. Once the data is accessed, the next step of integrating the raw data into refined business information can be a complex process.

What is an object-oriented database?

Object-oriented databases are a new type of technology that is emerging which can leverage from the knowledge that exists about a company's data and from the tools that exist to access data. They can make it easier for developers to get at heterogeneous data in a more cost-efficient way. The key to an ODBMS is the model used to represent information.

Originally applications stored their data in the computer file system. When commercial applications drove the need to share this data, network-model and hierarchical databases emerged. The internal model was data sets linked by a hierarchy or network of relationships. The need for more flexible decision support (ad hoc queries) led to the industry standard SQL language and relational databases. Internally, these systems define data in a table (rows & columns) model. There are no relationships modeled within the database however. The need driving the interest in object-oriented databases is a more complex decision support environment where the relationships between the data, and functions performed on the data, are as important as the data itself. The internal database model is built on user-defined types of information.

Before we get into object databases in more detail it is important to understand what the objects are which are being stored in these databases.

What are objects?

An object, simply stated, is business information that is simulated or modeled in a computer. Objects model this business information from a user-oriented perspective rather than a machine-oriented view that is common today. The objective for using objects is make it cost less for developers and end users to manage and keep track of valuable business information.

For example, the critical business information for a manufacturing manager could be the work cells, the employees on the shop floor, the suppliers as well as their parts, and the products being manufactured. For a soft drink company, this could be the process or recipe used to make a soft drink and then the process to distribute the product to customers. The key is that objects allow you to more directly represent the complexity of these real-world aspects in your computer. This differs from how information is handled today in that code and data are kept separate and the data is mapped into a machine-oriented structure like data sets or tables. The benefit of modeling using end user-oriented types reduces the complexity for application developers. This should speed up the availability of new applications over time.

Objects, from a more technical perspective, combine a related set of data and procedures together. When reading a document describing what a new application will do, objects can be identified by looking at all of the "nouns." In a personnel application for example, an "employee" would be modeled as an object. An object is really made up of two key components. First, at the core of this business information, is the data component. For an employee, this could be their name, address, business phone, and employee number. The second component can be identified by looking at the "verbs" in the application document. These are the functions that an object normally performs in the real world. For an employee-object, appropriate functions would be "hired," "promoted," "transferred," and "managed_by." These functions are represented in a computer by software that simulates the action. Putting the data associated with an employee-object together with its related functions allows a computer to model the real world as accurately as possible. This means that those in the personnel department can then call up an employee object and get all information related to that employee.

How does an object-oriented database help?

With this model, object-oriented databases are used to save, find, share, manipulate, and protect this business information.

The significance of the structure of an object-oriented design is that developers can model even the most complex business environments in an ODBMS. Since this model is managed in a general purpose database, it can be shared between different applications in the same way that just data is shared by preceding databases. This can lower the time and cost to get at business information.

Modeling the business is the only the first step. Once the model is designed, getting at the actual data presents a significant problem. Recognizing that most data today is not in relational databases, it is necessary to have a flexible way to map from the business model to where the data is physically located.

When Hewlett-Packard started research on object-oriented databases in 1984, one of the goals was to integrate into existing commercial environments. OpenODB, the product resulting from this research work, contains a mechanism called "external functions" that provides developers with a way to exit from the database to get access to any data throughout a company-wide network. This ability to model complex businesses and map that to any type of data wherever it is located is very attractive to different vertical applications. Figure 2 shows an architectural picture of this type of data integration application. The external functions can also be used to access existing code or tools to leverage from previous software development. Some examples of how this data integration architecture might be applied for different vertical markets are given below.

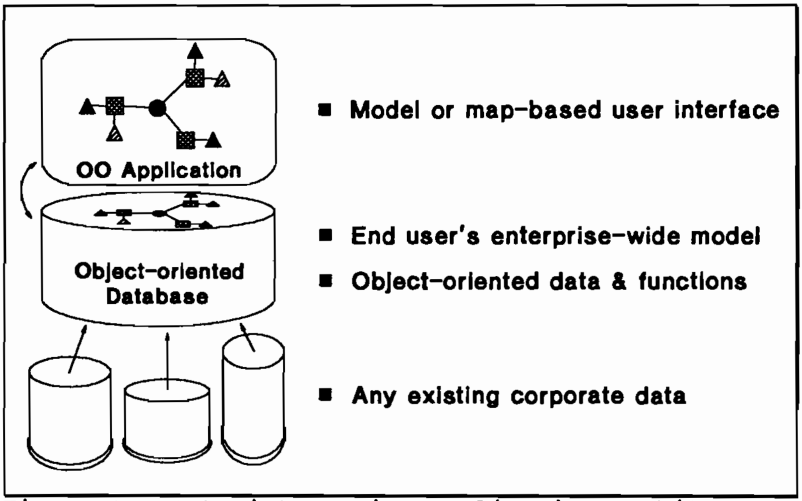

OO Application
Object-oriented Database
■ Model or map-based user interface
■ End user's enterprise-wide model
■ Object-oriented data & functions
■ Any existing corporate data

Figure 2. Data integration application architecture.

With this picture of a business model mapped to heterogeneous data in mind, there is one more dimension to complete the business solution. These systems are designed to support many people in environments that change frequently. As a result, there are a number of basic database capabilities that also need to be in place. When evaluating an ODBMS for commercial applications, it is important to investigate whether they support the following capabilities:

* concurrency control (many user requests at the same time)
* security and authorization
* database structure changes while applications run
* online maintenance (e.g. online backups)
* management of business processes (rollforward/rollback)
* query language (e.g. Object-oriented SQL)

OpenODB is an object-oriented database that uses a relational database, Allbase/SQL, as its internal engine to store and manage the data portion of objects. This was done for two reasons. First, it was determined that there is no difference in managing relational data and object-oriented data. By using a relational database, all of the basic database capabilities listed above were immediately available so that the new work just focused on all of the object-oriented capabilities. This made it possible to bring a complete, commercial-quality ODBMS to the market much faster than if the system were being built from the ground up.

The second reason for this hybrid approach involved HP's open systems strategy. The objective was to make this technology available on many different hardware systems. It was felt that the best way to do this was to license the object-oriented portion of OpenODB to other relational database vendors. Running on an independent software vendor's database would allow it to be ported across HP and non-HP computers.

What are some application examples?

The interface for applications that use an ODBMS lend themselves well to a map or model-based presentation. One example of this type of interface is found with Geographic Information Systems (GIS). A GIS application uses a map on the computer screen as a backdrop over which business information is displayed. GIS applications can be used in many different industries where information is related to maps like government agencies managing land, forests, or national defense; oil & gas exploration and production; and telecom network management. Model-based applications would use a different type of picture as the backdrop instead of a map. Applications like this could manage seats in an airplane or theatre as well as layouts for newspapers or magazines.

Data integration applications use the combination of enterprise modelling with heterogeneous data access available in OpenODB. These applications can be used in many industries like telecom, health care, pharmaceutical, oil & gas, financial, and computer software.

The telecommunications industry has been modeling their phone network with an object-oriented approach for almost fifteen years. ODBMS applications could access operations support systems. These systems assist operations, administration, management, and provisioning (OAM&P) activities. Applications that provide the needed information for these users to do their job must be able to access multiple databases and support multiple processes and are the bastion of legacy systems. In addition, these processes need to be easily modified as the business requirements change. For telecom repair personnel, an application like this could be used to help identify where a fault is located and what type of network elements are involved. To set the context for the scope of this problem, there are over sixty different telecom databases in existence from Bellcore alone storing valuable information about the network. An object-oriented database approach can shield this complexity providing the appropriate information to users.

Applications integrating medical information can provide doctors and the rest of the hospital staff with timely patient information. Research work at HP has demonstrated, using an OpenODB approach, that it is possible to hide the complexity of getting at many different types of medical information including patient data stored in MUMPS databases, X-Rays, laboratory tests, prescription drug information, and medical research results.

Laboratory Information Management Systems for pharmaceutical and chemical companies can provide a complete view of a new project tying together information from the initial tests, lab notebooks, and the resulting manufacturing processes. This information can be used to more effectively manage getting the government agency approvals on new products.

Oil & gas applications need to represent a complex business model involving oil wells, production rates, geographic locations, and geologic information. This data can be used to determine when and where to search for new deposits or manage the process of getting the raw resources to their refineries. The structure of this data has proven difficult to model in current databases. Currently integrating this data to make business decisions on land leases can require many days from the most experienced employees. This time could be reduced with an automated approach to integrating this heterogeneous data.

Any large decentralized company today has the challenge of keeping the financial model of the company accurate. As business units mutate, some dividing as they grow and others merging as the business focus changes, it is often difficult to get a true balance sheet representation of where the entire company is at any time of the year. Many hours of work go into consolidating the financial picture at the quarterly milestones and at the fiscal year end. Managing the business entities as objects and mapping these into the different sources of financial data (different in part due to mergers and acquisitions) would allow a Chief Financial Officer to see the impact of corporate structure changes even before they were consummated.

Developing computer software can be as complex as the previous examples. One type of tool being designed to make this task easier is a repository. A repository is supposed to provide a directory of data and services available in a computer network. Some companies building repositories today have recognized the power that an object-oriented database can provide as the underlying storage system. The ability to have coded functions in the database makes it possible for the repository to keep an active account of where key information is as it changes. Current repository products play a more passive role and have to be kept up to date by hand.

Is an object-oriented database right for your company?

Object-oriented databases are more complex to learn than relational or hierarchical databases. Therefore it is important to consider an ODBMS after having experience with existing databases. When the application domain is too complex to be supported by these products or the data needed for the complete picture is spread out in different locations and different formats, the OpenODB approach to object-oriented databases should be considered.

Douglas Dedo is the product marketing manager at Hewlett-Packard responsible for HP's object-oriented database program. Doug has held product marketing, marketing communications, and software development engineering positions within Hewlett-Packard's computer networking group. Previous to HP, he worked on networking and database projects as a software development engineer in the United States and Europe. Doug holds a B.A. degree in computer science from U.C. Berkeley.

PAPER #2025

HP-UX System Management with SAM

Douglas Drees,

Hewlett-Packard Co.

3404 E. Harmony Road

Fort Collins, Colorado 80525

(303) 229-3928

1. Abstract

Unix system administration presents numerous challenges because of the seeming haphazard mechanisms by which systems resources are managed. As Unix has moved out of the laboratory and into the commercial environment, vendors have used different strategies to reduce the difficulty and cost of administration. The HP-UX System Administration Manager (SAM) represents a unique set of trade-offs and has struck a balance which has allowed part-time, non-technical administrators to manage their systems. This paper describes SAM's design trade-offs and their impact on the part-time administrator. It also hints at the future of system and network management.

2. Introduction

Unix systems have moved off the campus and out of the laboratories into the technical computing market and are now moving into commercial environments. Along with this change has come a reduction in the technical expertise of those who must manage these systems. The old ad hoc tools and techniques used in the past to manage and administer them is no longer adequate.

SAM, the System Administration Manager, is a mature administration tool that solves many of the problems we have faced in the past and is poised to meet the challenges of single system administration in the future.

3. Yesterday's Problems

As Unix has moved out of Bell Labs into the universities, the technical computing markets, and, eventually, the commercial computing markets there were two forces at work that led to an unacceptable situation. The first was the way in which Unix developed - how new features and functionality were added. The second was the growing diversity in the user population and the applications they were using. These forces combined to yield a situation where fewer and fewer users had the ability to successfully manage their Unix-based computing systems.

a. A Diverse Development Environment

In the early years, development of the Unix environment was shared between Bell Labs and a number of universities and research centers. Their objectives and user communities were different, and so a variety of new capabilities were developed. But because this development was organizationally and geographically dispersed, there was no consistency or commonality maintained in these added features, and especially in the administration commands that were developed to manage them.

The tools and techniques required to add a new user to the system were completely different from those associated with adding new disks or establishing connectivity to new remote systems. No transfer of learning was possible from one tool to the next. The command line options were different and inconsistent, the output format differed, and there were no conventions regarding return values or error messages that could be counted upon.

The databases associated with these various features were also diverse in format, location, and usage. There was no single place to go to find out the current state of your system.

Common tasks required a number of sequential steps involving many different commands. This required the administrator to be familiar with a large suite of management tools. For example, adding a new disk to a system involved these steps:

⇨ Scan the system for available addresses (visual check, use bdf & mount -p, look in /etc/checklist, etc.)

⇨ Cable up the disk

⇨ Create device files (with mknod)

⇨ Initialize the disk (with mediainit)

⇨ Make a file system (with newfs/mkfs and /etc/disktab)

⇨ Edit the /etc/checklist file (with vi)

⇨ Mount the disk (with mount)

Figure 1: Adding a Disk using Commands

Such a common task required seven sequential steps, knowledge of at least five commands and two table formats, and some understanding of the overall process so that mistakes or problems could be handled if they occurred.

b. A Diverse User Population

To make things worse, at the same time Unix began to proliferate into the myriad of computer markets, the demands for the ease of management of Unix grew.

The average expertise of those responsible for managing these computers was going down. Downsizing and the availability or low cost multi-user and workstation Unix systems meant that more and more the professionals using the computers would have to be able to manage their own systems, much like the PC market.

Another problem was the growing number of these inexpensive Unix systems connected together in the office or lab. The sheer numbers overwhelmed the information processing departments and the support groups. It was too costly to keep hiring administrators to do the job.

Finally, the complexity of the systems themselves continued to grow. New security features were added. Disk performance was enhanced using arrays and logical volumes. Network communications were enhanced to provide a number of new features and capabilities that need administration.

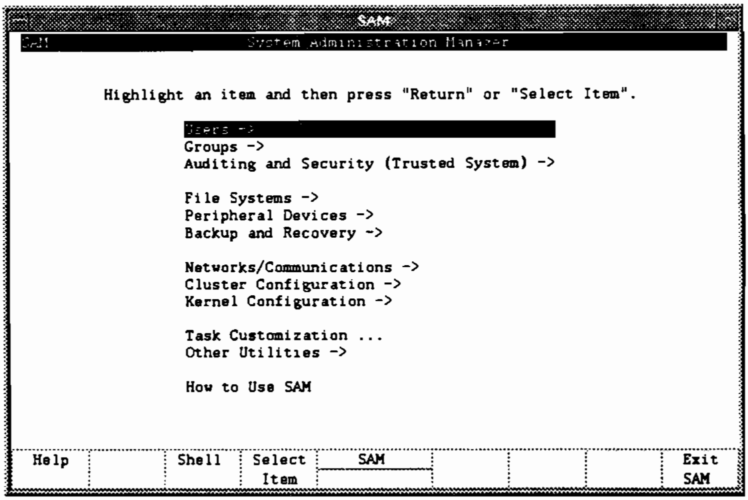
Clearly many of the non-technical users could not be counted upon to flawlessly perform the task outlined above each time it was needed. The secretary or office manager in a small branch office would be hopeless if faced with this situation.

3. The SAM solution

Unix system vendors have responded with integrated tools to address this problem. Tools such as IBM's SMIT, SCO's sysadmsh, and HP's SAM provide a one-stop shop for finding out the state of a single system, making necessary configuration changes, and generally managing such a system's resources. Third parties have also begun to market products that simplify the complexity and integrate a number of capabilities into a single tool or set of tools.

Additionally, network management tools have been developed that focus on the network rather than the computer. In time these may also meet the needs of managing the individual systems, too, but for now their features usually don't include the necessary tasks of the individual system.

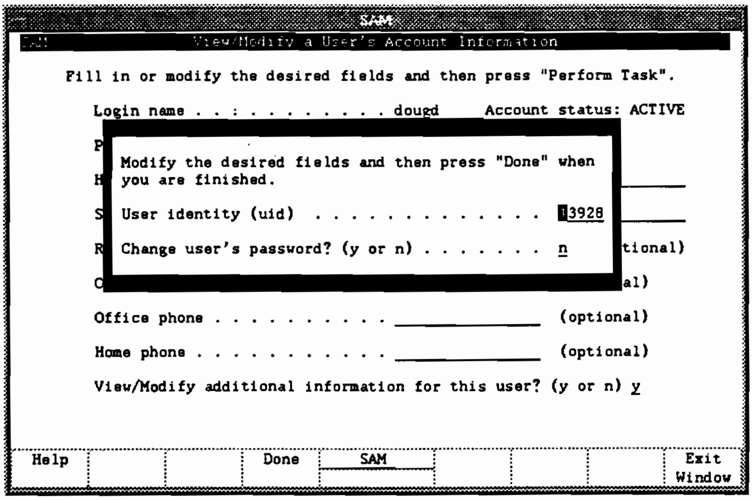
SAM, the System Administration Manager from HP, was uniquely designed with the less sophisticated administrator in mind. The focus of SAM is to enable the primary user or local manager of a system to manage their system directly. SAM provides a single place to come, a consistent interface, and a number of ease of use features to make the administrator's job easier.



Highlight an item and then press "Return" or "Select Item".

Figure 2: SAM Main Menu

The development of SAM has involved three significant trade-offs. The first trade-off was whether to provide user task orientation or to maintain a system oriented view. A user task might be adding a user or scheduling a backup. Whereas a system view would have tasks like modifying /etc/passwd or adding an entry to crontab. The user task orientation focuses on what the administrator is trying to accomplish. The system view focuses on what changes are required to the system. For SAM, we chose the user task orientation. This gave the novice administrator the best chance to succeed and also minimizes the new learning required for an administrator who knows some other OS and is now moving to HP-UX.



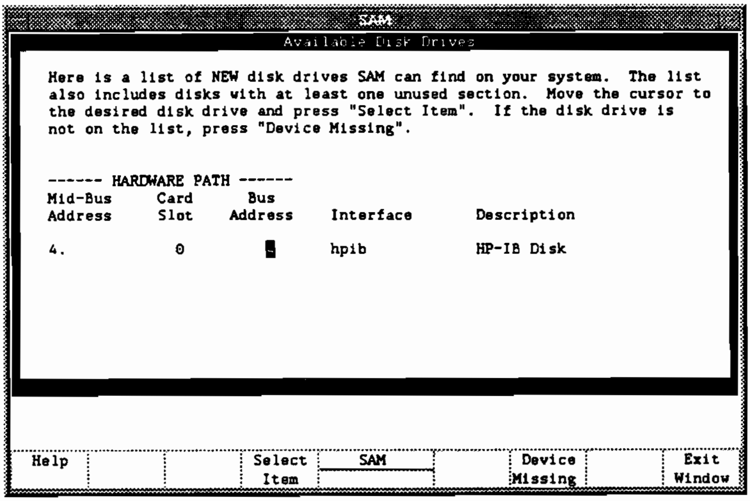
```
Fill in or modify the desired fields and then press "Perform Task".

    Login name . . : . . . . . . . . . dougd     Account status: ACTIVE

    P
      Modify the desired fields and then press "Done" when
    H you are finished.

    S User identity (uid) . . . . . . . . . . . . . 13928

    R Change user's password? (y or n) . . . . . . . n        tional)

    C                                                         al)

    Office phone . . . . . . . . . . _____________   (optional)

    Home phone . . . . . . . . . . . _____________   (optional)

    View/Modify additional information for this user? (y or n) y
```

Figure 3: Example of Optional Modify Defaults Screen

The second significant trade-off was whether to emphasize flexibility or simplicity. The HP-UX system allows a lot of variety in it's configuration. Many of the administrative commands have numerous options. Emphasizing flexibility would have meant providing access through SAM to all the possible options and configurations possible. While emphasizing simplicity meant reducing the number of possibilities and therefore the complexity of the tasks. SAM is designed to be as simple as possible while providing as much flexibility as is needed. For instance, in 7.0, SAM selected the **uid** for all added users and did not allow the administrator to change that value. This simplified the task somewhat because novice administrators did not have to learn how to select a **uid** because SAM would do it for them. In 8.0, we decided to add the ability to override SAM's selection, but put it in an optional screen (See Figure 3). Here in our lab, we always override SAM's selection so we can have a consistent set of **uid**'s across a number of systems. But the novice administrator is still not required to learn about **uid** selection.

The third trade-off was whether to pursue usability over functionality or vice versa. SAM was developed with the priority on usability. This meant that we have been slower than others to implement new administration functionality. But with every new capability we add, there has been a lot of usability testing, prototyping, and iteration to try to assure the highest success rate for the broadest user base.

The first SAM, in HP-UX 3.1, was the first step. It encapsulated many of the steps involved in administrating HP-UX systems and provided a consistent interface for those steps. In HP-UX 7.0, it picked up a more task oriented feel, collecting all the information to do a complete task and then performing all the steps automatically. In HP-UX 8.0, SAM picked up the capability to scan the system so that administrators could merely pick an object from a list rather than having to enter a lot of information from memory.



```
                              Available Disk Drives

   Here is a list of NEW disk drives SAM can find on your system.  The list
   also includes disks with at least one unused section.  Move the cursor to
   the desired disk drive and press "Select Item".  If the disk drive is
   not on the list, press "Device Missing".


   ------ HARDWARE PATH ------
   Mid-Bus      Card       Bus
   Address      Slot       Address      Interface        Description

   4.            0                      hpib             HP-IB Disk
```

Figure 4: Selection List for Available Disks

All along, the focus has been clear - make it easier for the average user to administer their own system. Such features as selection lists, simple form filling, and context sensitive help have been added to reduce the amount of relearning and memorization required. Tasks have been restructured and messages reworded to increase the likelihood of success.

The task of adding a disk with SAM is now just three steps:

⇨ Use SAM to determine an available address/interface to use

⇨ Cable up the disk

⇨ In SAM, select the newly attached disk from a list, fill in a form, and stand back while SAM does all the work

Figure 5: Adding a Disk using SAM

SAM in 8.0 now has the ability to manage users and groups, backup and recovery, cluster creation and configuration, disk and filesystem configuration, peripheral device configuration, print spooler management, network configuration for the single system, trusted system administration, and kernel configuration.

4. Maximizing Your Use of SAM

In 8.0, some new features have been added to enhance the use of SAM as an easy to use, one-stop administrative tool.

a. Logging

Late in the 8.0 development program, we decided to add a logging capability to SAM. That capability was developed too late to be documented in the administration manuals and had not been thoroughly implemented or tested at the time of release, so we kept it quiet for awhile. We have now thoroughly tested SAM logging and want you to begin to use it.

The only documentation that exists is in the shell script used to launch SAM, /usr/bin/sam. There are 4 levels of logging supported in SAM: none, summary, detail, and verbose. Look through that script and you can read a little about configuring the logging level. By default, SAM logs at the detail level. The logs go in /usr/sam/log/samlog.

Although the implementation in 8.0 is not complete - there are some actions taken by SAM that are not logged - it does cover most of SAM's actions and can be useful for debugging some weird failure or understanding just what SAM does.

We use logging as a debugging tool to help us track down where SAM is failing in development.

b. Task Customization

In 8.0 and later releases, users of SAM can add to the actions SAM takes for four of its tasks: adding users, deleting users, adding cluster clients, and deleting cluster clients. Scripts can be written that can be performed either before or

```
S:****** Adding local printer "tsm_test" on device
 "/usr/spool/lp/tsm_test".
D: Executing (pid = 12752):
 /usr/lib/lpshut
D: Child (pid = 12752) terminated with status 0x0
D: Executing (pid = 12754):
 /usr/sam/bin/mktsmpipe.sh /usr/spool/lp/tsm_test
D: mktsmpipe.sh: Executing: /bin/mknod /usr/-
spool/lp/tsm.pipes/tsm_test
D: Child (pid = 12754) terminated with status 0x0
D: Executing (pid = 12762):
 /usr/lib/lpadmin -ptsm_test -mdumb
 -v/usr/spool/lp/tsm.pipes/tsm_test -g0
D: Child (pid = 12762) terminated with status 0x0
D: Executing (pid = 12769):
 /usr/lib/accept tsm_test
D: Child (pid = 12773) terminated with status 0x100
S:****** Successfully added printer "tsm_test".

S:****** Starting and/or resetting lpspool scheduler.
D: Executing (pid = 12775):
 /usr/lib/lpsched
D: Child (pid = 12775) terminated with status 0x0
S:****** Successfully reset lpspool scheduler.
```

Figure 6: Example of SAM Logging Output

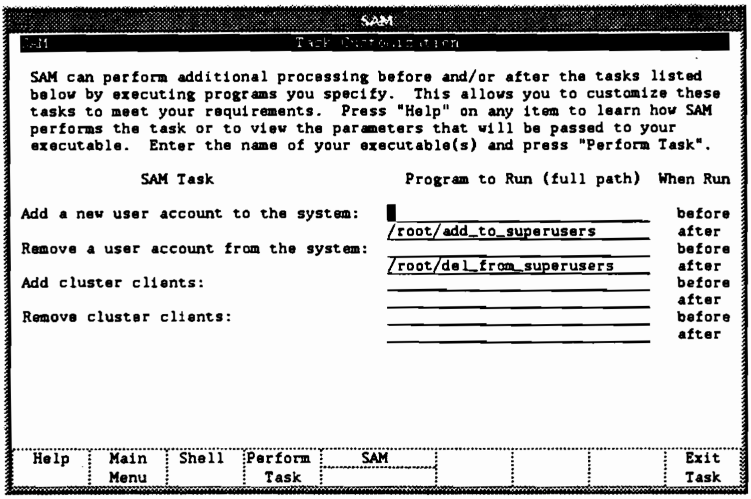
after each of these tasks. This can be useful if your environment requires other actions to be taken when performing any of these tasks.

In the SAM test cluster, we have a program that allows users to become root by entering their own password. This program excludes all users except those listed in a protected file. Since all users in the test cluster need to be able to run SAM and do root tasks as a matter of course, we have written a script that adds the users name to the protected file and have configured SAM to invoke this script after it has added a user. A similar script is configured to remove users from the protected file when users are deleted.

c. Other Utilities

SAM can now be extended somewhat. You can hook into SAM's menu hierarchy additional utilities that you have written yourself or that perhaps came with some application you are using. The file, /usr/sam/config/other_utils can be edited to add to the menu structure and instruct SAM which other utility programs that should be run. These applications or scripts can be interactive.

As we improve SAM in the future, we will preserve your investments in both the other utilities area and in task customization. File formats may change, but it will be done automatically if need be.



```
SAM can perform additional processing before and/or after the tasks listed
below by executing programs you specify.  This allows you to customize these
tasks to meet your requirements.  Press "Help" on any item to learn how SAM
performs the task or to view the parameters that will be passed to your
executable.  Enter the name of your executable(s) and press "Perform Task".

            SAM Task                          Program to Run (full path)  When Run

Add a new user account to the system:     ____________________________  before
                                          /root/add_to_superusers_____  after
Remove a user account from the system:    ____________________________  before
                                          /root/del_from_superusers___  after
Add cluster clients:                      ____________________________  before
                                          ____________________________  after
Remove cluster clients:                   ____________________________  before
                                          ____________________________  after
```

Figure 7: Task Customization Screen

d. Remote Administration

In 8.02 and 8.06 on the 800, SAM has the ability to set up a remote LAN connection to another 800 and allow the administrator to remotely administer the other system. There is no magic here - we are really just using rlogin to run SAM on the other system. But this feature covers up some of the complications of setting up such a link.

5. The Next SAM

We are just completing work on a new version of SAM to be released soon. SAM has been completely overhauled and will include the following improvements:

Motif Compliant UI on both X Window and terminals

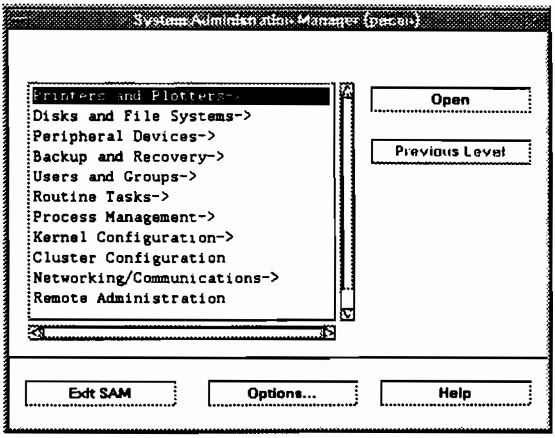
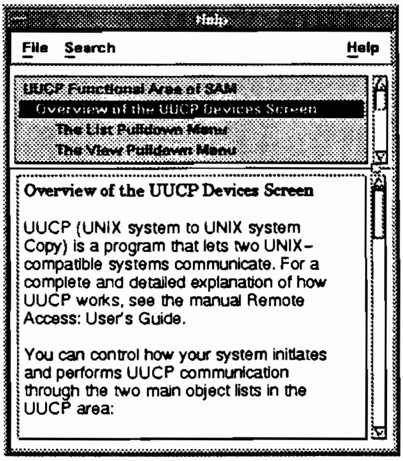


Figure 8: New SAM Main Menu

Context Sensitive Help with Hyperlinks


Help
File
Search
Help
UUCP Functional Area of SAM
Overview of the UUCP Devices Screen
The List Pulldown Menu
The View Pulldown Menu

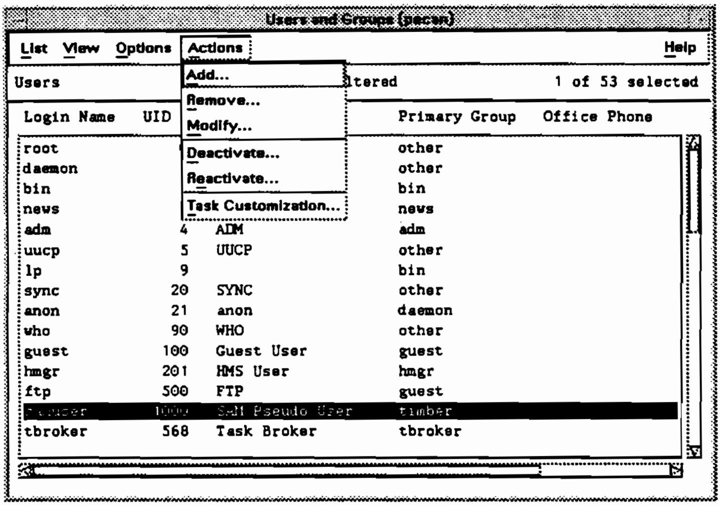
Overview of the UUCP Devices Screen

UUCP (UNIX system to UNIX system Copy) is a program that lets two UNIX-compatible systems communicate. For a complete and detailed explanation of how UUCP works, see the manual Remote Access: User's Guide.

You can control how your system initiates and performs UUCP communication through the two main object lists in the UUCP area:

Figure 9: Integrated VUE Help

Object-Action Behavior

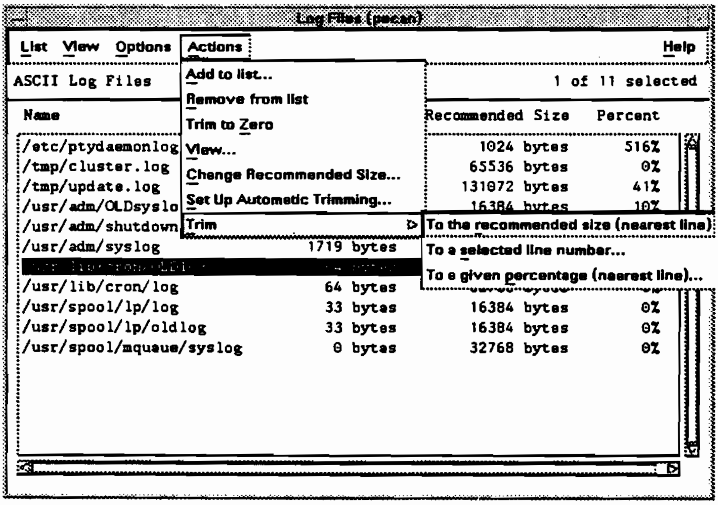


Users and Groups (pecan)

```
Users                                   ltered           1 of 53 selected

Login Name    UID                       Primary Group   Office Phone
root                                    other
daemon                                  other
bin                                     bin
news                                    news
adm              4    ADM               adm
uucp             5    UUCP              other
lp               9                      bin
sync            20    SYNC              other
anon            21    anon              daemon
who             90    WHO               other
guest          100    Guest User        guest
hmgr           201    HMS User          hmgr
ftp            500    FTP               guest
[illegible]user 1000  SAM Pseudo User   timber
tbroker        568    Task Broker       tbroker
```

Figure 10: User as Objects to Manipulate

Log File Trimming and Disk Hog Search



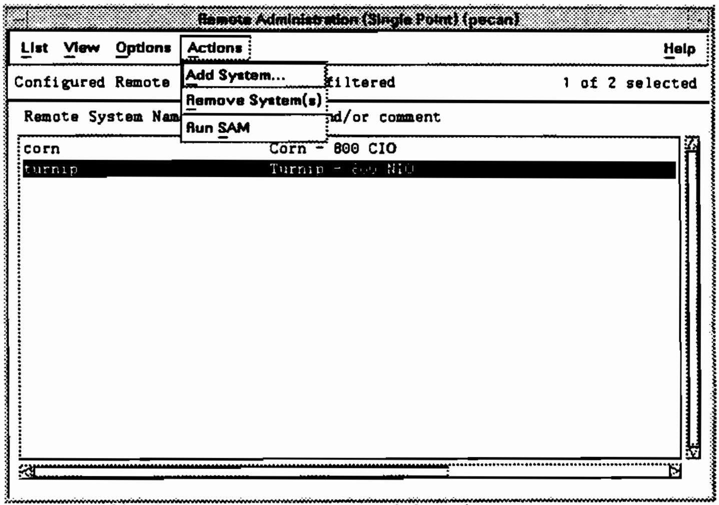
Log Files (pecan)

List
View
Options
Actions
Help
Add to list...
Remove from list
Trim to Zero
View...
Change Recommended Size...
Set Up Automatic Trimming...
Trim
To the recommended size (nearest line)
To a selected line number...
To a given percentage (nearest line)...

```
ASCII Log Files                                       1 of 11 selected

Name                                     Recommended Size    Percent
/etc/ptydaemonlog                             1024 bytes        516%
/tmp/cluster.log                             65536 bytes          0%
/tmp/update.log                             131072 bytes         41%
/usr/adm/OLDsyslo                            16384 bytes         10%
/usr/adm/shutdown
/usr/adm/syslog                 1719 bytes
[illegible]
/usr/lib/cron/log                 64 bytes
/usr/spool/lp/log                 33 bytes   16384 bytes          0%
/usr/spool/lp/oldlog              33 bytes   16384 bytes          0%
/usr/spool/mqueue/syslog           0 bytes   32768 bytes          0%
```

Figure 11: Log File Trimming Options

Single System Remote Administration



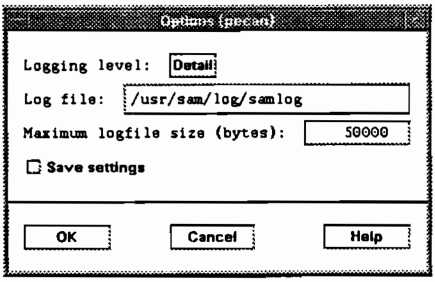
Configured Remote ... filtered 1 of 2 selected

Remote System Nam... ...d/or comment

```
corn                    Corn - 800 CIO
turnip                  Turnip - 800 NIO
```

Figure 12: Single System Remote Administration

Logging (including selection of levels)



Logging level: Detail

Log file: /usr/sam/log/samlog

Maximum logfile size (bytes): 50000

□ Save settings

Figure 13: Logging Options

And other improvements such as:

- ⇨ Add/Remove Tape Drive
- ⇨ Process Management (including kill capabilities and cron management)
- ⇨ Remove Terminal or Modem
- ⇨ Support for Optical Libraries for Disk and Backup/Recovery Use
- ⇨ Support for Network Printers

6. Some Hints about the Future

The future of Unix system management will include improvements along two fronts. The underlying system will become easier to manage as new kernel and network technologies are applied. Dynamic kernels will reduce the need to reboot to modify the kernel. Simplified clients that depend upon servers for the more complex functionality will require less individual management.

Also, the technology available to manage systems and networks of systems will continue to improve. OSF's DME and HP's Openview are notable examples of the kinds of technology that will be available in the coming years.

And HP is investing heavily in both of these directions. As a company, we will be working to meet the needs of the network administrators as well as the users/administrators of individual workstations or small systems.

Benefiting from Object Technology in a Distributed Environment

#2026
Jeff Eastman
Open Systems Software Division
447-4727

To gain maximum benefit from Distributed Computing, users will need a blend of new and existing applications which can work together and which can take advantage of their distributed nature. Object-oriented software technology offers a modern and comprehensive approach to blending old and new applications into value-added composites which exhibit this truly exciting potential. This paper shows how the various object-oriented technologies can be integrated with traditional technologies to enable customers to grow smoothly into the distributed era.

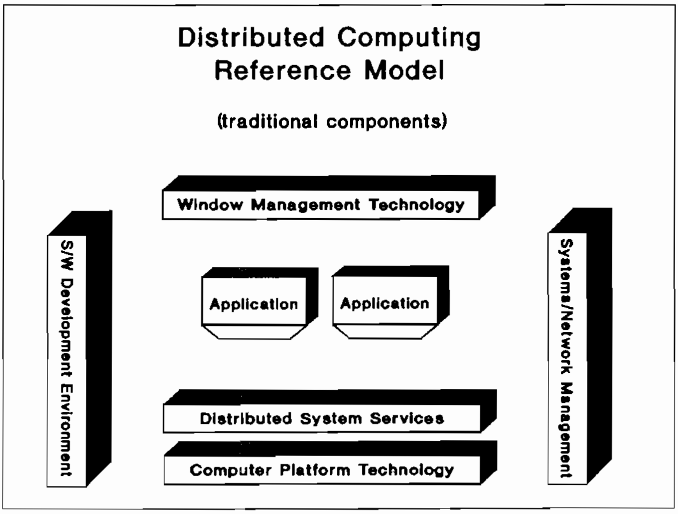
1. Introduction

To gain maximum benefit from distributed computing, users will need a blend of new and existing applications which can work together and which can take advantage of their distributed nature. Object-oriented software technology offers a way to blend old and new applications in a distributed computing environment. Choosing the right approach for an individual situation, however, is complicated by the lack of standardized object terminology, and by the often-conflicting claims of the various object technology vendors. HP is making a concerted effort, both through various standards organizations and within our own product offerings, to present a clear and unified model to our customers. This paper describes our model of how to apply the various object-oriented technologies most effectively as we help our customers grow smoothly into the distributed computing age.

The following illustrations present our high level reference model of the components involved in a typical distributed computing environment. In the first figure, the black shadowed boxes represent the components which are most closely associated with traditional computing methodologies. These will be outlined first in order to set the stage for the introduction of object technology. In the second figure, these traditional distributed computing components are augmented by the components which are most closely associated with object-oriented technologies.

Computer Platform Technology - CPUs, memory systems, and peripherals occupy this box along with their associated operating systems and OS-specific tools. The software in this box is designed to provide the basic facilities for operating and managing the underlying hardware configuration. In the early days of computing, these components were the only products offered by platform vendors.

Window Management Technology - Interactive applications spend much of their energy processing user inputs and generating dynamic displays. Window management systems such as X, Microsoft's Windows (TM), and the Apple Macintosh (TM) window manager provide application developers with a standard set of services for managing user interaction which is independent of the specific details of the underlying display hardware.


Distributed Computing
Reference Model
(traditional components)
Window Management Technology
S/W Development Environment
Application
Application
Systems/Network Management
Distributed System Services
Computer Platform Technology

Distributed System Services - The advent of computer networks fostered a plethora of products to allow systems to communicate information among themselves. Early products in this category were proprietary networking solutions designed to enable distribution of a single hardware vendor's products. The OSF DCE standard is the result of a multi-vendor effort to provide application developers with a standard set of services for managing distribution which is independent of the specific details of the underlying communication software and hardware. DCE is built upon HP/Apollo's Network Computing System (NCS), and other subsystems. This component will be explored in more detail in the next chapter.

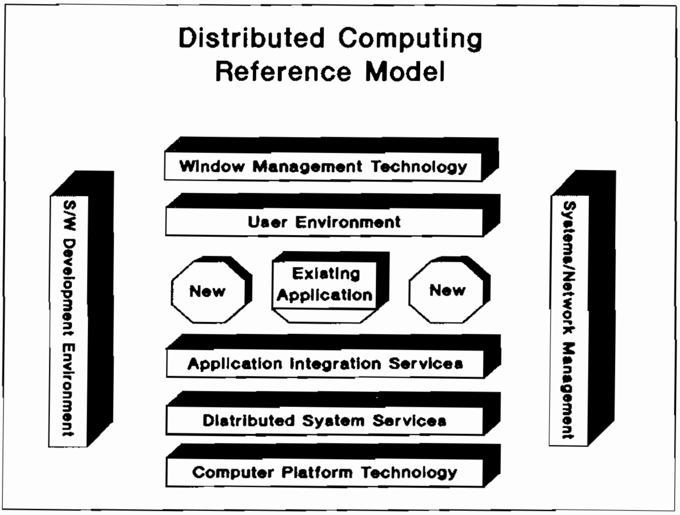
Software Development Environment - This component contains the programming languages and language specific tools which application developers use to create their applications. As applications have become larger and more complicated, and especially as they are becoming distributed, the adequacy of the services provided in this area has become more and more critical. HP SoftBench is an example of this kind of product. The final chapter will address this component in terms of its role in object-oriented application development.

System Network Management - While Distributed System Services allow applications to communicate information in a uniform manner, that component does not address the administration of distributed systems and applications. HP OpenView is an example of a current product in this category. The OSF's emerging DME standard is a multi-vendor effort to provide a uniform set of platform independent services to address this need.

Existing Applications - Existing applications must continue to function in a distributed context. In order for users to benefit from distributed access to these applications, however, they must often be embedded in a richer environment which facilitates this kind of use.

The following figure adds three new components to the previous model. These components are those which are most closely associated with object-oriented technologies.

New Applications - While today's applications must continue to function in a distributed environment, they cannot be made to deliver the true potential of distributed computing to their users. New applications which must be created to deliver these benefits fall into this category. This is the main area where object-oriented languages and methodologies can be applied.


Distributed Computing
Reference Model
Window Management Technology
User Environment
New
Existing Application
New
Application Integration Services
Distributed System Services
Computer Platform Technology
S/W Development Environment
Systems/Network Management

Application Integration Services - These services provide the richer environment which is necessary to allow embedded applications to function in a distributed topology, plus those required by new applications to deliver the full potential of distributed computing. HP Software Integration Sockets is an example of a current product in this area. New services which support object-oriented development will be discussed in the chapters which follow.

User Environment - While Window Management Technology can elevate the application developer to a hardware-neutral service layer for construction of interactive applications, that component does not provide for behavioral uniformity ("look and feel") between different applications and platforms. A User Environment provides a standard set of services which promotes inter-application uniformity and which improves the learning curve of users within the environment. HP VUE is an example of a current user environment product on the HP-UX platform.

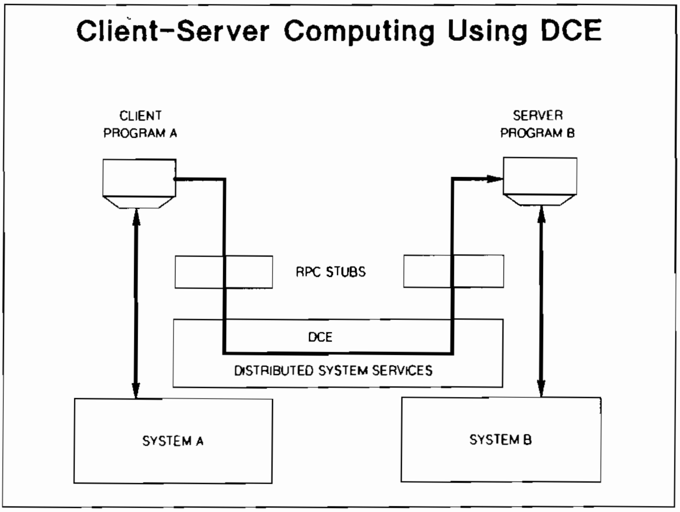
2. Client-Server Distribution

The OSF Distributed Computing Environment [1] supports a client-server model of distribution between traditional computing entities: programs, files, databases, hosts, and peripherals. It provides essential services for security, distributed file access, diskless support, distributed time management, distributed directory access and remote procedure calls between programs which reside on different systems. In order to understand how object technology can be employed in the DCE framework, we need to look deeper into its remote procedure call (RPC) mechanisms.

In this framework, a server program must first publish a formal description of the services which it provides. This is done by writing an *Interface Definition* in the DCE Interface Definition Language. IDL allows the programmer to specify the data types and procedures that a client must understand to request remote services from the server. These definitions are processed by the DCE IDL compiler to produce header files for use by the client and server programs. A client program wishing to obtain a service from a server must be compiled with the IDL header files of the interfaces supported by the server. These result in the automatic creation of local procedures, called RPC stubs, which can be called from the local program to access the remote service. Then, at run-time, the client must establish a network connection with the server program running on the server's host. Once the connection has been established, the client may make a local procedure call to the RPC stub procedure for the particular service which is required.

The client's RPC stub procedure takes the parameters of the request and encodes them into a standard data format using a process called *marshalling*. The marshalled parameters are bundled into one or more network packets and forwarded to the server's host using DCE routing mechanisms. Once at the server end, the packet header is decoded to determine the correct server stub procedure to invoke. The server stub procedure performs the decoding of the parameters in an *unmarshalling* process which converts the data formats to those used by the local hardware. Once in native mode, the parameters can be passed to the server to perform the desired service.

Results produced by the server are transmitted back to the client using an analogous process: The server stub marshalls the results and bundles them into packets which are then unmarshalled by the client stub procedure. The whole process is mostly transparent to both the client and the server programs, since they each appear to be making and responding to local procedure calls.


Client-Server Computing Using DCE
CLIENT PROGRAM A
SERVER PROGRAM B
RPC STUBS
DCE
DISTRIBUTED SYSTEM SERVICES
SYSTEM A
SYSTEM B

3. Remote Server Applications as Objects

By publishing an IDL interface, a server program makes some of its services available to clients using the remote procedure call mechanism outlined above. Exactly *how* a server provides its services is not a part of this interface description. Thus, a server is completely free to utilize any of its own host's local computing services in order to fulfill the requests of its clients. It may even turn around and become a client of some other server in order to fulfill these client requests. Since the effect of the interface definition is to hide all of these implementation details from the outside client world, the IDL interface is often said to *encapsulate* its server application.

Now, an *object* is a computational entity which is defined entirely by the services which it can perform. Each object also has an interface, which defines these services in terms of *operations* which it can perform. To get an object to perform one of its operations, a client makes a request of that object. Service requests include parameters and specify the desired operation. Once the object has completed this activity, a reply is returned which may contain results. These computations may be very complex, involving interactions with peripherals, non-object computer subsystems, and/or requests of other related objects. This is exactly the same principal as holds in an RPC system. In fact, from the point of view of a client, it is quite impossible to distinguish a server program using traditional software technology from an object employing a more modern software technology.

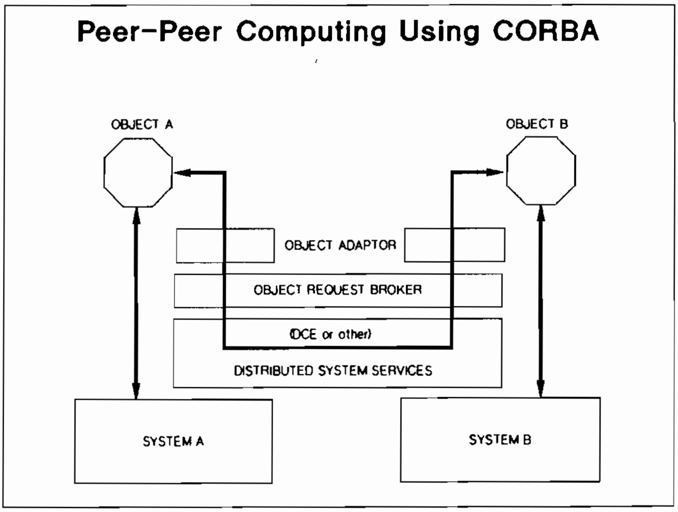
Object technology unifies data and procedure into objects which have a unique identity. In object-oriented systems, each object's identity is manifest in an *object reference* which can be used to reliably access the object during its entire lifetime. By holding references to other objects within themselves, objects are able to form rich networks which can be used to model complex application relationships with elegance. As the relationships are complex and often bidirectional, the respective objects cannot easily be cast into static client and server roles. For any given request, the requester is in the client role and the provider is in the server role. Since these roles may be easily reversed, objects are said to operate in a *peer-to-peer* capacity.

This places additional demands upon the underlying services which are not completely met by the DCE alone. As object-oriented technology is rapidly gaining momentum as the vehicle of choice for new application development, it is not unreasonable to ask what, if any, additions are needed beyond the DCE services in order to more completely support this new computing paradigm?

4. Object-Oriented Distribution

To address the need for object technology standards within the industry, HP and other companies formed the Object Management Group (OMG). The OMG's *Common Object Request Broker Architecture* (CORBA) [2] is the first of a series of open standards to be produced by this group. Heavily influenced by HP's experiences with our NewWave and NCS products, the CORBA defines a set of object management and distribution services which can augment the DCE to provide a modern object-oriented paradigm to software developers.

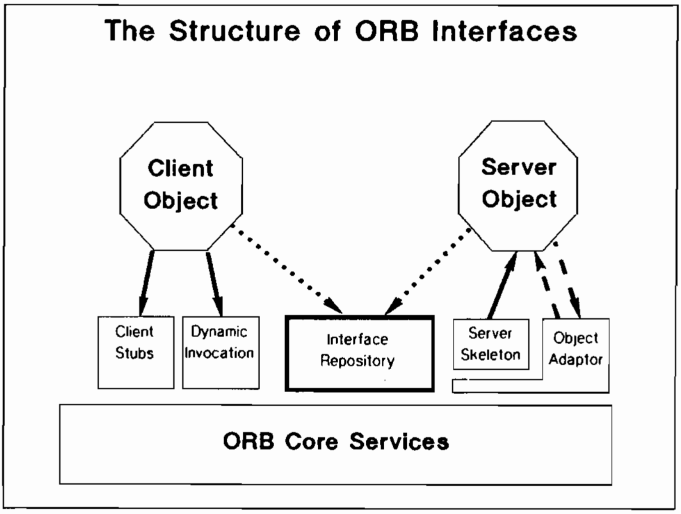
In any given interaction, the requesting object is in the client role and the providing object is in the server role. As in the DCE model, communication is via Remote Procedure Call, and the DCE RPC services may be directly employed for this part of the interaction. Other portions of the interaction, however, are automated by the *Object Request Broker* (ORB) and its different *Object Adaptors (OA).* These are discussed next.



Recalling that the client object holds an object reference of the server object, it will issue an operation request on that object reference. The actual request is handled by the client's Object Adaptor. To resolve the object reference into a host address, the OA first accesses the underlying ORB location services to determine the current host of the server object. Once this is completed, a connection is opened with the Object Adaptor on the server's host and the parameters of the request may be marshalled and packetized using the DCE transport layer. At the receiving end, the packet is passed to the server's Object Adaptor which is responsible for activating the target object, unmarshalling the parameters, and making the local procedure invocation. Results are communicated back through the Object Adaptors as before with the DCE RPC stubs. This is a very high level discussion of the object request mechanism which has purposefully ignored most of the details. In order to understand how various object-oriented technologies are supported by this architecture, we need to look deeper still. Readers who wish to skip over this further digression may skip directly to Section 7.

5. The Interface Repository

The following figure illustrates the object request mechanism in more detail. Here, the client and server sides of Object Adaptors in the interaction have been exploded to show the sub-components which must participate in the request. A new entity, the *Interface Repository*, has also been introduced to play an essential role.



The client object holds an object reference to the server object and wishes to request a service of that object. The first question which must be answered is whether or not the server object can actually perform the desired service. Information about the services which the server can perform is provided to the client in terms of its Interface Definition, in an interface definition language which is also called IDL. While this unfortunate redundancy of names is bound to create confusion in the marketplace, both IDLs fill a similar role in their respective systems. Fortunately, they are also closely related in their lineage: DCE IDL is identical to the HP/Apollo NCS 2.0 NIDL language, and the OMG IDL is a derivative of that language which incorporates additional object-oriented formalisms (e.g. interface inheritance and attributes). Thus it is probably correct to describe the OMG IDL as an object-oriented version of the DCE IDL.

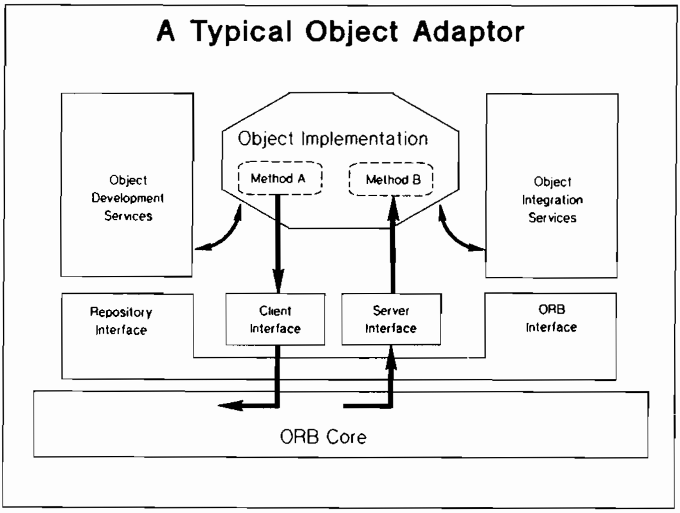
In order to make its services available to clients, the IDL specification of the interface of a server object must be disclosed to the developer of the client at some point prior to the client actually initiating the request. The Interface Repository contains the set of all interfaces which have been disclosed in this manner. In practice, there are two ways in which this information may be used: Statically and Dynamically.

> **Static Interfaces:** In the simplest situation where the interface of the server is known to the client object at its compile time, then this information can be used by the compiler to generate a *Static RPC Stub* to encode a request into the packet. The CORBA specification defines a Basic Object Adaptor which is suitable for compiled languages such as FORTRAN, C and C++ [5] where complete definitions of the server's interface can be processed at compile time to produce these Static Stubs.
>
> **Dynamic Interfaces:** The Basic Object Adaptor is inadequate, however, for languages and systems where interfaces are allowed to evolve over time, or where the interface of a particular object is not knowable until the request is actually made. HP's new OpenODB object-oriented database [3], object-oriented languages such as Smalltalk [4], and many 4th generation language systems share this characteristic. For these languages, Interface Repository information must be available at run-time to enable a request to be encoded. The CORBA specification provides for the existence of many different object adaptors. To support dynamic languages, it defines a *Dynamic Invocation* interface for run-time access to the Interface Repository which enables the dynamic creation of object requests.

At the client end, a combination of static and dynamic invocation mechanisms may be used in situations where some of the interface information is statically known but some is also dynamic. At the server end of the request, however, an object adaptor will typically provide a single mechanism, formally called the *server skeleton*, to dispatch the request to the server object. This mechanism may be implemented using either static or dynamic techniques and this choice will typically depend upon the implementation language environment of the server object.

6. Object Adaptors

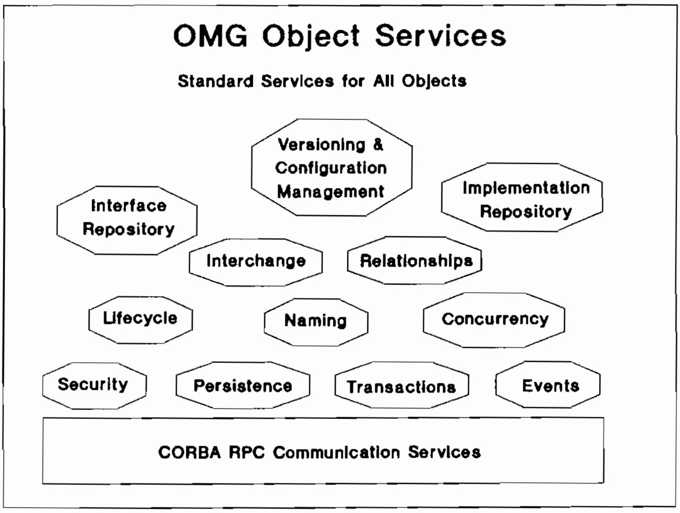
The Object Adaptor provides the interface between a set of object implementations and the ORB communication services. It also provides an interface to the Interface Repository which contains the client and server interfaces (stubs & skeletons) that its contained objects need to gain access to the outside world. The Object Adaptor also typically provides the execution environment in which a collection of objects operate. These objects usually share a common implementation language, and may have implementations which are highly interdependent. The figure below depicts a set of *Object Integration Services* which would be provided in a typical object adaptor product. These are Application Integration Services (in terms of our overall reference model) which relate to a given object adaptor, and which provide standard implementations for major application components.



In addition, an object adaptor product will typically also provide a set of *Object Development Services* which operate in close conjunction with the Interface Repository to provide a high productivity Software Development Environment (also from our reference model) for object developers. The next chapters will explore these two service areas in more detail.

7. Object Integration Services

RPC communication services are necessary, but not sufficient to enable meaningful interworking between objects. While the CORBA provides a mechanism for defining and utilizing interfaces, it does not define any particular application services which would enable useful distributed applications to be constructed. In recognition of this fact, the OMG has issued a request for technology in the area of common Object Services. These services will address the critical areas of application behavior which must be standardized in order to allow objects to engage in meaningful and consistent interactions with each other.


OMG Object Services
Standard Services for All Objects
Versioning & Configuration Management
Interface Repository
Implementation Repository
Interchange
Relationships
Lifecycle
Naming
Concurrency
Security
Persistence
Transactions
Events
CORBA RPC Communication Services

Object Lifecycle - Objects may be created, may alternate between activity and passivity for periods during their lifetimes, and may subsequently die. During their lifetimes, they may move from one host to another in response to their user's needs for security, availability, and interaction. They may be copied to produce new offspring, and may be shared in a variety of ways.

Object Persistence - The state of an object must persist over several activation and passivation cycles over the entire lifetime of the object. An object may initially be created on a laptop, for example, and live on a floppy disc in its users briefcase for several weeks before migrating to a department's server for a large portion of its active life. Subsequently, it may spend a long time in the company's central storage facilities where it is accessed infrequently before finally being archived or deleted. The object's state must be reliably maintained across these dissimilar systems.

Object Interchange - If I want to move an object from one host to another, there must be a way for the state of the object to be accurately transmitted. Since the state of an object is part data and part procedure, interchange mechanisms must be able to account for the possibility that the code which implemented the object's behavior on its old host is not available on its new host. Compromises must be made which do not degrade the object's usefulness after this process.

Object Relationships - Objects may hold references to each other to support a rich variety of application relationships. "Hot Links", as introduced in HP NewWave, provide a powerful way to share information objects within a single workstation. In the distributed era, users will be able to exploit this capability with links across systems giving them the most current information directly from the individuals who are charged with maintaining its accuracy and currency.

Object Concurrency Control - Since several clients may wish simultaneous access to services provided by a single server object, the possibility for deadlock exists. Database systems currently provide locking and concurrency mechanisms which are largely unsuitable for wide area distribution in a semi-reliable networking world. New techniques must be developed to provide the right kinds of guarantees and recovery strategies.

Object Naming - "A rose by any other name..." While objects know each other by their object references, users need ways of accessing them by human-readable names. This needs to be uniformly supported across many objects to be a useful capability. Object naming is closely related to services for query processing, and these need to be considered in conjunction with each other.

Object Transactions - The state of a system of objects must be maintained in a consistent manner despite failures which may occur from time to time. Database transaction mechanisms which are suitable for managing accounts receivable are unworkable in an environment which is designed to facilitate a high degree of sharing and collaboration between members of distributed workgroups.

Object Events - Object requests are communicated in a point-to-point manner using the ORB when the target object is explicitly known. Often the target is willing to accept delayed reports of changes in a server object. Some times, the same message needs to be sent to a large group of objects asynchronously.

Object Security - Access to an object's services must be subject to proper authorization controls. These must be uniformly implemented by solution providers to guarantee that loopholes will not exist which could be exploited by unauthorized users.

Object Versioning & Configuration Management - Object behavior must be able to evolve over time to support the need of a changing world. Object interfaces must evolve, yet only certain configurations of interfaces will work together. Old object instances must often be upgraded when new software is released which extends their behavior.

Interface Repository - The CORBA specification addresses only the minimal interface repository mechanisms required to support dynamic operation invocation. These mechanisms must be extended to meet many of the above needs.

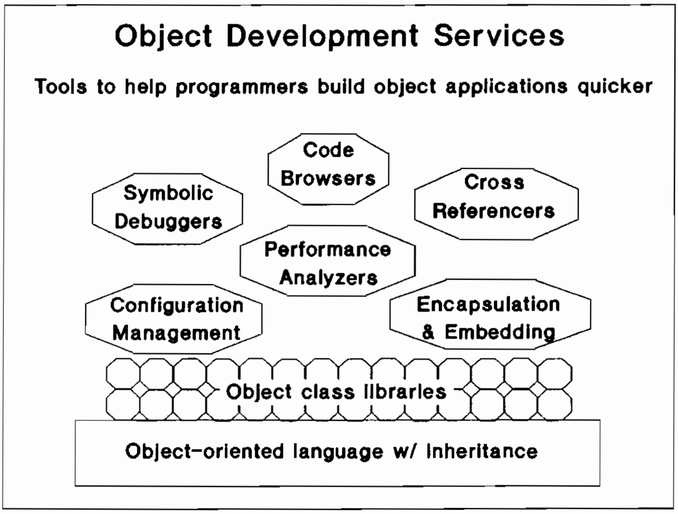
Implementation Repository - The CORBA specification addresses the need for an implementation repository, but defines nothing. Mechanisms must be defined and implemented.

We are presently engaged in a joint development program with SunSoft Inc. to produce compatible offerings of the DOMF core which will work across our respective UNIX systems. In addition, we are collaborating on the development of the above kinds of object integration services, so that applications may be developed for a variety of platforms which can interwork to a significant degree. This offering will be designed to meet the needs of static and dynamic language environments, and to foster the rapid development of distributed, object-oriented application software.

8. Object Development Services

Object-oriented development environments offer the prospect of improved productivity and maintainability for object-oriented application objects. In order to deliver these benefits, an environment needs to provide several key services with a high degree of internal integration. To support the development of object-oriented distributed applications, these environments must also be able to function as good citizens in the distributed object community as a whole.

Implementation Inheritance - The CORBA interface definition facilities address the behavior of objects in abstract terms i.e. with no assumptions about how the object is to be implemented. This is needed in order to allow objects in different Object Adaptors to implement the same services differently. In IDL, *Interface Inheritance* is used to allow services to be defined as independent interfaces and then combined in a variety of ways to produce the actual interface of a given object. In IDL, an interface may inherit from any number of other interfaces. If an object's interface inherits from another interface, then that object is required to provide an implementation of all of the operations defined in that interface. The object is free, however, to implement these operations using any technique available.


Object Development Services
Tools to help programmers build object applications quicker
Code Browsers
Symbolic Debuggers
Cross Referencers
Performance Analyzers
Configuration Management
Encapsulation & Embedding
Object class libraries
Object-oriented language w/ inheritance

Implementation Inheritance is a powerful technique which allows the implementation of a given operation, its *method*, to be automatically generated as well. While not required by IDL, an implementation inheritance graph which models the interface inheritance graph can yield significant code reuse benefits.

Class Libraries and Reuse Aids - Since the inner workings of an object are hidden from its clients by a procedural interface, objects are highly modular software components which can be reused extensively. By providing a rich set of already implemented objects, a modern software environment elevates the programmer to a higher level of productivity. Through a combination of implementation inheritance and a rich class library, the development of a new application object can be reduced to a minimum amount of effort.

Browsers and Cross-referencers - In order to allow programmers to make effective use of class libraries and inheritance, they must have rapid access to these components. This requires tool which support an overview perspective of a set of related implementation modules.

Symbolic Debuggers & Performance Analyzers - Debugging a distributed program is vastly more complicated than debugging a single program. Tools are needed which give programmers full access to the execution environment for debugging and for performance analysis tasks.

Version & Configuration Management - Object interfaces must be expected to evolve over time. It will also be the case that a given object version will only function correctly in the presence of other versions of its clients and servers. A configuration is a collection of interfaces which may be tested and guaranteed to work in conjunction with each other.

Encapsulating & Embedding - New and existing applications must be able to function together so that each customer's computing environment can evolve in a smooth and gradual process. Since an existing program and its associated data files appears to its clients to be very similar to a remote object, tools to fill in the remaining differences will allow existing applications to be smoothly integrated into a distributed object community.

HP's distributed object technology program is actively involved with our internal software engineering tools providers and with third parties to provide the kinds of tools which will be needed by our customers to realize the potential of object-oriented software development.

9. Summary

This paper has presented an architectural picture of the role of various object-oriented software technologies in creating distributed systems. It has shown how object technology is complementary with the OSF DCE and how it extends DCE's usefulness. Finally, it has shown how distributed object-oriented applications can be constructed with ease from common sets of reusable object services and integrated with existing applications as well.

Bibliography

[1] Open Software Foundation, *Introduction to OSF(TM) DCE*, Revision 1.0, December, 1991.

[2] Object Management Group, *The Common Object Request Broker: Architecture and Specification*, OMG Document Number 91.12.1, December 1991.

[3] Hewlett-Packard Company, *OpenODB Preliminary Developer Release Reference Document*, February 1992.

[4] Goldberg, Adele, and Robson, David, *Smalltalk-80 The Language*, Addison-Wesley Publishing Company, New York, 1989.

[5] Ellis, Margaret A., Stroustrup, Bjarne, *The Annotated C++ Reference Manual*, Addison-Wesley Publishing Company, New York, 1990.

Paper Number 2027

Security Aspects in a Backup Environment

Wolfgang Friedrich

Hewlett Packard
Herrenberger Str. 130
D-7030 Boeblingen
Germany

Tel. +49-7031-143754

Security Aspects in a Backup Environment

Introduction

One of the most important issues regarding data center management is security. Especially in a backup environment, the security issues go beyond the well-known aspects of physical, procedural, and system security.

In a backup environment, a security policy must ensure the data security and integrity of the online data and the data stored on the backup media. It is important to realize that on the backup media not only a piece of data can reside in one place, but also an entire collection of data, such as a file system or a database. Both theft of backup media and deliberate change of data on the backup media must be prevented. Another area of concern is the recovery of data: a single operator error during a restore operation could lead to the loss of a huge amount of online data.

These and others security issues specific to the backup environment are discussed in this paper.

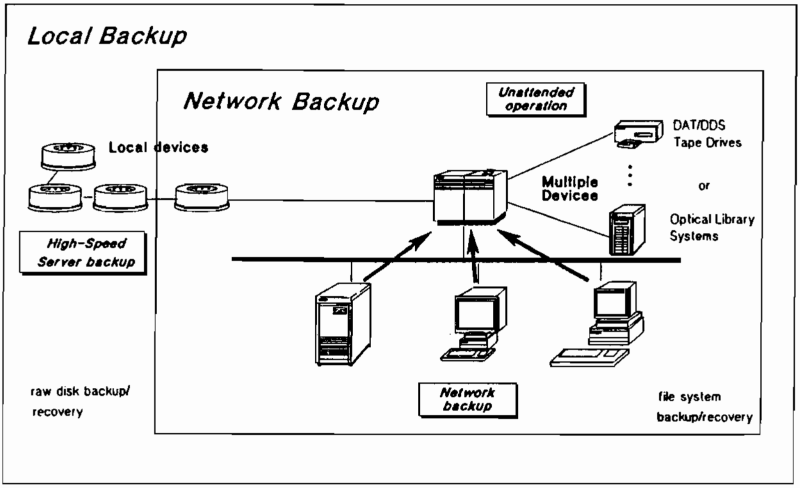
Backup in a Network Environment

Today's data center environments are becoming more and more complex. Past reliance on standalone systems meant that all system management tasks, including backups, were assigned to these standalone systems. Today, however, we find multiple systems linked through a network with applications distributed on the machines in the network. With this trend to networked environments, backup has also become very complex, due to increased sharing of resources, higher demand for system availability, and the growing number of systems at a site. Usually both operating and administrative staff are necessary to control and maintain the environment.

Typically, one can distinguish between two different ways of performing backups in a networked environment:

- backup of one machine standalone
- backup of multiple machines in the network

Backup in a Network Environment



Local Backup

Network Backup

Unattended operation

Local devices

High-Speed Server backup

Multiple Devices

DAT/DDS Tape Drives

or

Optical Library Systems

Network backup

raw disk backup/recovery

file system backup/recovery

The first approach, local backup of single machines, is the traditional way of doing backups. This approach is still useful for performing local high-speed backups or when the dependence on the network is considered too high. This local backup might be implemented by accessing the raw disk data for a better performance.

The second approach, network backup of multiple machines, relies on the network and offers a way of resource sharing in the sense that one needs fewer backup devices in the data center. A network backup is performed from a central system and might be implemented with a client/server concept. Different parts of the backup and restore processes run on the different machines in the network. PCs can also be included in the backup. Over time, multiple copies of data have to be stored in order to have a secure backup, which might result in a very large total capacity requirement.

Backup Devices

Backup devices must offer an easy, fast, and inexpensive method of performing frequent backups. In order to perform an unattended backup, the backup media must have sufficient online storage capacity, the transfer rate from disk to the backup device must be sufficient, and the media must be reusable and easy to store. Since backups are intended to preserve data, the media must offer good data integrity over a long period of time. Currently the most popular backup devices are:

- 1/2 inch 9-track tape drives
- 1/4 inch 16-track or 32-track tape drives
- rewritable optical disk drive
- DAT (DDS Format) tape drives

The 1/2 inch 9-track tape drives are typically very fast devices in terms of throughput of data. This relatively old technology is still very popular for exchanging data between machines of different vendors. Newer models offer online data compression, which leads to a maximum capacity per medium of up to 650 MB. For larger backups, manual intervention is needed to change the tapes.

The 1/4 inch 16-track or 32-track tape drives are comparatively slow devices. There are different tape formats from the different vendors. The maximum capacity per medium is up to 200 MB. However, autochanger mechanisms are available to load cartridges from a magazine automatically.

Rewritable optical disk drives are a relatively new technology. They offer 325 MB capacity per side, which adds up to 650 MB per medium. Autochangers are available for up to 94 GB of online storage. Unlike tape systems, where a file must be located before being accessed, rewritable optical disks offer direct access.

The DAT (DDS Format) tape drives are also a relatively new technology. Their capacity exceeds 4 GB per medium with drives that offer online data compression. The Digital Data Storage (DDS) recording format permits interchange of files with DAT drives of other vendors. An autochanger mechanism is also available on the market.

When deciding on a particular backup device, one has to consider aspects such as the potential for unattented backup, the capacity, ease of use, performance, etc. These aspects will have a direct influence on security issues for backup devices and media handling. In the latter part of this paper I will discuss these issues in detail.

Backup Strategies

The setup of a backup strategy is a very sensitive area for security. The system administrator is responsible for preserving the data stored on a system. The end user expects that the administrator has planned and implemented regular backup procedures to minimize the data loss.

To minimize the chance of data loss, all backup media should be stored at a different physical location from the file systems.

The exact backup procedure employed is affected by a number of factors. A heavily used system, both in terms of number of users and amount of activity, may require some form of daily backup. An infrequently used system may only need to be backed up weekly or bi-monthly.

Full Volume Backup Strategy

One type of backup procedure is copying the entire system on a regular basis. Every file, whether it has undergone change or not, is copied onto backup media, for what is often called a full volume or archive backup.

A full volume backup is only as good as the frequency of the copy. For example, if a full volume backup is done once a week, and no backups are done on the intervening days, any files created or changed from the time of the last backup until the next one are vulnerable. A full volume backup done every day provides maximum protection against data loss. Restoring a file system from a full volume backup consists of restoring from the most recent copy of the file system. On the other hand, full volume backups can be expensive in terms of the amount of media necessary, since every file is copied, regardless of whether it has undergone change. It might not really be necessary to copy every file on such a frequent basis.

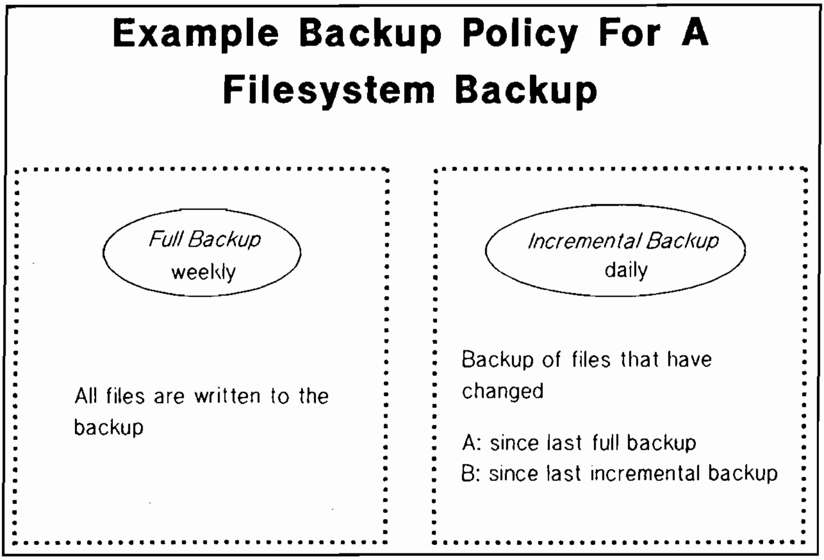
Incremental Volume Backup Strategies

Incremental backup procedures result in copies only of those files that have changed since the last backup. Incremental backups tend to be more difficult to design since they are data sensitive. However, since not as many files are copied as with full volume procedures, incremental backups usually takes less time and fewer media. Restoring an entire file system with incremental backups can be difficult, since each medium only contains copies of files that have undergone changes. Restoring an individual file is fairly easy as long as the user is able to determine the last time the file was changed, so that the correct medium can be identified.

There are two different kinds of incremental backup:

- incremental backup based on the last full backup
- incremental backup based on the last incremental backup

If the base is the last full backup, all files changed since the last full backup are backed up, even files that have been on a previous incremental backup. If the base is the last incremental backup, only the files changed after the last incremental backup are backed up.



Example Backup Policy For A Filesystem Backup

Full Backup weekly

All files are written to the backup

Incremental Backup daily

Backup of files that have changed

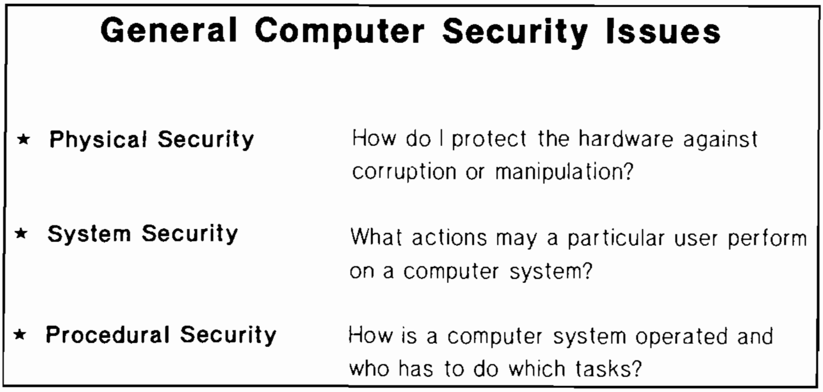
A: since last full backup
B: since last incremental backup

Mixing Backup Strategies

The ability to use both full volume and incremental backup procedures allows for the design of an effective backup strategy that is secure and cost-efficient in both resources and time. One common approach to backups involves an incremental backup of a file system every day and a full volume backup once a week. Using this type of mixed approach also provides a fairly simple way to restore the entire file system in the event of data loss.

For example, suppose a full volume backup is made on Monday and incremental backups are made on Tuesday through Sunday. Each incremental backup is based on the backup done the day before. If, on Thursday, the file system is lost, restoring a nearly complete copy is possible. First, the full volume copy from Monday is restored. Then the files from Tuesday's and Wednesday's incremental backup can be added to the restored file system.

In this situation, the only data lost are those files that were changed since Wednesday's incremental backup was performed. In the case where the incremental backups are based on the last full volume backups, the restore is even simpler. The operator restores the full volume copy from Monday and the incremental backup from Wednesday, since all files changed after the full volume backup are on Wednesday's incremental backup.



General Computer Security Issues

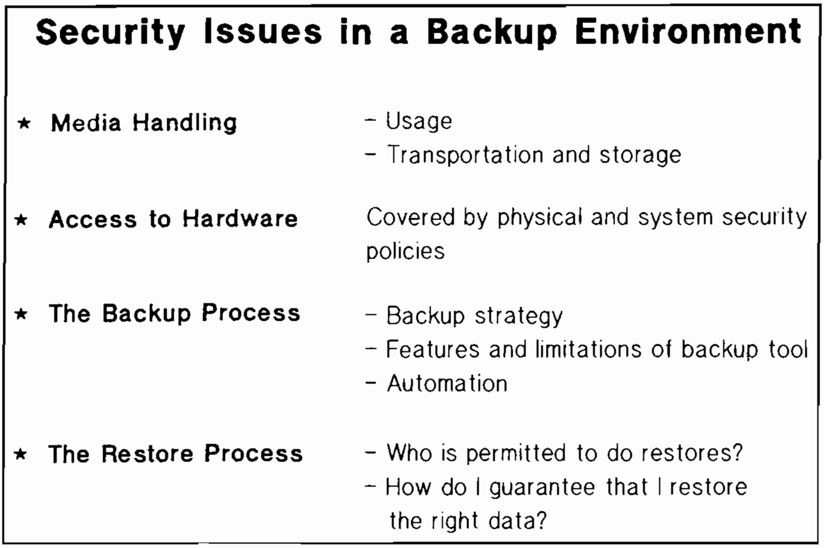
★ **Physical Security**	How do I protect the hardware against corruption or manipulation?
★ **System Security**	What actions may a particular user perform on a computer system?
★ **Procedural Security**	How is a computer system operated and who has to do which tasks?

In general, computer security aspects can be divided into 3 different areas:

- **Physical Security** addresses the security of hardware like the computer systems, terminals, disk drives, printers, backup devices, etc. Software is also addressed in the sense of outside corruption or manipulation of the data media. The corruption or manipulation of data on the computer disk is not part of **Physical Security**, but of **System Security**. A **Physical Security** policy must ensure the protection of computer resources against loss or corruption due to physical damage to computer equipment or physical damage to data media. The policy should define issues such as restricting access to areas containing system equipment or data media. Physical access to terminal or network cables must also be restricted.
- **System Security** addresses the question of what actions a particular user can perform on a computer system. Some users can be allowed to use specific computer devices, others might only have access to a couple of files, etc. In order to map permitted actions to a user, the following processes have to be carried out during login: The user must be identified by a unique login identity. The user must be authenticated by a valid password to confirm that the user is the one he claims to be. With this information the computer system must authorize a user for particular actions. The auditing of users and the actions they have executed must also be defined in the **System Security** policy.
- **Procedural Security** addresses the questions of how a computer system is operated, and who has to do specific tasks. Typically, a **Procedural Security** policy defines the role and the tasks of the system administrator and the operators. The **Procedural Security** policy also defines how new users are set up, how software is managed, how information is handled, how audits are performed, etc.

Security Issues Specific to a Backup Environment

The factors influencing backup security can be divided into 4 major areas, as shown in the following figure.



Security Issues in a Backup Environment

★ Media Handling	- Usage - Transportation and storage
★ Access to Hardware	Covered by physical and system security policies
★ The Backup Process	- Backup strategy - Features and limitations of backup tool - Automation
★ The Restore Process	- Who is permitted to do restores? - How do I guarantee that I restore the right data?

Media Handling

One has to consider that on one set of backup tapes there could be more than one unique data collection. In other words, a set of media could be a complete file system, database, etc or several file systems, databases, etc. If a set of backup tapes is stolen, sensitive data could be misused, regardless of whether the data is from software development or from a production environment. Misuse is more likely when the media can be mounted on other computer systems and even more so when the data format on the tape is a standardized format. A set of backup tapes might not only be stolen, but also destroyed or manipulated for whatever reason. These issues necessarily lead to an exact definition of how the backup media are transported and stored. The media must be stored in a safe place in a building physically separate from where the actual computer with the online data resides and also where any damage can be excluded. There must also be a process in place to delete backup media that will not be used for backup in the future. Imagine leaving a set of unused backup tapes somewhere–this is just like having top secret data lying around!

The media used for the backup must guarantee good data integrity and good shelf life. There should be a procedure to determine how often a specific medium has been used. Depending on the kind of medium, it should not be overwritten again after it has been used for a specific time.

Another issue of media handling is the definition of a process for labeling the media used for a backup. The media should have physical label stickers as well as virtual label information at the beginning of the tape. There must be a clear and simple way to identify a tape. If a tape holds more than one backup, special attention is needed. In this case some kind of "append" mechanism has probably been used by the backup product to exploit the whole capacity of the tape for multiple backups. The operator must be aware that during such a backup, the tape already holds some valuable data and that the tape should be treated appropriately. If the backup is unattended, care must be taken that unauthorized personnel cannot access that tape.

Access to Computer Machinery

Access to computer machinery, especially the backup devices, should already been addressed by the Physical Security policy. Physical access to the backup devices must be restricted and also access to the backup devices from the computer must be set up correctly. For example, on a computer system, if the device files are universally readable, virtually anybody can access that device. Any user could read a tape mounted on that device and could start the read process from any hardwired or virtual network terminal connected to that computer.

Another issue is access to configuration or logging information for the backup. With unprotected configuration data, an unauthorized user might see data about the file systems or database in the network. Also, with access to logging information of backups, an unauthorized user might find out where the backup data resides and then find an way to gain access to the data.

The Backup Process

The backup process must support a timely backup with minimal manual intervention. First, a backup strategy must be defined to reflect the users' needs and minimize the chance of data loss. Backup strategies have already been discussed at the beginning of this paper.

Before a backup process is in place, the system administration collects a list of requirements for the backup process. Based on this list, an appropriate backup product can be chosen. Another important issue is the question of backup devices. As previously mentioned, there are many different device types with different features on the market. To back up a large amount of data without any manual operator intervention, an automated library system, like the rewritable optical disk autochanger is appropriate. For network backups the appropriate network services must be in place. Performance over the network might also be an issue, if the network is heavily used by other applications. To restore single files from your backup media, a backup process based on the file system must be in place. To back up the file system based on raw data, the backup performance will be much higher, but you will not be able to recover single files directly. However, a raw disk backup may be the right solution for the backup of databases.

It is essential to ensure that data being backed up is not changed during the backup process. This is especially an issue with database backups. Here, either an online backup tool or shutting down the database before the backup are appropriate backup processes. For file system backups the operator needs information about all active files, since these files will not be correctly stored on the backup media.

For backup failures there must be a process in place to guarantee that the backup is started again in a timely manner. This process may be supported by the backup product or by guidelines for the operator. Whenever new machines or new disks are introduced there must be an easy way to integrate the new data into the backup. There must be a single person or group responsible for setting up the configuration of the backup process for an entity.

Another area of concern is the knowledge level of the personnel performing the backup, usually the operator. The operator must be able to use and understand the backup product and must also know its limitations. Required are a good understanding of the whole backup process which is built around the actual backup product and the ability to react correctly in abnormal situations. New processes should be introduced smoothly, step by step, to guarantee a consistent backup.

The Restore Process

Although the restoration of data might be considered an exceptional situation, this area is probably the most vulnerable one. A process must be in place to guarantee a fast and reliable restoration of single user data, as well as fast and reliable disaster recovery. A disaster can be a single disk crash or the loss of a whole computer system.

There must be dedicated personnel authorized and trained to do restores. If, however, a normal computer user is permitted to do restores, his restore capabilities must be very restricted. Before performing the actual restore there must be a well defined way to identify the medium that holds the data that needs to be restored. Usually, is done through the logging information made during the backup. This logging information might be available online on a computer system off-line on paper or on any other medium. No matter where the information resides, there must be an easy and fast way to access it.

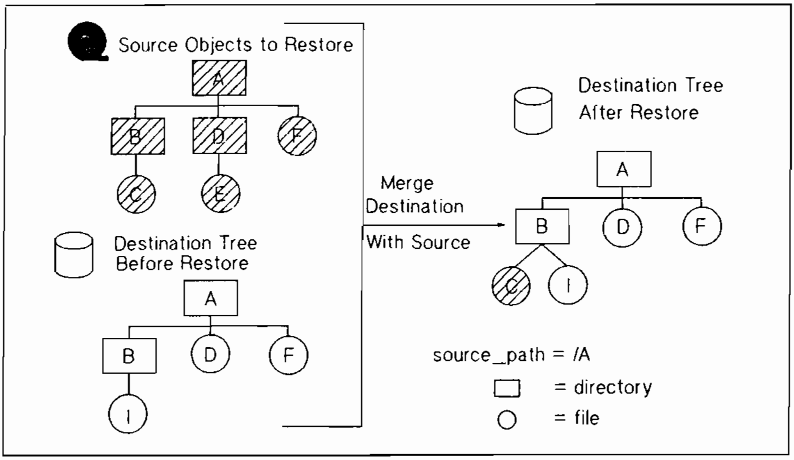
When the restore command is entered, the operator must be careful to restore only the precise data necessary. Otherwise, restoring less data will not satisfy the users, and restoring more data could have other negative side-effects. In addition, the data must be restored to the right location. Especially in a network backup environment, there are many ways of restoring to the wrong place. The data could be restored to a wrong directory or a wrong file system, and it might even be restored to the wrong computer system!

For example, when restoring a whole file system after a disk crash, a file system overflow could occur. This could be the case when the backup product does not keep track of deleted files. If all data from a full backup tape and the subsequent incremental backup tapes are restored without the operator taking note of the files that have actually been deleted from the computer's disk, a file system overflow might occur.

Another very sensitive area during the restore process is conflict resolution. A conflict exists if during the restore process objects are found on the backup tape that already exist on the computer's disk. Objects may be files or directories.

In the following scenario, directory A has the same underlying structure on the backup tape as on the computer's disk.

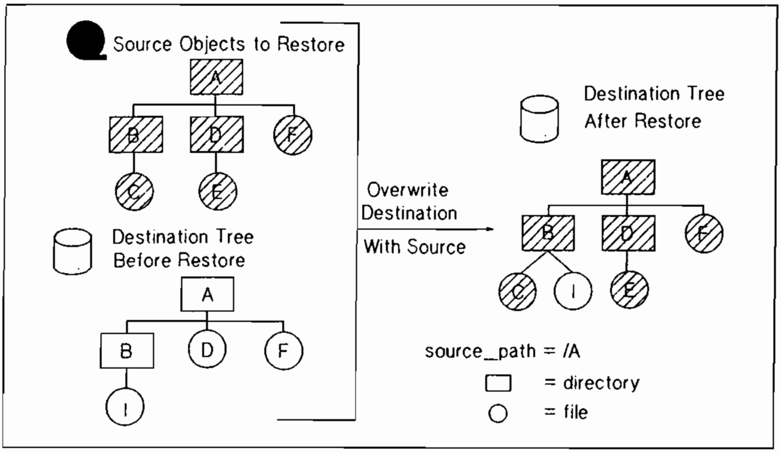
Conflict Resolution by Preserving Disk Files



To give preference to the file and directory structure on the disk, a backup tool must offer a way to do a restore, adhering to the following rules:

1. Files or directories that exist on the disk but not on the tape are left unchanged (file I).
2. Files with the same name on the disk as on the backup tape are left unchanged (file F).
3. Dissimilar objects with the same name are left unchanged. For example, if there is a directory on the backup tape and a file on the disk with the same name, the directory will not be restored (object D).
4. Files or directories that are on the backup tape but not on the disk are restored (file C).
5. If a directory with the same name exists on the disk as well as on the backup tape, rules 1-4 apply for the underlying files and subdirectories. The attributes of the directory on the disk remain unchanged (directories A and B).

Conflict Resolution by Overwriting Disk Files


Source Objects to Restore
A
B
D
F
C
E
Destination Tree Before Restore
A
B
D
F
I
Overwrite Destination With Source
Destination Tree After Restore
A
B
D
F
C
I
E
source_path = /A
□ = directory
○ = file

To give preference to the files and directory structure on the backup tape, a backup tool must offer a way to do a restore, adhering to the following rules:

1. Files or directories that exist on the disk but not on the tape are left unchanged (file I).
2. Files with the same name on the disk as well as on the backup tape are restored (file F).
3. Dissimilar objects with the same name are restored. For example, if there is a directory on the backup tape and a file on the disk with the same name, the directory will be restored (object D).
4. Files or directories that are on the backup tape but not on the disk are restored (file C).
5. If directories with the same name exist on the disk as well as on the backup tape, rules 1-4 apply for the underlying files and subdirectories. The attributes of the directory on the backup tape replace the attributes of the directory on the disk (directories A and B).

With these rules it is very easy to destroy a large amount of data on the disk. Just suppose that you restore a file called "/users" to a disk with a directory "/users". During the restore the whole directory "/users" with all underlying directories and files will be removed and replaced by the file "/users". In this case the system administrator really has to be careful!

There are also other sets of rules possible. The main issue is that the backup tool completely describes these rules, adheres to them, and that the system administrator knows their effects.

Conclusions

Security in a backup environment is more than another charge on the budget for data center operation. Some aspects, like the physical access to computer machinery, are already covered by the general security policies that should be in place at a well managed data center. However, there are issues specific to backup and restore operations that must be addressed separately. The main tasks is to analyze thoroughly the backup and restore requirements for a list which will influence the purchase of a backup product as well as for setting up a backup strategy.

After a backup strategy is in place, the whole backup process, as well as the restore process must run smoothly and a security policy must be in place. The overall goal for optimal security and good backup strategy is to ensure that data can be restored correctly in a timely manner. One of the most important issues here is that the operating staff completely understands the concepts of the backup product so they can exploit all its features. Another goal of the security policy is to make sure that the backup data, as well as any backup configuration and logging information, is stored securely, to guarantee that only authorized persons have access to it.

Paper #2028

Cutting the "Super" Hype--Evaluating a RISC Architecture

Linley Gwennap
Hewlett-Packard Company
19490 Homestead Road MS 42UH
Cupertino, California 95014
(408)447-6164

INTRODUCTION

Most major computer vendors have turned to RISC microprocessors to provide the heart of their latest systems, both desktop workstations and multi-user systems. Hewlett-Packard began this trend by introducing a line of PA-RISC minicomputers in 1986, and the latest convert has been Digital, which promises to use their new Alpha chips in all of their future systems. In between, Sun, IBM and several vendors using the MIPS chip have also adopted RISC.

With today's market crowded with RISC chips, vendors have started a debate over which one is the "best". Often, the debate revolves around esoteric technical arguments such as whether superscalar or superpipelined designs are better. Is 64-bit virtual addressing needed? Are high clock frequencies good or bad? Which SPECmarks are more important, integer or floating point?

Stepping back from these issues, a better question is how important are these technical specifications. In the end, the most important question is how fast (and how well) a system runs your application. In a sense, it doesn't matter what's under the hood if the system satisfies your needs. In situations where you cannot directly test your application, some of these technical issues can help predict how the system will do. In order to unravel the vendors' claims, you need an understanding of the terminology being used as well as a feeling for which issues are most important to your application.

This paper explains the issues and terminology you need to understand to make the right decisions. Several of the latest RISC processors are used as examples, including Sun's SuperSPARC chip, DEC's Alpha chip, MIPS' R4000, IBM's "America" chip set and HP's PA7100 microprocessor. An appendix provides additional technical specifications on these chips.

RISC MICROPROCESSORS

A decade ago, the central processing unit (CPU) of a minicomputer system usually occupied dozens of logic chips spread across multiple circuit boards. These minicomputer CPUs allowed many users to share a single system and could process large batch jobs. Unfortunately, these CISC (Complex Instruction Set) processors were big and expensive. At the same time, personal computers using Intel's 8086 chip were becoming popular. This "microprocessor" packed the functions of larger CPUs onto a single chip, but its lower performance was only suitable for desktop systems with a single user. Still, its small size and low cost enabled the creation of the PC market.

RISC (Reduced Instruction Set) technology is now used to build microprocessors that provide minicomputer (or even mainframe) performance. These RISC processors are both smaller and less expensive than their CISC minicomputer predecessors. They also improve reliability, since there are fewer chips which can fail. While similar in cost to CISC microprocessors such as Intel's 8086 family, they offer much better performance. Today's RISC chips take advantage of the latest advances in computer architecture to provide superior price/performance compared to CISC processors.

PIPELINES AND SUPERPIPELINES

One of the key differences between RISC and CISC processors is that RISC chips execute most instructions in a single "tick" of the clock, called a clock cycle. CISC chips usually take several clock cycles to execute an instruction. Since clock frequency measures how fast the CPU clock ticks, a RISC chip will therefore execute more instructions than a CISC chip if both are running at the same frequency.

Figure 1: Simple Pipeline

→ Instructions →

Fetch	Execute	Write
cycle 1	cycle 2	cycle 3

But this is not the only advantage of RISC. Because all RISC instructions take about the same amount of time to execute, a technique called "pipelining" is used to speed up the clock frequency. Pipelined processors are designed like a factory in which instructions are processed on an assembly line. For example, a simple pipeline with three steps (or "stages") is shown in Figure 1. Like the factory, these CPUs don't wait for one instruction to be completed before beginning the next; instead, instructions become overlapped with each other. Figure 2 shows instructions flowing through the three-stage pipeline. During clock cycle 3, for example, the first instruction is nearly complete, the second instruction is half-finished, and the third instruction is just starting. The figure shows that, although each instruction takes three cycles to complete, a new one is started each cycle. Nearly all RISC CPUs today are pipelined in this manner, although many use four (such as SuperSPARC) or five stages (such as the PA7100).

Figure 2: Pipelined Instructions

INSTRUCTION #1	Fetch	Execute	Write		
INSTRUCTION #2		Fetch	Execute	Write	
INSTRUCTION #3			Fetch	Execute	Write
clock cycle	1	2	3	4	5

Figure 3: R4000 Superpipeline

→ Instructions →

Fetch1	Fetch2	Read	Execute	Load1	Load2	Tag	Write

Superpipelining is an extension of simple pipelining. For example, the superpipelined R4000 uses an eight stage pipe (Figure 3). Adding stages to the pipeline means dividing the task of executing an instruction into more steps. For example, the R4000 divides the "fetch" activity from Figure 1 into two steps. By doing less work in each step, the assembly line can move faster. In a CPU, this speed in measured in megahertz (MHz) and is called the clock frequency. Thus, the net result of superpipelining is to increase the clock frequency by increasing the number of stages in the pipeline. The R4000 reaches a speed of 100 MHz using its superpipeline, compared to just 40 MHz for its predecessor, the R3000, which uses a five-stage pipe. DEC's Alpha chip uses superpipelining to reach even higher speeds, up to 200 MHz.

The difficulty of pipelining is that, unlike in the factory, many instructions require the results of a previous instruction before they can proceed. For example, if an instruction loads a number from memory into the CPU, and the next instruction adds that number to a sum, the load must finish before the add can start. In this case, the processor must stop and wait for the load to complete; both the R4000 and the Alpha chip must wait two cycles. This is called the "load-use penalty" and creates wasted cycles. By contrast, the pipelined PA7100 CPU has only a single cycle load-use penalty which is easier for the compiler to handle.

Figure 4: Pipeline Comparison

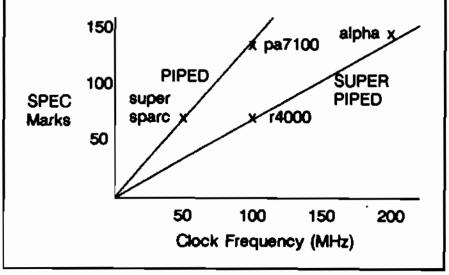
	PA7100	Alpha
Superpipelined?	No	Yes
Load-Use Penalty	1 cycle	2 cycles
Bad Branch Penalty	1 cycle	3 cycles

Another problem area for superpipelines is branches, which are decision points in the code. Because the outcome of the decision affects which instruction is loaded into the pipeline next, the processor must wait for the decision to be completed before starting the next instruction. Many processors try to overcome this problem by predicting the outcome of the decision, but when they guess wrong, the R4000 takes a two-cycle penalty, while the Alpha chip wastes three cycles. By comparison, the simpler PA7100 has a single-cycle penalty for badly predicted branches.

An often ignored problem for superpipelined CPUs is the development time. The biggest portion of the effort to develop a new CPU is verifying that it works correctly with all possible combinations of instructions. Adding stages to the pipeline means that more instructions are in the pipe at any given time, particularly in the critical execution stages, and the number of possible combinations increases geometrically. Because of this added test time, a superpipelined processor could take an additional 6-12 months to reach the market, which can be a major problem when competitive performance is increasing by 60% per year.

In summary, a processor can use superpipelining to increase its clock frequency and thus the speed at which it executes instructions. This increased speed, however, is not always available to the user. With more instructions in the pipe, a superpipelined CPU is like a juggler with too many balls in the air: it's hard to start, stop, and change directions. Significant penalties for loading data and branching, which can be a third of all instructions, reduce the efficiency of the CPU. For example, both the R4000 and the Alpha chip achieve about 0.7 SPECmarks per MHz, less than most scalar

Figure 5: Pipelining versus Clock Speed

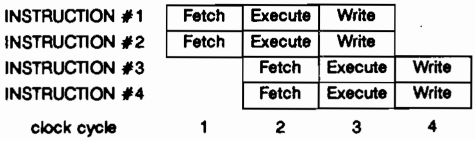


pipelined CPUs and much less than superscalar pipelined CPUs. At 200 MHz, DEC's superpipelined Alpha chip has just slightly better performance than the 100 MHz PA7100. As shown in Figure 5, superpipelined processors must achieve nearly twice the clock speed of pipelined processors to provide the same performance.

SUPERSCALAR PROCESSORS

Another technique that takes advantage of the similarity of RISC instructions is superscalar design. The term "superscalar" (derived from *scalar*, meaning "one-dimensional", thus superscalar meaning "multi-dimensional") refers to a design which uses two or or more pipelines, each capable of executing an instruction. A superscalar processor can execute multiple instructions in each clock cycle. Figure 6 shows the same three-stage pipeline from Figure 1, only this time in a two-way superscalar design. Each cycle, two instructions are started simultaneously and proceed together until they are completed. This is possible because all RISC instructions generally take the same amount of time to execute. Both the Alpha chip and the PA7100 are two-way superscalar, while SuperSPARC is able to begin up to three instructions per cycle and IBM America up to four.

Figure 6: Superscalar Pipeline



INSTRUCTION #1	Fetch	Execute	Write	
INSTRUCTION #2	Fetch	Execute	Write	
INSTRUCTION #3		Fetch	Execute	Write
INSTRUCTION #4		Fetch	Execute	Write
clock cycle	1	2	3	4

In theory, a two-way superscalar CPU could execute two instructions every cycle, but in practice there are many limitations, because instead of duplicating all of the functional units in a chip, most current superscalar designs only allow instructions to be grouped if they use different functional units. For example, the PA7100 includes an integer unit and a floating point unit, each of which can begin an instruction in the same clock cycle. In contrast, the scalar R4000 has both integer and floating point units but can use only one of them on each cycle.

Because of these hardware limitations, superscalar programming can be like ordering from a Chinese menu: only one from Column "A" and one from Column "B" per cycle. If the program needs to execute two instructions from Column "A", they must be done one after the other, just as with a scalar processor. Therefore, the performance of superscalar processors varies from application to application, depending on the mix of instruction types.

SuperSPARC gets around these limitations by including two integer arithmetic units on the chip. Furthermore, these units are cascaded so that data can flow through both units in a single cycle. For example, a number can be doubled and the result added to a sum by using two instructions which execute in a single tick of the clock. This feature helps improve the chances that SuperSPARC will execute two or even three instructions in any given cycle; in fact, the chip is expected to achieve about 1.4 SPECmarks per MHz, more than any other processor except the four-way superscalar America chip set. The tradeoff is that the duplicated arithmetic units help make SuperSPARC the largest, and thus most expensive, single chip microprocessor of the group.

In addition, remember that the objective of a superpipelined CPU is to divide the work into smaller steps, increasing the pipeline speed. SuperSPARC takes the reverse approach by combining two integer math operations into a single cycle, which has the effect of limiting the clock speed to a relatively low 50 MHz. As a result, SuperSPARC is not only costly, but it also has a slow clock.

IBM has also paid the price for a highly superscalar design. Their America processor can execute up to four instructions per clock cycle and averages 2.0 SPECmark per MHz. Like SuperSPARC, however, the complexity of this design makes the chip circuitry very large. In fact, the America design is so large that it requires three chips, compared to just one for other processors. Not only does this make America very expensive to manufacture, but it slows down the clock as well, because sending signals between chips is much slower than keeping them on a single chip. In other words, spreading the design across multiple chips has prevented IBM from increasing the clock frequency beyond 50 MHz. This limitation prevents America from reaching the performance of the PA7100 or Alpha and also drives up the cost.

HP's PA-RISC architecture provides an additional way to do more than one task in a single clock cycle. It includes some "multiple operation" instructions which actually do two operations in the same instruction. For example, instructions such as COMPARE AND BRANCH and ADD AND BRANCH allow mathematical operations to be performed at the same time as a branch instruction. PA-RISC also allows memory addresses to be modified during load and store instructions, and can begin a separate floating point add and a multiply using a single instruction. Because of these features, actual measurements* show that PA-RISC requires 30% fewer instructions than the sparser SPARC or MIPS instruction sets to complete the SPECmark benchmark. Because Alpha is very similar to MIPS, PA-RISC should have a similar advantage over Alpha, and a smaller advantage over IBM's POWER. These combination instructions supplement the simple superscalar design of the PA7100 to give it an efficiency rating similar to the more complex SuperSPARC, but at a far lower cost and without requiring recompilation.

The compiler plays a important role for superscalar processors, because it will try to arrange a program's instructions in the best order for the particular constraints of that processor. A good compiler will significantly improve superscalar performance. Conversely, a new superscalar CPU running an old program that has not been recompiled may provide relatively little performance improvement.

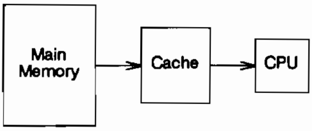
In summary, superscalar design allows a processor to execute two or more instructions together as a group. While in theory this can double or triple the "peak" performance, in practice superscalar designs today provide 20% to 80% improvement because of limitations in the hardware design. Even to achieve this improvement, code must be recompiled to take best advantage of the superscalar design. The PA-RISC architecture offers a richer instruction set that can substitute for complex superscalar design at a lower cost.

*See *Pathlength Reduction Features in the PA-RISC Architecture*, by R. Lee, M. Mahon and D. Morris, *COMPCON Spring 92 Digest of Technical Papers*, February 1992.

CACHES AND ON-CHIP CACHES

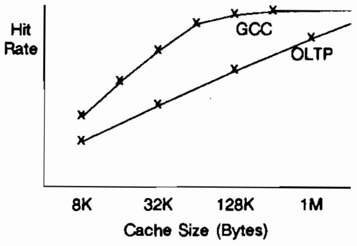
Another technique used to speed up RISC processors is the addition of cache memory. A cache is a small fast memory used by the CPU to hold frequently needed information. The data in the cache is a subset of the data in main memory, much like that pile of important papers on your desk is a subset of the papers in your file cabinet. Figure 7 shows data moving from main memory into the cache, and then into the CPU itself. The cache is important because the CPU can access data in the cache 10 to 50 times faster than data which must be fetched from main memory. The high cost of fast cache memory circuits forces caches to be relatively small, typically from 64 KB in small systems up to 1 MB on expensive systems.

Figure 7: CPU with Cache



The advantage of a larger cache is that the probability of finding needed information is higher. This probability is called the "hit rate". (The "miss rate" is the probability of *not* finding information.) The hit rate is also affected by the type of application, as shown in Figure 8. A small application such as GCC (a compiler) will improve in performance as the cache grows, then level off once the entire data set is in cache, which occurs around 64KB in the figure. Larger applications such as OLTP will continue to improve in performance as the cache size increases, since the data set is much bigger.

Figure 8: Performance vs. Cache



Recent improvements in integrated circuit technology have allowed some small amount of cache memory to be included on the same chip as the processor itself. For example, the R4000 includes 16 KB of on-chip cache, while SuperSPARC has 36 KB. On-chip cache can provide an advantage in the lowest cost products because no additional cache memory chips are required. For most systems, however, processors will use a larger off-chip cache to improve the hit rate. In this case, the on-chip cache can be a cost *disadvantage* by increasing the cost of the CPU chip itself. Furthermore, by splitting the cache between on-chip and off-chip, the on-chip cache tends to impede access to the off-chip cache, as shown in Figure 9. The PA7100, with its single-level cache, can read from its off-chip cache much more quickly than other processors with two levels of cache. These other processors spend much of their time stalled, waiting for the off-chip cache to respond. The ability to access up to 3 MB of cache in just one

Figure 9: Off-Chip Cache Access

	Cycles*	Time
PA7100	1 cycle	10 ns
Alpha	5 cycles	25 ns
R4000	5 cycles	50 ns
SuperSparc	6 cycles	120 ns

*Measured by load-use penalty

cycle gives the PA7100 an advantage in large applications such as OLTP or data bases.

Unlike other processors, America uses special IBM chips for its cache memory. These chips include cache control logic along with the cache memory. Only two configurations are supported, 32 KB and 64 KB of cache. The special design allows IBM to access main memory faster than competitors' systems. But unlike the industry-standard cache chips used by the other CPUs, these parts will have low manufacturing volumes since they are only used by a single product family, and thus are more expensive.

In summary, the use of cache memory significantly improves the performance of RISC processors. Bigger caches provide better performance, particularly for large applications. Using small on-chip caches may reduce cost for low-end designs, but will also reduce performance. For higher performance, the on-chip cache must be coupled with an off-chip cache, eliminating any cost savings. Also, the on-chip cache will get in the way of off-chip cache accesses, resulting in more delay cycles than processors without on-chip cache.

64-BIT PROCESSORS

The smallest amount of data that can be stored in a computer is called a "bit". The original RISC processors all stored data in chunks of 32 bits each. A 32-bit value can store numbers as large as four billion (4,294,967,295). Although initially this seemed to be a big number, over time some programs began to require more space. This need affected processor designs in various ways.

The first area affected was the computational registers. Most RISC processors have two ways of doing math, called integer and floating point. Integer math is suitable for most business calculations (for example, dollars and people can be counted using integers), while floating point math is used mainly for scientific calculations (π is a typical floating point number). Technical applications were the first to need very accurate calculations on very large numbers. To speed up these calculations, all of the latest RISC chips mentioned in this paper provide 64-bit floating point registers.

Many chips still use 32-bit integers. After all, 4,294,967,295 is a lot of dollars or people, and Fortune 100 CEOs can count their sales dollars in millions. Calculations which do require very large numbers can be performed on the floating point side; the PA7100, for example, requires just two cycles to do addition, subtraction or multiplication using the floating point unit. The R4000 and the Alpha chip are the only ones to include 64-bit integers, and these are intended primarily for large memory addressing.

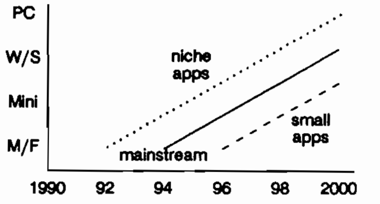
VIRTUAL AND PHYSICAL MEMORY

Memory addressing becomes an issue when the system needs more memory than can be represented by a single 32-bit number. In the case of memory, that means more than four gigabytes (4 GB), which is over 4,000 MB. There are two types of addressing, known as virtual and physical. Virtual addressing determines the maximum amount of *data* that can be used by the system at any given time, and physical addressing determines the maximum amount of *memory* that can be installed in the system.

In the case of physical addressing, it is unlikely that the 32-bit limit will be a problem in the near future. At today's prices, the cost of 4 GB of memory would be well over

$1,000,000, which would be too expensive except for a large mainframe or supercomputer. With the price of memory falling by about 40% per year, 4 GB of memory will be feasible for high-end minicomputers by the middle of the decade, and possibly for high-end workstations by the end of the decade. Certain niche applications will require large addressing in these timeframes, but most mainstream applications will not need more than 32-bit addressing until farther in the future (see Figure 10). HP plans to provide this capability when needed by its customers.

Figure 10: Need for >32-bit Addressing



Virtual addressing is used whenever a user needs to access data, whether it is a file, a set of numbers in a spreadsheet, or the result of a single calculation. For typical applications, a single user may need several megabytes of data, although a large application may require as much as 100 MB. Since a 32-bit processor has 4 GB of virtual address space, it is very difficult for a single user to exceed the available virtual space. On a multi-user system, however, all data requirements of all active users must fit into the virtual address space. Therefore, a system with 50-100 users may exceed the address limits of a 32-bit processor, depending on the needs of each user. If this happens, no new users can access the system.

PA-RISC solves this problem by extending the virtual address space to 64 bits using segments. With this method, each user (or process) is assigned a "segment", or section of memory, of up to 4 GB in size. Since the system has literally billions of these segments available, even very large systems will not run out of segments. And the segments are large enough that very few users will need more than one. If very large spaces are needed, multiple segments can be used, although there is some performance overhead for switching between segments. IBM's POWER architecture uses a similar method with smaller 256 MB segments.

Both DEC's Alpha and the R4000 solve the 32-bit address limit differently, using a "flat" (unsegmented) 64-bit address space. With this method, users can be allocated more than 4 GB of space if needed. Today, very few applications need this much space (see Figure 11). Furthermore, 64-bit flat addressing requires 64-bit integer registers and larger address translation buffers, increasing the chip cost compared to segmented processors. The segmented approach gives users flexibility without the additional cost of flat addressing. HP's philosophy is that customers should not pay for flat addressing until it is really needed (see Figure 10). When the need arises, PA-RISC can be easily extended to allow larger segments in the same way that MIPS extended their architecture.

Figure 11: Sample Applications Needing More Than 4 GB

- Airflow model of an entire plane
- Weather model of entire U.S.
- Database of all U.S. residents with name and address

As a final note, although the PA-RISC, MIPS and Alpha architectures all support a

64-bit address range, it is common for the actual chips to use a "subset" of the full range. For example, the first Alpha CPU only implements 43 bits out of 64. In order to actually use more than 32-bit addressing, the MIPS and DEC operating systems must be modified, as neither company is shipping a 64-bit operating system today. HP is the only major computer vendor currently shipping an operating system (MPE/iX) which handles 64-bit addressing.

PROCESSOR COST FACTORS

In addition to performance, cost is an important factor when comparing RISC systems. The easiest way to compare cost, of course, is to get price quotes for the different systems. In cases where this is not practical, or if you wish to understand processor cost, this section discusses the significant design issues which influence the cost of the processor.

In the processor itself, the two most expensive areas are the large (VLSI) chips, and the cache memory chips. For the VLSI, the number of chips is most important. For example, IBM's America uses three VLSI chips for the CPU itself. By contrast, the SPARCstation IPC uses a single chip CPU. This is one reason why the IPC is priced around $6,000 while the IBM 320H (using America) is priced at $12,000. IBM's Model 220 uses a new single-chip CPU to bring the system price down to around $9,000.*

Since many of this year's new products will be using single-chip CPUs such as the PA7100, SuperSPARC, R4000 or the Alpha chip, the size of the chip becomes important. The SuperSPARC is the largest while the PA7100 is the smallest (see Figure 12). The figure also shows an estimated cost for each chip, compared to the PA7100, based on the chip die area and package cost and assuming that standard manufacturing processes with equivalent defect rates and volumes are used. Because the chip cost varies exponentially with the die size, the SuperSPARC may cost 50% more than the PA7100 to manufacture.

Figure 12: Processor Chip Cost

	# of Chips	Total Area	Cost Ratio*
PA7100	1	196 mm2	1.00
R4000	1	210 mm2	1.00
Alpha	1	233 mm2	1.35
SuperSparc	1	256 mm2	1.50
America	3	484 mm2	1.67

*Estimated from chip area & package cost

Design techniques such as 64 bits, superpipelining or superscalar can increase the processor cost indirectly. Using 64-bit registers and addressing will increase the size of the chip. Since *superpipelining* increases the CPU clock frequency, system vendors can (like DEC) attach a fast, expensive cache to try to keep up with the CPU or (like MIPS) use a slower, less expensive cache and give up some performance. The *superscalar* designer must be cautious to prevent the added complexity from drastically increasing the CPU size; both the multi-chip America design and the very large SuperSPARC chip are highly superscalar designs. The PA7100's simpler design avoids this added expense, as shown by its low cost.

*System prices are approximate list prices for entry-level workstations with color monitors, as of 5/1/91.

Although processor cost is only a single portion of overall system cost, it is often the driving issue, particularly for low-end systems. In these systems, the cost of memory, graphics and peripherals is nearly the same between competitive systems, so it is the processor that makes the difference. Vendors who design low cost processors will have an advantage in the price/performance game.

MEASURING PERFORMANCE

First of all, MIPS is basically a Meaningless Indicator of Processor Speed. MIPS vary widely from vendor to vendor, and can be misleading because they may be measured using different programs, with very different results. "Peak MIPS" is an even worse measure of speed which generally indicates that a processor can run very fast while doing absolutely nothing. MIPS should never be used when comparing processors between vendors, although it may be useful when comparing processors from a single vendor if that vendor uses a consistent definition.

For pure processor performance, the major vendors in the computer industry have agreed on the SPECmark. The original SPECmark suite consists of ten different programs, which are averaged together to create a single SPECmark rating. An important feature is that four of the programs use only integer calculations, while the other six emphasize floating point performance. The average rating of the first four programs is called "integer SPECmark", or SPECint, and the other six are included in "floating point SPECmark", or SPECfp. These two breakdowns are important because SPECint is a more representative of performance of integer-only programs, while SPECfp is representative of highly scientific or graphics-intensive programs. The overall SPECmark is a good measure of general technical workstation performance.

Recently, the SPEC team has refined the SPECmark suite and produced a new set of programs called SPEC92. Sometimes they are called the "new SPECmarks", compared to the original SPECmarks which are now called SPEC89. The major change is the elimination of the controversial "matrix300" benchmark, for which most major vendors have posted very high scores using a technique called the Kuck preprocessor. Since most vendors are now using Kuck, taking out matrix300 will affect everyone’s scores in about the same way. The SPEC92 numbers are generally presented as SPEC92int and SPEC92fp to distinguish between integer and floating point performance.

For multi-user commercial environments, a number of vendors have agreed on the TPC benchmarks. These measure OLTP (on-line transaction processing) performance in different environments. TPC-A represents a typical terminal-based system, and TPC-B is a client-server environment. Although both are measured in TPS (transactions per second) TPC-B numbers are generally much higher than TPC-A numbers. Don’t try to compare one type to the other; ask the vendors for both sets of numbers, or standardize on the TPC benchmark that best matches your environment.

Of course, the final measure of performance is always running your own application. Other benchmarks are simply methods of predicting the performance you will achieve, but they can be very inaccurate because of the large number of factors that affect applications performance. Wherever possible, use your own benchmarks.

PROJECTING APPLICATION PERFORMANCE

The performance of computer systems varies from application to application. For the latest generation of RISC processors, the variation can be even greater due to the effects of superscalar, superpipelining and on-chip cache designs. As an example, Figure 13 shows the wide variation in the ten SPECmark benchmarks for an IBM system. It is important to pick the right benchmarks to characterize your application, or as mentioned previously, to test your application on the target system.

Figure 13: Performance Variation
(IBM Model 340 SPECmarks)

gcc	26.0
espresso	27.3
li	28.4
spice	34.0
eqntott	34.2
doduc	38.8
fpppp	69.4
tomcatv	98.7
nasa7	103.0
matrix300	524.1

When considering which benchmark best matches your application, an important consideration is integer versus floating point usage. In general, most commercial applications such as OLTP, data bases, and spreadsheets use very little floating point arithmetic. For these applications, a benchmark such as SPECint may be appropriate. On the other hand, applications performing scientific or technical calculations are best represented by the SPECfp number. Graphics and multimedia also emphasize floating point performance. The overall SPECmark number is a good representation for the typical technical workstation user.

The size of the application is also an important concern. Because processors such as Alpha or the R4000 run at full speed only when the application fits into the small on-chip cache, performance can drop significantly if that limit is exceeded. Performance drops again if the external cache is full and the remainder of the data must be kept in main memory. Processors which use a single-level cache, such as the PA7100, have a steady rate of performance until the size of the external cache is exceeded. In any case, knowledge of the amount of cache needed by an application is useful when comparing systems; this information can be obtained by trying the application on a few systems with different cache sizes and carefully comparing the results.

For extremely large applications, virtual memory size can become an issue, although this is very rare today. IBM POWER systems have a maximum segment size of 256 MB, so single applications requiring more than this amount of data may see a performance degradation. PA-RISC systems have segments up to 4 GB, so even fewer programs will run into this limit. Alpha and the R4000 provide 64-bit flat addressing which has practically no virtual address limitations.

Superpipelined CPUs have their own quirks. These processors have larger penalties than other processors for loading data and for branches. Because of this, superpipelined processors do best when performing many calculations on a small set of numbers; applications such as database searches which require a large number of loads will have lower performance. Similarly, technical calculations often have fewer branches than commercial applications, which typically branch every seven or so instructions. Superpipelined CPUs, such as Alpha and the R4000, generally do better on technical programs than commercial applications.

As for superscalar designs, the key issue here is that recompiling may be necessary to wring the maximum performance out of these CPUs. The more changes that are made from the previous generation, the bigger the performance difference will be between old binaries and recompiled code. For example, the three-way superscalar SuperSPARC will have a large slowdown when running old binaries, while the two-way PA7100 will have only a small penalty and only for floating point applications. The scalar R4000 may see little difference between old binaries and recompiled code. A related issue is that sharing binaries on a network between different superscalar systems may be difficult, because code optimized for one superscalar design may not be best for another superscalar design with different programming rules.

Multiprocessing has recently become very widespread for multi-user applications. In such applications, it is relatively easy to spread the work load across several processors, since each processor can handle one or more users. If hardware and software are well designed, up to 80% to 90% of the maximum theoretical performance can be achieved on real applications. However, multiprocessing is not well-suited to single-user systems because current compilers and operating systems cannot take a single task (such as a compile or a batch job) and share it between more than one processor. Desktop systems will be limited to one or two processors until more advanced software is available.

COMPARING THE RISC CHIPS

This section summarizes the key features of five of the latest RISC processors. Comparisons must be made carefully because some of these chips are currently available while others have only recently been announced. Additional technical information on these chips is contained in the Appendix.

Hewlett-Packard's PA7100 chip is a single-chip processor packing a PA-RISC CPU, floating point unit, memory management unit (TLB), cache control and bus interface onto one integrated circuit. This is accomplished by relying solely on off-chip cache, eschewing the complexities of superpipelining and u·ing a simple two-way superscalar design. This design philosophy results in a chip that is smaller, and thus less expensive, than any of the other four. Yet at 130 SPECmarks, performance is comparable to the more expensive Alpha chip and well ahead of the rest.

The PA7100 can execute one integer and one floating point instruction on each cycle (two-way superscalar). Its performance is enhanced by the combination instructions of PA-RISC. This ability to execute two operations using a single instruction reduces the need to have a more complex superscalar design. For example, using these instructions, the PA7100 could perform an integer operation, a branch, a floating point add and a floating point multiply in a single cycle. The chip is rated at 1.3 SPECmarks per MHz. 64-bit virtual addressing is implemented using 4 GB segments.

The low cost and high performance of the PA7100 make it suitable for a range of products from low-end desktop systems to mainframe-class multiprocessor systems. It will be shipping in such products before the end of the year at clock frequencies up to 100 MHz.

MIPS' R4000 is also a single-chip microprocessor, including all of the functions of the PA7100 plus a small 16 KB on-chip cache. Additional cache can be added externally.

The R4000 uses superpipelining to reach speeds of 100 MHz on-chip, although the off-chip circuits run at 50 MHz. The chip is a scalar design capable of executing one instruction per (100 MHz) clock cycle. It is rated at 70 SPECmarks and the first system, Silicon Graphics' Crimson workstation, began shipping in March.

The R4000 is a full 64-bit implementation including 64-bit flat addressing. It is designed for low cost and is less expensive than any of the other processors except for the PA7100. Due to the superpipelining and two-level cache, efficiency is low at 0.7 SPECmarks per MHz.

Floating point performance is particularly weak, making the chip poorly suited for technical or graphics-intensive applications. Integer performance is better but does not reach the levels of the Alpha chip or PA7100. Because of these limitations, the R4000 is best suited for the desktop business systems specified by ACE.

DEC's Alpha chip is another single-chip CPU very similar to the R4000. The key differences are that DEC's chip runs twice as fast, up to 200 MHz, and that in addition to being superpipelined, the Alpha chip is also two-way superscalar. This combination of features provides performance of around 150 SPECmarks, more than any of the other chips listed here, but only 0.75 SPECmarks per MHz.

Like the R4000, Alpha also uses 64-bit flat addressing. Like the PA7100, both integer and floating point performance are strong, making Alpha suitable for both DEC's technical (ULTRIX) and commercial (VMS) markets. Like the R4000, however, the superpipelined design and small on-chip cache may hinder performance on large applications.

One drawback is that the chip is larger (thus more expensive) and gives off more heat than the R4000 or the PA7100, meaning that it may not be suitable for systems under $10,000. It is not expected to be shipping in products until the end of 1992. Because Alpha is a new architecture, all code must be recompiled to be ported to Alpha.

IBM's America processor is a three-chip design, excluding the special cache chips. This design stresses high efficiency, reaching 2.0 SPECmarks per MHz using a four-way superscalar design. The complex design requires the use of multiple chips to contain the added circuitry, which not only increases the cost of the processor but prevents operation at high frequencies. The fastest America chips reach clock speeds of just 50 MHz, producing up to 100 SPECmarks.

The America chip set is not superpipelined. It provides 52-bit virtual addressing using 256 MB segments. Because of the cost of the multiple chips, peak performance is available only in high-end servers, and America is not sold at all in systems under $10,000. IBM has designed the less powerful "RSC" chip for low-end systems.

America, the first POWER processor, has been shipping since 1990, although the most recent high-end chip set was just announced for mid-1992 shipments. Even with the new design, integer performance is less than any of the other four, making America relatively poor for integer-intensive applications. Strong floating point performance makes this CPU best suited to technical applications.

SuperSPARC is a joint effort of Sun and Texas Instruments to extend the performance of the SPARC architecture. It is a large single-chip design which incorporates all of the functions of the PA7100 plus 36 KB of on-chip cache, more than twice that of the R4000 or Alpha chips. It is able to execute up to three instructions per cycle, including any two integer instructions together. Like America, SuperSPARC is limited to a 50 MHz clock speed by the complexity of its design, and overall performance will be much less than Alpha or the PA7100.

SuperSPARC is larger than any of the other single-chip designs and is manufacturing in a BiCMOS process which is more expensive than the standard CMOS processes used by the other chips. Because of this cost, SuperSPARC will be used initially in high-end (above $10,000) systems only, although the cost of the chip will decline over time as IC manufacturing processes improve.

In the interim, its built-in bus interface (MBus) will allow SuperSPARC to be built into relatively inexpensive two- and four-processor systems that will challenge the performance of faster uniprocessors in multitasking applications, although these multiprocessor SPARC systems will not be as well suited to single-user desktop systems. Like IBM, Sun will use a different chip for its low-cost systems during the next year or two.

Figure 15: Summary Comparison of Five RISC Processors

	Super-scalar	Super-pipelined	64-bit Addressing	Cost Ratio†	SPECmark Rating	System Availability
HP PA7100	2-way	No	Yes	1.00	130*	1992*
DEC Alpha	2-way	Yes	Yes	1.35	150*	1992*
IBM America	4-way	No	52-bit	1.67	100	mid-92
MIPS R4000	No	Yes	Yes	1.00	70	3/92
SuperSPARC	3-way	No	No	1.50	70*	3Q92*

*Estimate based on industry sources; no system announced as of 5/1/92.

†Estimate calculated from chip die size, package cost and wafer cost.

CONCLUSIONS

RISC processors use a variety of techniques to improve performance. Most use simple pipelining to overlap instruction execution and increase clock speed. Some use superpipelining to increase the pipeline length, which can offer even higher clock rates. This increased speed, however, is not always available to the user because significant penalties for loading data and branching reduce the efficiency of a superpipelined CPU. Therefore, superpipelined processors must achieve nearly twice the clock speed of pipelined processors to provide the same performance.

Superscalar design allows a processor to group two or more instructions together and execute them at the same time. While in theory this can double or triple the peak performance, in practice superscalar designs today provide 20% to 80% improvement because of limitations in the hardware design. Even to achieve this improvement, code must be recompiled to take best advantage of the superscalar design.

The use of cache memory significantly improves the performance of RISC processors. Bigger caches provide better performance, particularly for large applications. Using small on-chip caches may reduce cost for low-end designs, but will also reduce performance. For higher performance, the on-chip cache must be coupled with an off-chip cache, eliminating the cost savings. In addition, these two-level cache systems have more delay cycles than processors without on-chip cache.

Extending the virtual address space to 64 bits impacts only those applications which require more than 4 GB of total address space. PA-RISC and POWER use a segmented method to provide extended addressing without the cost of full 64-bit registers. The R4000 and Alpha provide flat 64-bit addressing which only provides a performance advantage for those few applications larger than 4 GB.

All of these design techniques require tradeoffs between cost and performance, and also tradeoff performance on some applications versus performance on other applications. No single processor is best for everything. Because of the variability in performance across different applications, it is important to choose the right benchmarks which best represent the performance of your application. The best case is to measure the price/performance of competing systems running your application. In the final analysis, this is far more important than the "super" hype.

NOTE: The information contained here has been verified where possible with the manufacturer's published literature as of the date of writing, but Hewlett-Packard assumes no responsibility for any errors.

APPENDIX: TECHNICAL INFORMATION

Table A-1 compares the instruction set architecture of the five RISC CPUs. The five are very similar in instruction size, number of registers and address space, although the R4000 and Alpha have added 64-bit integer registers and address segments. PA-RISC floating point registers can be configured as 32 64-bit registers or 64 32-bit registers. Note that Alpha and MIPS are more limited in the number of addressing modes.

The last section notes some miscellaneous features of the various architectures. PA-RISC and POWER implement a large number of of compound instructions that are available to a limited degree, or not at all, in other architectures. PA-RISC includes some simple support for decimal arithmetic, which is useful for COBOL programs. Three of the five use a branch delay slot to reduce the branch penalty. "Endianness" refers to whether bytes are numbered from highest to lowest or vice versa.

TABLE A-1: ARCHITECTURE INFORMATION

	HP PA7100	IBM America	DEC EV4	MIPS R4000	Super- SPARC
Architecture Version	PA-RISC 1.1	POWER	Alpha	MIPS-III	SPARC v8
Instruction Width	32 bits	32 bits	32 bits	32 bits	32 bits
General Registers (number)	32	32	32	32	32
General Registers (width)	32 bits	32 bits	64 bits	64 bits	32 bits
FP Registers (number)	64/32	32	32	32	32
FP Registers (width)	32/64 bits	64 bits	64 bits	64 bits	32 bits
Virtual Address Space	64 bits	52 bits	64 bits	64 bits	40 bits
Maximum Segment Size	32 bits	28 bits	62 bits	62 bits	32 bits
Addressing Modes:					
Register+Offset (disp)	Yes	Yes	Yes	Yes	Yes
Register+Register (indexed)	Yes	Yes	No	No	Yes
Register+Register (scaled)	Yes	No	No	No	No
Byte/Halfword Addressing	Yes	Yes	No	Yes	Yes
Compound Instructions?	Yes	Yes	Few	Few	No
Decimal Arithmetic?	Yes	No	No	No	No
Branch Delay Slot?	Yes	No	No	Yes	Yes
Endianness?	Big	Big	Little	Both	Big

Table A-2 gives specifications for the instruction execution of the CPUs. Instructions per clock indicates the degree to which the design is superscalar. The next section shows the penalties (in cycles) for some common pipeline hazards. The load-use penalty determines the number of cycles between when data is loaded from cache (either on-chip or off-chip) and when it can be used in a computation. The branch penalty is the number of cycles wasted when a branch occurs and the processor has made an incorrect prediction for the outcome of the branch. The branch penalty is reduced by the number of delay slots (if any). The latencies (in cycles) for common math operations (in double precision) are then listed.

TABLE A-2: PIPELINE INFORMATION

	HP PA7100	IBM America	DEC EV4	MIPS R4000	Super- SPARC
Pipeline Frequency	100 MHz	50 MHz	200 MHz	100 MHz	50 MHz
Number of Stages	5 stage	5 stage	7 stage	8 stage	4 stage
Instructions / Clock	2 max	4 max	2 max	1 max	3 max
Penalties:					
Load-Use (on-chip hit)	n/a	n/a	2 cycles	2 cycles	0 cycles
Load-Use (on-chip miss)	1 cycle	1 cycle*	5 cycles‡	6 cycles‡	5 cycles‡
Badly Predicted Branch	1 cycle	0-3 cyc†	3 cycles	2 cycles	0 cycles§
Latencies:					
Floating Point Add	2 cycles	2 cycles	6 cycles	4 cycles	3 cycles
Floating Point Multiply	2 cycles	2 cycles	6 cycles	8 cycles	3 cycles

n/a = not applicable

* estimated value, not confirmed by vendor

† penalty is 3-i cycles where i is the number of instructions between the condition code setting instruction and the branch.

‡ may be longer if external cache is running at less than maximum frequency (see Table A-3).

§ may reduce number of instructions executable in that cycle

Table A-3 describes the cache implementations of the five chips, showing the cache size, frequency and bandwidth (data transfer rate) for both on-chip (if any) and off-chip cache. For America, "on-chip" is considered to be the first-level instruction cache while "off-chip" is the second-level unified cache. America has two designs which support either 8 KB or 32 KB of instruction cache. The type of SRAM chips used to implement the off-chip cache is also shown.

TABLE A-3: CACHE INFORMATION					
	HP PA7100	IBM America	DEC EV4	MIPS R4000 ·	Super-SPARC
On-Chip Cache					
I-Cache Size	none	8/32 KB	8 KB	8 KB	20 KB
D-Cache Size	none	none	8 KB	8 KB	16 KB
Max Frequency	n/a	50 MHz	200 MHz	100 MHz	100 MHz
Peak Bandwidth	n/a	800 MB/s	3200 MB/s	1200 MB/s	1600 MB/s
Off-Chip Cache					
I-Cache Size (Min)	4 KB	32 KB	128 KB	128 KB	???
D-Cache Size (Min)	4 KB	unified	total	total	unified
I-Cache Size (Max)	1 MB	64 KB	8 MB	4 MB	2 MB
D-Cache Size (Max)	2 MB	unified	total	total	unified
Max Frequency	100 MHz	50 MHz	66 MHz	50 MHz	50 MHz
Peak Bandwidth	1600 MB/s	400 MB/s	1050 MB/s	800 MB/s	400 MB/s
SRAM Required	9 ns standard	IBM custom	8 ns standard	17 ns standard	17 ns* pipelined

* = estimated value, not confirmed by vendor

Table A-4 gives the general parameters of the memory management unit, or TLB. TLB entries are divided into instruction entries and data entries for chips which support this distinction. The next section shows the address space as actually implemented on the chip (compare to the architectural definition in Table A-1).

TABLE A-4: MEMORY MANAGEMENT INFORMATION					
	HP PA7100	IBM America	DEC EV4	MIPS R4000	Super-SPARC
Page Size	4 KB	4 KB	8 KB	4 KB	4 KB
TLB Entries (I/D)	136	32/128	16/32	48	64
Ways Associative	fully	2 way	fully	fully	fully
Virtual Address Space	48 bit	52 bit	43 bit	42 bit	40 bit
Maximum Segment Size	32 bit	28 bit	41 bit	40 bit	32 bit
Physical Address Space	32 bit	32 bit	34 bit	36 bit	36 bit

Table A-5 gives information about the physical parameters of the die and its package. The "feature size" is often called the gate size. The "channel length" is LEFF for the p-channel. The chip cost is estimated using a mathematical model based on the number of chips, die size, and difficulty of the IC process used; these costs are then expressed as a ratio of the estimated PA7100 cost.

Using Agent Concepts to Migrate to New Technologies

Edward Hansen

Hewlett-Packard Company
3000 Hanover St.
Palo Alto, CA 94304
(415) 857-8925

In June of 1990, my manager asked me to meet with him regarding a new assignment that he had in mind for me. My manager is the section manager in charge of the central order management systems for Hewlett-Packard. At that time, the development effort for the first phase of a comprehensive HP3000-based order management system designed for world-wide implementation (WWOMS) was nearing completion. Despite the high caliber of team leaders and technical people which staffed the development teams, it appeared that those problems which seem to plague most such large scale implementations were also afflicting this project.

During our discussions of the development history of this project, what my manager and I concluded was that perhaps the technologies which were being used at that time may no longer be effective in the implementation of such large scale projects in a timely manner. In order to provide the reader with some background information, a brief history of the order management application follows.

The system then in development had not been designed and built from the ground up. Prior to this world-wide version, there existed three similar HP3000-based order management systems, each focusing on one of the three major HP field sales operations existing at that time: the U.S., Europe and Inter-continental. After some investigation of those three systems, the design team concluded that the European version already had much of the functionality required for a world-wide system. The thinking then was that it could be used as the base system to which a series of enhancements could be applied. A developmental strategy was thus formulated around that reasoning. With such a strategy, a system with the desired world-wide feature set would be implemented in considerably less time than it would take to design and build a completely new one.

The logic in that strategy seemed incontestable. Yet it was taking as long to deliver just the first phase functionality (i.e., the new U.S. feature set on top of the core order management functionality) as the design team had estimated to deliver the entire world-wide feature set. What had originally been estimated as an eighteen-month duration project was now turning into a multi-year effort. Clearly, the work estimates were proving to be far too optimistic to be realized despite the fact that everyone on the development teams was working as hard as they could.

What went wrong? Well, at least in this case the premise that it would be easier to modify the existing code than to design and build new code proved to be incorrect. Upon reflection, it became apparent that some of the existing code was already almost a decade old and that code had passed through many programmers' hands and styles. Although the programs consisted of many called COBOL modules, most of these modules were being passed most, if not all, global data structures declared in the main COBOL programs which were calling them.

Upon analysis of the existing program logic structures, it may even be inferred that the modularity which did exist resulted from exceeding code segment size limits on the classic HP3000 rather than by reasoned and deliberate design. That is to say, as a program grew in size, the code was simply broken into "chunks" which all had access to the global data structures just as if they were still physically part of the main program. Logic which was managing the VPLUS screens was intermixed with database access logic together with logic which enforced policies and business rules. Indeed, the programs were really monolithic constructs masquerading as modular structures.

My manager and I concluded that it would not be cost effective to continue to invest more resoures in that type of system architecture beyond what was required to fulfill this application's original proposed level of functionality. We felt that adopting some of the newer technologies would bring greater productivity to our development environment. However, there was no "in-house" expertise available in the newer technologies in our department and we would have to determine how to capitalize on these newer technologies in our own particular situation.

I was then chartered to lead an investigation into these new technologies with the objective of discovering how they could be effectively utilized in our future system development activities. A survey of the technologies then being used in the implementation of the order management system and alternative newer technologies which were already available yielded the information in the following table.

Technology	Being Used	Available
Language	3GL (COBOL)	4GL, OOPL
DBMS	networked (IMAGE) indexed (KSAM)	relational with advanced features
User Interface	block mode terminal screen (VPLUS), single screen at a time	GUI, multiple screens at a time (e.g., X-Windows)

All well and good! However, our department charges all its expenses back to our internal customer base. It would be unlikely that a proposal to embark on yet another major development effort even before we had completed the one in progress would be well received by our customers' management. Considering that the deliverables were also behind schedule made the presentation of such a proposal a virtual impossibility.

Information collected in the initial investigation supported the contention that there was indeed a productivity gain when these newer technologies were properly utilized. It was imperative that we create an opportunity whereby we could construct a functioning prototype which incorporated these newer technologies. In addition, we had to be able to build and demonstrate this prototype in a very short period of time with limited resources. Failure to achieve this goal could conceivably restrict us to the less than desirable development paradigm which was then in place.

Fortunately, there was a funding source internal to Hewlett-Packard but external to our department which could provide some amount of "seed money" to get such high risk prototype developments started. The challenge then was to conceive and propose an architecture with a working prototype which fulfilled the following requirements.

- The prototype would provide sufficient order entry functionality to demonstrate that it is "real" and not simply a "model."

- The prototype could be able to run concurrently with the production version of the World-Wide Order Management System.

- The prospective customer must be able to perceive value added by the user interface of the prototype beyond that provided by the production version of the World-Wide Order Management System.

- The potential for a gain in system development productivity should be shown by demonstrating the working prototype within six months from the start of actual system development including training time in the usage of the new technologies.

We knew that these were very aggressive parameters but the remaining seed money would provide funding for only six months of contract programming resources. These objectives would have to be met if we expected to attract any internal investors to fund further development.

During the initial investigation, the set of products and tools which INGRES offered as a system development environment was especially attractive. Many of the new technologies were presented at high levels which made learning and usage easy. This ease of learning and usage was an important aspect to consider because such time was included in our six month development "time box."

In addition to a Relational Data Base Mangement System (RDBMS) which has a rich set of advanced features, three other leading edge technologies are incorporated in another INGRES product, Windows/4GL. This product not only provides a highly functional Graphics User Interface (GUI) development tool, it also offers a 4GL programming language which is both object oriented and event driven.

When these program development facilities are coupled with the capability to utilize the RDBMS to manage private data structures with SQL as if they were part of the program's working data space, that combination becomes an extremely powerful tool. Because of this, we were fairly confident that if we simply had to develop a prototype which was based only on the INGRES products, we could satisfy the time box requirement. We also knew from documented studies and our own observations of how people react to graphical user interfaces that the potential customer would indeed perceive value added in the GUI versus the VPLUS screens.

Well, two out of four wasn't bad! But, neither was that good enough. Satisfying the other two requirements was even more important because those were fundamental to a strategy which permitted an orderly migration to the newer technologies. The prototype had to be "real" in the sense that a user could actually enter and maintain orders which were stored in the WWOMS database. An architecture must be created which allowed the prototype to co-exist with the concurrently active WWOMS so that they could access the same order data repository.

We wanted to use the INGRES Windows/4GL and RDBMS products for the reasons described above but required access to TurboIMAGE databases and KSAM files resident on the HP3000. Since the INGRES products that we needed were supported only on HP-UX and not on MPE XL, it was clear from the beginning that we would have to base the architecture on some kind of a Client-Server model. That architecture would also have to somehow reconcile seemingly incompatible technical requirements.

First or all, we knew from previous experimentation in LAN environments that many transport requests for relatively small amounts of data was a performance killer for On-Line Transaction Processing (OLTP) applications based on a Client-Server model. On the other hand, a single request to transport a relatively large amount of data did not appear to take significantly longer than a single request for a small amount of data. With this knowledge, it seemed reasonable to somehow incorporate the transportion of larger but fewer chunks of data into the architecture for performance reasons.

WWOMS already treated the entire order as an entity. In other words, when a user accessed an order with update intent, the entire order was made unavailable to anyone else even if it was for viewing purposes only. A new order was added to the database only when a user indicated that the order was to be posted. Also, even though WWOMS never has update intent on product and customer resource data, it does view that data on an entity basis as well.

Fortunately, adopting this methodology and transporting the entire set of data associated with an order or a product or a customer in a single transmission was consistent with the performance characteristics of the LAN. Unfortunately, the INGRES Windows/4GL product could not pass a large array of data as a parameter to a called 3GL procedure because of its object oriented attributes. Arrays of data are declared as data objects and accessed with methods available only to Windows/4GL components. Only simple variables like integers, reals and null-delimited character strings can be passed as parameters to called 3GL procedures.

In order to exchange large amounts of data between a Windows/4GL Client running on HP-UX and a Server running on MPE XL, we incorporated Agents into the architecture to act as intermediaries between the Client and the Server. Webster's defines the term "agent" as "one who acts for or in place of another by authority from him ..." Such is a suitable definition for the function performed by our Agents. In our architecture, we assign an Agent to represent the Client and another to represent the Server. Therefore, we name them the Client's Agent and the Server's Agent. We also call the Client or the Server the Patron of its respective Agent.

Unlike other architectures based on a Client-Server model, our Clients and Servers do not communicate or exchange data directly with each other. All communications and data exchanges are performed by the Agents on behalf of their

Patrons. In this way, Agents can effectively insulate their Patrons from changes in message formats and protocols. They can be enhanced to adapt to necessary changes while performing the necessary translations in order to communicate properly with each other and their Patrons.

Additionally, the Agents play an important role in the indirect exchange of large amounts of data between the Client and the Server. We created a memory table model as a component of the architecture whereby tabular data structures are used as the primary data exchange media between a Patron and its Agent. Only short messages representing Patron requests or return status and data events are directly exchanged by the Patron and its Agent.

Since INGRES database tables are easily accessible by both Windows/4GL Clients and 3GL Agents utilizing embedded SQL, they are used as the memory table media on the HP-UX platform. Short messages are communicated between the Client and its Agent using Berkeley Sockets.

On the MPE XL platform, mapped files structured as tables serve as the primary data exchange media between the Server and its Agent. A combination of a mapped file and a RIN is utilized to enable short message communication between the Server and its Agent.

The last year has been an extremely exciting time. In late January, 1991, we had finally recruited the contract programming resources who had the appropriate starting skill set. It was important that any required training be confined to the INGRES products only. By late July, 1991, we were demonstrating a prototype which satisfied the four critical requirements outlined previously. Based on that prototype's functionality, we were also successful in attracting additional funding from internal investors to continue experimentation with the development of "custom" Clients.

The reader may ask where the project is today. In late January, 1992, the steering committee which oversees the order management systems' activities was sufficiently convinced of the merit and potential of the new architecture. We gained the approval to proceed with the development of a full function WWOMS Server to be used to service a number of "custom" Clients which will be developed by our customers' Information Technology departments.

The implementation of this Server and its associated Clients will be the first step in the migration process which will eventually allow us to finally leave the older technologies behind. By the end of this year, we should be able to start formalizing the next few steps in the migration strategy. That, however, will be yet another story to tell. Stay tuned!

Postscript

As of this writing in late April, 1992, we are in the midst of refining the design of some of the components of the new architecture based on our experience with the prototype implementation. An update on our progress will be given when this paper is presented in early August.

An Introduction to HP-UX Security Issues & Controls

Paper #2030

Chris Hauck

Hewlett-Packard Company
4501 Erskine Road
Cincinnati, OH 45242
(513) 891-9870

Abstract

As HP-UX systems are moving into the mainstream commercial data processing environment at an accelerating rate, more and more mission-critical applications are running on HP9000 systems. These applications and related databases of information have become one of the most valuable and non-replaceable assets for many corporations. As with all other company assets, protecting the integrity and privacy of this information is, or should be, of paramount concern.

Very often security is an after-thought when installing or maintaining a system. The key to successfully maintaining a secure system is to begin with an understanding of the key issues facing the security/system manager and the available means of securing the system. Only then can an adequate, effective security plan be developed and implemented, otherwise it's like trying to hit a target in the dark.

This paper begins with a discussion of general commercial security requirements, outlining the concerns and issues facing most security and system administrators today. These issues are then discussed as they relate specifically to the HP-UX environment, with a presentation of available security controls, monitoring tools and potential areas of vulnerability. With a number of shops now operating in a mixed MPE and HP-UX environment, this paper also relates HP-UX security controls and concepts to the MPE/iX operating environment.

Introduction

System security planning is very similar to an insurance policy; until something is lost, stolen or damaged, it is just a costly expense and an inconvenience for an organization. Often overlooked during system implementation and planning, system security is typically implemented haphazardly at best. Initial efforts at securing a system become ineffective after on-going security maintenance and management activities are forgotten. Controls and security measures that have been implemented in the past have not kept pace or been updated to accommodate newer applications, network connections and client-server implementations. This often leaves gapping holes in the security of the systems which could cost the organization a considerable amount of money due to loss of data (through either malicious or accidental acts), loss of business (through temporary loss of computing facilities) and causing a general loss of credibility in the industry.

Security is a risk. And just like all risks encountered in doing business, the risks can be managed to avoid these potential security hazards. Implementation of procedure modifications, operational changes and additional controls can assist in reducing these risks (as in managing any risk, the cost of avoidance must be weighed against the potential consequences). The greater the value of the information being processed, the greater the need to implement controls to restrict access and monitor activity. After balancing the costs and risks, it is important to identify any remaining vulnerabilities not taken into account. This involves a commitment by all levels of an organization, with a security plan detailing goals and policies set forth by the upper level management. The purpose of this security plan includes:

- Identifying the goals and objectives of implementing security policies and procedures
- Detail the current operating environment (hardware, software, data, networking, etc) and value to the organization, identifying need for privacy & integrity
- Review existing security controls and policies
- Identify general security requirements, guidelines and policies
- Identify areas of vulnerability and necessary controls to minimize the risk
- Establish an implementation schedule for additional controls and on-going audit procedures; assign responsibilities for information security along with accountability policies
- Identify necessary training programs for various groups within the organization

The Unix®[1] operating system was developed at Bell Laboratories by programmers, for programmers. Security of the system and files was not a major concern since all access was either by an individual or by a local development team. Unix has traditionally been utilized in this type of "open" atmosphere, with almost every user having access to any file or resource of the system. In addition, there has been quite a bit of information published about the internal workings of the operating system, networking capabilities and potential security holes in the system. This widespread knowledge of the bowels of Unix, has contributed to its reputation as an insecure operating environment. Outside of Bell Labs, Unix was popular among many universities and gradually during the past decade, Unix-based systems have permeated mainstream commercial data processing environments. During that period of many years, countless new features and utilities have been added to the core operating system. Included in these developments are features designed to enhance the security of the system and files; however, very often these facilities go unimplemented.

When the word "security" is usually mentioned the term that frequently comes to mind is "hacker." The media has publicized a number of both fictional and real-life computer security breeches over the past few years, popularizing the hacker role in the movie "War Games." Although these types of concentrated attacks to break security of a system certainly exist (as evidenced by the 1988 Internet "worm", the 1986-87 international Internet computer break-ins and the various viruses and trojan horses), evidence suggests that the greatest potential threats lie within an organization: through current employees or ex-employees attempting to access unauthorized information, lack of effective data backup procedures or unintentional modification or destruction of data due to carelessness or lack of understanding.

In order to begin to understand the potential vulnerabilities of your system, it is worth considering the types of security "attacks" that may be waged. In a broad sense they can be categorized as either:

(1) Attempting to access (logon) an unauthorized system or user id, or
(2) Attempting to access (read/modify/execute) an unauthorized file, database or executable program.

For each of these areas, a few of the potential vulnerabilities and controls will be reviewed.

[1] UNIX is a registered trademark of AT&T in the U.S.A and other countries.

Controlling System Access

Keeping unwanted users from knowing about your system and logging on is more than half the battle in securing the system. Once signed on, the possibilities for causing some type of damage greatly increase; therefore, it is at this point that a good portion of the effort involved in maintaining a secure system lies. With the development of sophisticated networking protocols and services, and the proliferation of systems attached to corporate, metropolitan and global networks (such as the Internet), identifying a list of accessible systems has become a trivial task, even for the Unix novice. Published lists of systems, including logons for non-privileged file transfer (FTP) access is widely available on nearly every bulletin board and even the Internet system themselves. Even if a system is not directly connected to the public networking environments, most multi-user business systems have some type of modem access available, and all have directly attached terminal or X-window devices. This minimizes the challenge for a potential intruder, and places the full burden of identifying permissible users of the system squarely on the shoulders of the security administrator(s) and/or system manager(s).

Prevention of a problem by restricting access is very often many times less costly than attempting to deal with the problem after the fact. Controlling access to the system's hardware, networking capabilities and user ids is the first line of defense. Devices such as call-back modems and address filtering bridges and routers can assist in controlling who may access a system; however, this paper will focus on the Unix operating system level controls and security functionality available to the security administrator. Very often, implementing a basic set of security provisions and policies along with continual monitoring for compliance, will provide the greatest incremental improvement in overall system security and integrity.

With regard to controlling system access, these basic security provisions relate to:

- Password protection
- Default Accounts (user ids)
- Obsolete Accounts
- Login scripts

Password protection

In order to determine whether or not a person should be allowed access to a system or not, some method of identifying that person is required. Sophisticated techniques (such as retnal scans, voice prints and finger prints) have been implemented to verify a person's identify, but also have some limitations due to costs, complexity and today's distributed environments.

Passwords remain a cost effective, generally available, and if properly administered, fairly reliable means of identifying a user based upon that person's knowledge.

With Unix (and HP-UX, Hewlett-Packard's implementation of Unix), user account information, including user name, user id, group id, password, comments, home directory and startup program are maintained by default in the file '/etc/passwd', formatted as follows:

```
$ more /etc/passwd
root:7v48qzZbiEhok:0:3::/:/bin/ksh
daemon:*:1:5::/:/bin/sh
bin:*:2:2::/bin:/bin/sh
adm:*:4:4::/usr/adm:/bin/sh
uucp:*:5:3::/usr/spool/uucppublic:/usr/lib/uucp/uucico
lp:*:9:7::/usr/spool/lp:/bin/sh
hpdb:*:27:1:ALLBASE:/:/bin/sh
tftp:*:77:10:TFTP daemon:/usr/tftpdir:/bin/false
ppluser:*:80:10:ppl user:/usr/tftpdir:/usr/bin/ppl
bsmith:rqSCr405ystZY:502:21:Bill Smith,555-1234,555-
      5678:/users/bsmith:/bin/sh
```

General format:

```
<username> <encrypted password> <user id> <group id> <comments> <HOME
      directory> <startup program>
```

User id and Group id are numbers associated with a user at login time, based upon the supplied user name. The password is stored in the file using a one-way encryption algorithm. During the login process, after the user specifies a user name and password, the system performs a one-way encryption on the supplied password and verifies both the user name and encrypted password against the /etc/passwd file entries. If they match, the login process continues by executing any login 'scripts' (a sequence of operating system command interpreter or 'shell' commands stored in a file) defined by the system administrator and/or user. The user and group id's are used by the Unix file system in establishing file level access controls.

A special user id of zero (0) identifies the user as a "super-user" to the system. This is analogous to System Manager or SM capability in the MPE/iX operating system of the HP3000 computer family. Unlike MPE which allows specification of a "capability list" to users/groups/accounts, Unix only provides two type of accounts: root (or super-user) and everything else. Super-users can access (read/write/execute/delete) any file in the file system. In an operating state known as a "trusted system", the super-user is also capable of maintaining the security and integrity of the system. The HP-UX operating system limits execution of many system management commands to super-users (such as adding new users, removing users, establishing login startup scripts, kernel configuration/modification, etc).

A person attempting to gain illegitimate access to a Unix-based computer system will likely attempt to gain super-user privileges. Once gained, the damage or access to confidential data is nearly limitless. The user name 'root' is similar to 'MANAGER.SYS' on the 3000, in that it is the user id with the most capability, and exists on every Unix-based system. Consequently, access to the root password should be restricted to the minimum set of individuals, and changed frequently.

The 'root' user name is generic and does not uniquely identify an individual, especially if the logon password is known by more than one person. Therefore, the better method for gaining root access to the system is to have those individuals login to the system using their unique non-root privilege user names, and using a Unix facility called '<u>su</u>', or switch user, to temporarily gain root privileges when necessary. <u>su</u> allows a logged in user to effectively switch to a new user id and group id without logging off and on again. If the current user id is non-zero (not root privileges), the person must specify the password of the requested user name. The advantage of using this technique versus directly logging on as 'root' is that the <u>su</u> command can optionally log into the file '/usr/adm/sulog' both successful and unsuccessful attempts to switch user, if the file exists within the system. This provides an audit trail for users gaining access to root or any another user id via the <u>su</u> command. The file /usr/adm/sulog contains the date/time, success or failure (+/-), terminal device and original-destination user names, as shown below:

```
$ whoami
jshoe                                     (our current user name)
$ su root                                 (now switch to user 'root')
Password:
# more /usr/adm/sulog                     (examine the su log file)
SU 04/20 09:41 +    ttyu0 msmith-root
SU 04/22 16:18 -    ttyp3 bmiller-root
SU 04/22 16:37 +    ttyu0 jshoe-root
                                          (log of our su access to root)
```

By default, HP-UX passwords must be constructed to meet the following requirements:

- Each password must have at least six characters. Only the first eight characters are significant.

- Characters must be from the 7-bit USASCII character set; letters from the English alphabet.

- Each password must contain at least two alphabetic characters and at least one numeric or special character. In this case, "alphabetic" means uppercase and lowercase letters.

- Each password must differ from the user's login name and any reverse or circular shift of that login name. For comparison purposes, an uppercase letter and its corresponding lowercase equivalent are treated as identical.

- New passwords must differ from the old one by at least three characters. For comparison purposes, an uppercase letter and its corresponding lowercase equivalent are treated as identical.

The objective in choosing a password is to make it as difficult as possible to guess, while also making it meaningful enough for the user to remember without the need to write it down. Passwords must be kept secret at all times, since its limited knowledge by other users is what allows us to uniquely identify a person requesting access to the system. User ids and passwords shared among a group of users should be avoided, if at all possible. Not only does this defeat the purpose of attempting to uniquely identify a user and limit the effectiveness of activity audit trails, the task of changing a password is much more difficult and often is a reason why some passwords are never changed.

Password security is the responsibility of the security administrator and every user who has a valid user id on the system. At the time a user is added to the system, a password should be assigned by the system administrator. Every user should have a unique user id (uid) and should have the same uid assigned for each system for which they have a login id. To verify that there are no duplicate user ids defined in the /etc/passwd file, perform the following:

```
$ sort -t: +2n /etc/passwd |\
> awk -F: '{if (duplicate == $3) print $1 " "
uname; uname = $1; \
> duplicate = $3}'
```

Under HP-UX, the individual user has the ability to change their login password if they know their current password by using the '<u>passwd</u>' command. After prompting for the current password, the system requests a new password twice, once for verification since the user's typing is not echoed to the display. The super-user is permitted to change any user password on the system, or can remove a password by simply pressing return when prompted for the new password.

Even though the passwords in the file are encrypted and not able to be decrypted, theoretically, someone who has access to the encryption algorithm used by the operating system to encrypt the user passwords and a list of the encrypted passwords from a system (simply a copy of the /etc/passwd file), could write a program to try many combinations of words in an attempt to find a matching encrypted password (in fact, this technique has been used to crack passwords on many systems). Even though there are a large number of combinations of the available letters and special characters, very often only a small subset is actually used - drastically decreasing the total number of possibilities. Since it is possible to programmatically (using the standard utility /usr/lib/makekey) test a large number of passwords against the encrypted passwords stored in the /etc/passwd file (such as all the words in the dictionary), here are some additional guidelines in choosing a password:

- Do not use any form of the words from the comment field portion of the /etc/passwd file.
- Do not use a spouse's or child's name.
- Do not use other data easily obtained or known about you (such as favorite foods, activities, sports, teams, characters, car models, etc).
- Do not use simple keyword sequences such as "1qwerty".
- Do use a password with mixed upper & lower case letters.
- Do use a password that is easy to remember. Try misspelling words, or making nonsense words from multiple syllables. "Pass-phrases" are good - choosing the first letter from a phrase, such as "All men are created equal" - yields "Amace".
- Passwords should never be stored as part of a file on the computer - either on the Unix system or a PC terminal emulator application logon script.
- Force users to change their passwords on a regular basis.
- <u>Every</u> user id defined in /etc/passwd should have a password.
- Entries in /etc/passwd for placeholder or no-user accounts should have an impossible password ("*"), and if possible a startup program of "/bin/false".

After making any changes to either the /etc/passwd or /etc/groups files, use the HP-UX utilities <u>pwck</u> and <u>grpck</u> to scan the files and report any inconsistencies. The checks include validation of the number of fields, login name, user ID, group ID, and whether the login directory and optional program name exist.

Password aging is put in effect for a particular user if his encrypted password in the password file is followed by a comma and a non-null string of characters. (Such a string must be introduced in the first instance by the super-user.) This string defines the "age" needed to implement password aging. The first character of the age - such as M - denotes the maximum number of weeks for which a password is valid. A user who attempts to login after his password has expired is forced to supply a new one. The next character - such as m - denotes the minimum period in weeks that must expire before the password can be changed.

A second password file can also be established by the system administrator (/.secure/etc/passwd) to maintain the encrypted passwords on the system and prevent non-privileged users from viewing them. Unlike the /etc/passwd file which is readable by all users of the system, /.secure/etc/passwd is accessible by super-users only. The passwords contained in /.secure/etc/passwd take precedence over those contained in the encrypted password field of /etc/passwd. User authentication is done using the encrypted passwords in this file. The password aging mechanism described above also applies to /.secure/etc/passwd.

Default and Obsolete Accounts

On many systems, especially those with a large number of user ids defined, there typically exists a number of user ids which have been setup over the years for special purposes and are no longer of value. These may have been required by 3rd party software, a demonstration id for evaluation software or for users to access for playing games, etc. These accounts can very often be a large security hole in that either they have no passwords or easily guessable passwords. These types of accounts, if necessary, should only be created for a specified period of time and with a password that follows the conventions described above. Some accounts setup for use by the operating system should have impossible passwords set, including: (daemon, bin, adm, uucp, lp, hpdb, tftp and ppluser). These user names/ids are used by the various HP-UX subsystems to establish proper file access right sets for the files associated with the subsystem.

Notification of the security administrator or system manager should be a part of the processing policy when an employee leaves the company. It is important to the security of the system(s) to remove all user ids associated with the employee and remove and/or archive their files. If the ex-employee had access to root privileges, all passwords on the system, including root, should be modified as soon as possible. Using 3rd party software or customized shell scripts, procedures can be implemented to remove all files associated with a user id, and also check for accounts which have not been accessed for a period of time.

Login scripts

The first level of defense that the system administrator can use to protect the system is through the login process (user name and passwords). Login scripts provide another level of control during the signon process. Similar to "OPTION LOGON" UDCs on the HP3000, logon shell scripts are executed by the user's shell (command interpreter) before presenting the user with a prompt or initiating an application. As mentioned previously, the "shell" is the part of the Unix operating system which processes the commands entered by a user. This is actually nothing more than a program with special capabilities, and over the years a number have been developed. The four primary shells provided by HP-UX are:

/bin/sh	"Bourne" shell
/bin/ksh	"Korn" shell
/bin/csh	"C" shell - with "C" language type syntax.
/usr/bin/keysh	Function key driven shell, based on the ksh.

In general, each of these shells have similar functionality, but have implemented commands and options differently. There are also versions of these shells known as "restricted" shells (typically denoted by the letter 'r' before the name, such as /bin/rksh). These restricted shells provide only a subset of the capabilities to the user; however, there has been documented potential ways around some of the restrictions. This set of shells also use slightly different methods for processing login script files, although all have the ability to process two sets of scripts, one executed for all users logging on the system and another executed on a user by user basis (similar to a system-wide UDC and a user UDC on the HP3000).

Shell	Global Logon Script	Per User Logon Script
/bin/sh, /bin/ksh, /usr/bin/keysh	/etc/profile	$HOME/.profile
/bin/csh	/etc/csh.login	$HOME/.login

Standard actions performed by the global logon scripts typically include:

- Setting the 'PATH' variable (similar to the PATH command in MS-DOS®, which defines which subdirectories the shell will search for commands when processing user input, or the HPPATH variable in MPE/iX).
- Setting the system name and timezone variables.
- Configuring the type of terminal being used.
- Display the copyright notices and any "Messages of the Day" (/etc/copyright & /etc/motd).
- Check if your user id has been sent any mail.

In addition, this global script is a prime location for executing a second script which could provide additional access protection, such as prompting the user for additional passwords if the device is a modem, restricting access to certain user ids based upon time of day, day of week or device file. HP-UX also has a standard feature for providing additional security for dialup (or other tty) devices. Two files, /etc/dialups & /etc/d_passwd, can be used to force the user of specified devices to enter another password based upon the program name from /etc/passwd (the startup shell program name). The root user startup script has been a target for hackers by inserting commands into the file, if the file permissions allowed a non-privileged user write access. These commands, added to the file, very often will create what is known as a "setuid" to root copy of one of the standard shell program files in another location, with execute permission for all users. This allows any user on the system which knows about the program to effectively run as if they were logged in as the root user (super-user), and gain full access to restricted commands and files. In the next section, file access rights and setuid flags are discussed in greater detail.

Controlling File and Resource Access

After a user has passed the login process and startup profiles, they will be executing the program file specified in the last field of the /etc/passwd file for their user name. If the user has logged-on illegitimately, at this point the major part of the work of breaking into a system is complete. The potential for finding some type of security loophole is much greater since the user is able to search the system for programs, scripts, profiles or data which may provide them with information regarding other systems on other networks, dialup information to other systems, or even the ability to acquire super-user privileges. The Unix system's file (and device) access rights are based upon three categories of users: owner (the "owning" user id of the file), the group (the "owning" group id of the file), and all others (world), and three types of access: read, write and execute. These combinations provide a set of nine permissions associated with each file (and directory) on the system. This is similar to the MPE/iX file access rights available on a file (read, write, append, lock, execute), and the user classification (creator, group user, account librarian, group librarian, account user, etc). The ls command can optionally display these permission rights for any file on the system, for example:

```
$ ls -l /
total 5628
drwx------    2 root     mail          24 Feb  1 12:15 Mail
-rwxr-xr-x    1 root     sys      1168403 Feb  4 20:07 SYSBCKUP
-r-xr-xr-x    1 bin      bin       152178 May 13  1991 SYSDEBUG
-r-xr-xr-x    1 bin      bin       299064 May 13  1991 SYSTEST
drwxr-xr-x    3 root     root        4096 Feb  3 18:57 bin
drwxr-xr-x   10 root     root        5120 Apr 20 08:55 dev
drwxr-xr-x   12 root     root        7168 May  4 10:49 etc
-rwxr-xr-x    1 root     sys      1168403 Feb  3 21:06 hp-ux
dr-xr-xr-x    3 bin      bin         1056 Apr 11  1991 lib
drwxrwxrwx   11 root     root        3072 May  6 23:08 tmp
dr-xr-xr-x    8 bin      bin         1024 Mar 25 20:22 users
dr-xr-xr-x   30 bin      bin         1024 Feb 25 08:45 usr
```

The first section of this listing reflects each file's access rights. The first character identifies any special file attributes (i.e. d=directory, l=symbolic link, b=block device file, c=character device file, etc), and the remaining nine characters are logically divided into groups of three characters, with each group representing the rights for the file owner, group member and others, respectively. Within each character group, r, w, x represent read, write and execute access. A dash (-) signifies that a specific type of access is disabled for that user classification. The third and fourth columns identify the file's "owner" user name and member group name. It is for this user and group that the "owner" and "group" file permission flags apply. File permissions may be changed on a file (assuming the user has access to the file) by using the chmod command. A file's "owner" or "group" may also be modified by using the chown and chgrp commands by the file's owner or super-user. In addition, a file's contents can be encrypted using the crypt command. Although not too difficult for a serious hacker to decode, it can offer some additional level of security for some non-sensitive material against casual reading of a file's contents. However, encrypting files should not be used as a replacement for proper file permission settings.

With MPE, there exists a default set of file access rights based upon the group and account in which the file is created. With Unix, file permissions for newly created files are controlled by two mechanisms. The first is the program or command this is actually creating the file. Different programs and subsystems supplied with Unix create files with different default access rights. This could be a potential problem since some create the files with all access permissions enabled for all users. However, a second mechanism called umask (user mask) enables a user to define what the default file permissions should NOT allow. The umask command defines on a permission by permission basis (for all nine settings) which should be denied. A default user mask setting for strong security may be: 'umask 77', which means to deny all access other than for the file owner (and super-user). The user mask setting is often executed in the login profile script (.profile or .login) and remains in effect as long as the shell is executing.

In MPE there are additional discretionary access controls known as "Access Control Lists". These ACLs allow a file creator to identify specific users (identified by name) specific access rights to a file or set of files. This same capability exists within HP-UX. Using ACLs provides a method of specifying access rights for files to a much greater detail then standard Unix file permissions. Like standard permissions, ACLs can be changed by using the chacl command and listed using the lsacl command. Some standard Unix commands do not understand ACLs and consequently may cause them to be removed (i.e. chmod without the -A option, cpset, tar, cpio, etc). If ACLs are used, system backups should only be performed using the fbackup and frecover utilities, since only these programs will save and restore ACL settings on files.

Directories are actually files in Unix that have a predefined structure which creates a linkage between file names and "inode" indexes. "inode" entries contain information about the file such as file location, permissions, owner, group, etc. So directories are files which contain lists of files and their corresponding inode numbers. It is possible for multiple file names to have the same inode number, and this is a special file type known as a "link". Both file names actually reference the same physical locations on disk (or the same physical device). A directory file (like a standard file) has a set of permissions associated with it. For a directory, these permissions have different interpretations however:

Read — permits listing the filenames contained in that directory (via the ls and other commands).

Write & Execute — permits changing or removing file names in a directory.

Execute — permits access to the files which are referenced by file names in the directory (inode).

If a user is allowed write and execute to a directory, any file in that directory can be removed, regardless of the individual file permission settings. This combination also allows renaming files and creating new files in the directory. This combination allows the possibility of a user creating unwanted files in a directory which could compromise system security (i.e. a Trojan Horse). Removing read access from a directory removes the ability of listing the file names. However, a specific file could still be accessed if the name was known. Some Unix subsystems take advantage of this ability for maintaining security while still allowing some access.

Home directories of users should not be writeable by all users, since a file could be added or removed by any user on the system. A command to check for writeable directories that are 'home' to a user is:

```
$ find `awk -F: '{print $6}' /etc/passwd` \
          -prune -perm -02 -exec ls -ld {} \;
```

The PATH variable, which is usually set in either the global profile script or the user script, identifies the directories the user's shell program will search when attempting to execute a command or script. If the user does not qualify a command or script name (by specifying the directory in which the file is located), the shell will by default search each directory in the PATH variable for the specified name. This functionality makes using Unix much easier, but also opens the door for potential disaster. For example, if the PATH variable consisted of the directory "." (meaning the current directory) or any world-writeable directory, a file or script may be executed which was not intended by the user. In the current director a file named 'ls' could exist which could remove all files in the directory. When an unsuspecting user which is in that directory attempts to list the files (or so they think), all files are removed for which the user has write permission. This use is fairly obvious, however consider the same PATH variable set for the root user. A script with the same name as one or more common commands could exist which perform some function, such as creating a new user id with super-user privileges, and then execute the original command. The unsuspecting user would not even know that the bogus script had even been executed. Therefore, the path for superusers should only contain the directories '/bin' and '/usr/bin', with appropriate security maintained on each of those directories.

With this in mind, the user-level profile (.profile or .login) should contain at least the following security related commands:

- Insure the PATH variable is set securely
- Use umask to set greater restrictions on new files.
- Display the last time this user logged on.

This could be accomplished with the following shell script portion:

```
PATH=/bin:/usr/bin:/usr/bin/X11
export PATH
umask 27
set - `ls -lc .last_login`
echo "The last record login was on $6 $7 at $8"
touch .last_login
```

A user's login startup script should also not allow anyone else on the system modification. To verify compliance use the following commands:

```
$ find `awk -F: '{printf %s/.profile\n", $6}' /etc/passwd` \
       -prune -perm -022 -exec ls -l {} \;
$ find `awk -F: '{printf %s/.login\n", $6}' /etc/passwd` \
       -prune -perm -022 -exec ls -l {} \;
$ find `awk -F: '{printf %s/..cshrc\n", $6}' /etc/passwd` \
       -prune -perm -022 -exec ls -l {} \;
```

Device Files

In Unix, like MPE, physical devices are referenced and accessed via standard file names. However, unlike MPE, Unix actually has a file that exists in the directory structure for the devices on the system, including device files that refers to memory, networking interfaces, swap areas, and kernel virtual memory. Granting access to these files, allows access (read/write) to the devices themselves. By convention, these device files reside in the /dev directory, although a device file could exist in any subdirectory.

Unsecured device files could not only open up the possibility of a hacker being able to read any portion of the disks or memory, but also the ability to write over any data contained on a device, thus potentially destroying data or the operating system itself. In general, make sure that any disk, memory (/dev/mem & /dev/kmem) and other non-terminal device file is secure from all "other" access (i.e. chmod 600 <files>). One exception may be to allow greater access to a magnetic tape or cartridge tape drive file, depending on the operating environment. The owner of these files should be 'root', and the group id 'sys'. Changing the group id associated with a set of device files would allow for certain classes of users access to devices. For example, an 'operator' group may allow access to the tape device files for backup and recovery purposes. ACLs could also be used to further define access to a specific set of users for special purposes. For management purposes, device files should only reside under the /dev directory. The following command will locate any block or character device file on the system:

```
$ find / -hidden -name /dev -prune -o \( -type b -o \
       -type c \) -exec ls -l {} \;
```

Directly attached terminal devices also have a device file associated with them, known as a 'tty' device (since the name usually begins with the letters 'tty'). The permissions and owner/group ids of these device files are modified during the login process to allow access for the current logged on user. Typically, the "other" write permission is also allowed - this permits any user on the system to send messages directly to your screen through the write command (or directly by accessing the device file). This write permission can be disabled using the mesg n command (mesg y re-enables write access). This is similar to "SETMSG OFF/ON" in MPE/iX, which disallows a terminal from receiving messages from users via the "TELL" command. As soon as the user logs off the system, the ownership of the tty device file is changed back to 'root'.

Set User ID (setuid) and Set Group ID (setgid)

There are two additional flags maintained as part of the file permission settings known as the setuid and setgid bits, identified by the letter 's' or 'S' in the owner-execute (for setuid) or group-execute (for setgid) positions of the file permission settings.

Whenever a program (or shell script) is executed, there are four numbers associated with that process: the real & effective user id, and the real & effective group id. During the login process the real & effective id pairs are set to the user id and group id from the /etc/passwd file for the user name. However, when a user executes either a setuid or setgid program, the corresponding effective id is changed to that from the inode of the file. The user then has temporary rights and privileges of the file's owner and/or group.

The most dangerous case exists for "setuid to root" programs and shell scripts. In this case the owner of the file is the root user and the setuid bit is set (this bit can be set via the chmod command), and the executing user will temporarily have super-user privileges, until the program terminates. Certain programs within Unix use the capability to perform the necessary work, such as passwd, which requires temporary superuser rights to be able to change the password field in the /etc/passwd file. Setuid or setgid shell scripts should never be used or exist, since they represent a large potential security problem. It is much easier for someone to change a shell script to add "hidden" functionality than it is to modify an executable file. A common tactic used by hackers of Unix systems is to create a copy of one of the standard shell programs in a directory in which any user has execute access to program files and change the owner to root and set the setuid bit. From then on, any knowing user can execute the setuid-to-root shell and gain complete access to files and commands. As described above, this act of copying and changing modes could actually be performed by a legitimate super user without their knowledge, if the PATH variable is not set properly.

To locate setuid and setgid programs on the system, the following commands could be used (setuid-to-root programs could be critical, setuid to other users is less critical, but both should be investigated):

```
$ find / \( -perm -4000 -o -perm -2000 \) -exec ls -ld {} \;
```

Monitoring System Activity

Maintaining a secure system is a process, and aside from establishing the policies, procedures and controls to restrict access to the system and resources, monitoring the activity on the system is required to insure compliance with the policies. This involves examining system logs of various activity and searching for extraordinary circumstances, identifying modifications to established security controls and in general, islocating attempts to defeat the security mechanisms of a system.

Unix systems can be enabled to log both successful and unsuccessful attempts a logging onto the system. These log files can exist on the system to record access information:

Log File	Type of information	Accessed By:
/etc/utmp	Records which users are currently logged on (similar to JMAT table in MPE)	who command
/etc/wtmp	Records each login and logout information.	who /etc/wtmp /etc/last (matches login & logout records to show time logged on)
/etc/btmp	Records unsuccessful login attempts	/etc/lastb
/usr/adm/sulog	Records successful & unsuccessful attempts to "switch user" with the su command.	more (file is ASCII readable).

These files are only written to by the login & su programs if they already exist, otherwise no record is maintained. To initially create these files, login as root and touch each of the file names to create them. These logs are similar to the logging of "Job Initiation" and "Job Termination" events within MPE logging.

HP-UX also has additional accounting facilities for logging a variety of additional information. This accounting information can be enabled (usually at boot time in the /etc/rc file) by executing the /usr/lib/acct/startup command. Additional process-level logging information is recorded in various files, including /usr/adm/pacct. The HP-UX accounting facility includes a number of utilities to provide formatting and summarization of the logged information, including:

acctcms	Produce command execution summary report (used by runacct
acctcom	Produce detailed process reports from /usr/adm/pacct (used by runacct)
acctconl & acctcon2	Generate connect time information reports (used by runacct)
acctprcl & acctprc2	Generates CPU usage information by user id (used by runacct)
acctdusg	Calculates disk usage by login name
ckpacct	Checks the /usr/adm/pacct log file. If it has exceeded a specified size, the existing log is renamed and a new one created. (similar to SWITCHLOG in MPE)
dodisk	Checks current disk space usage and creates a "snapshot" of the information.
monacct	Produces monthly accounting report files.
prdaily	Generate previous day's accounting report
runacct	Produces daily accounting reports
startup	Enables logging of additional accounting information
shutdown	Appends a shutdown record and disables logging

These accounting commands and utilities are typically used in conjunction with cron to execute accounting report daily (and/or monthly) and to monitor the size of the log files. These tools can provide very detailed historical information of activity of users on the system and may be very useful in determining a sequence of events in the event of a security breakin.

In addition, HP-UX can be converted to a "Trusted System" fairly easily using the "SAM" utility. Prior to converting to a trusted system, all known potential security holes should be removed, permissions modified and user ids validated. Once converted to a trusted system, the only way to remove this state is to re-install the operating system.

Once executing in a trusted state, additional auditing features are available to the system administrator. These include:

- Ability to audit specific user ids
- Audit logins, run state changes

- File permission changes, acl and ownership changes
- Administrative commands, file creation, deletion and access, process creation and termination
- User-defined logging events
- Some network related events (socket level calls)

Certain programs, utilities and portions of the kernel have been instrumented to capture and log these events when executing as a trusted system. Primary and secondary log files are maintained, and reporting tools are provided for analysis. However, auditing does increase system overhead, and performance considerations must be weighed against the security information. Selective logging of specific events can reduce the amount of overhead on the system substantially.

Network Security

Security of the networking components of a system (i.e. UUCP, NFS, ARPA/Berkeley services, NS, RPC, DCE) are unique to each service. Specific security issues relating to these products are not addressed by this paper;, however, if a system has any of these services installed, the security controls available should be reviewed, understood and implemented since many of these services can gain access to a system without requiring the login process. Specific to review include:

- inetd.conf, /etc/services and inetd.sec file parameters and permissions
- .rhosts file usage
- hosts.equiv
- /etc/exports & /etc/netgroup
- Anonymous FTP capability

Summary

This paper has just provided an introduction to many of the Unix system security related issues which have surfaced over the years. It is the responsibility of the management of an organization and every user of the system to maintain security in their "operating domain". Without the full commitment of all involved, the task of attempting to maintain a secure system by a few individuals is difficult, if not impossible. Upfront planning, through establishing policies and procedures is key to the successful implementation of security controls and the on-going monitoring and auditing activities required of the system administrator. These activities must become a standard operating procedure in the day to day maintenance of the system, and plans must be established in advance for handling and recovering from a potential security breech. "An ounce of prevention is worth a pound of cure."

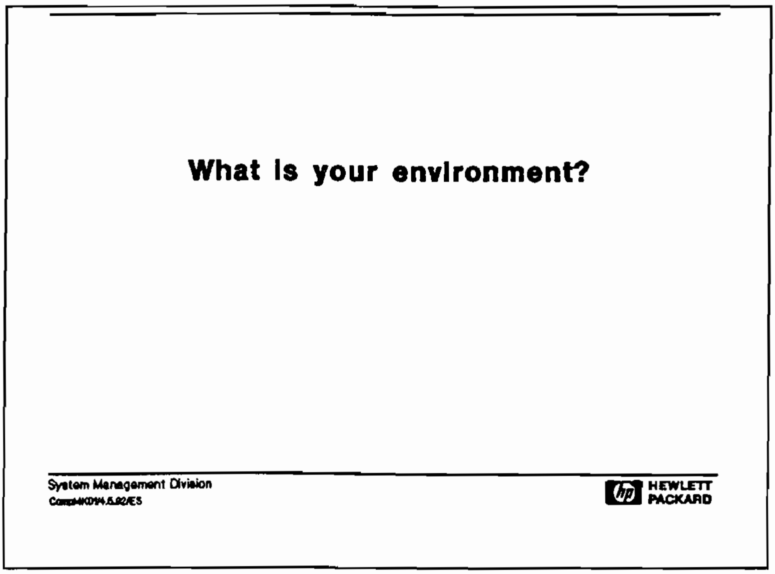
Paper 2031

The Real Cost Savings in Managing Open Systems

Authors: Reiner Lomb
Martin Kellner

HEWLETT PACKARD

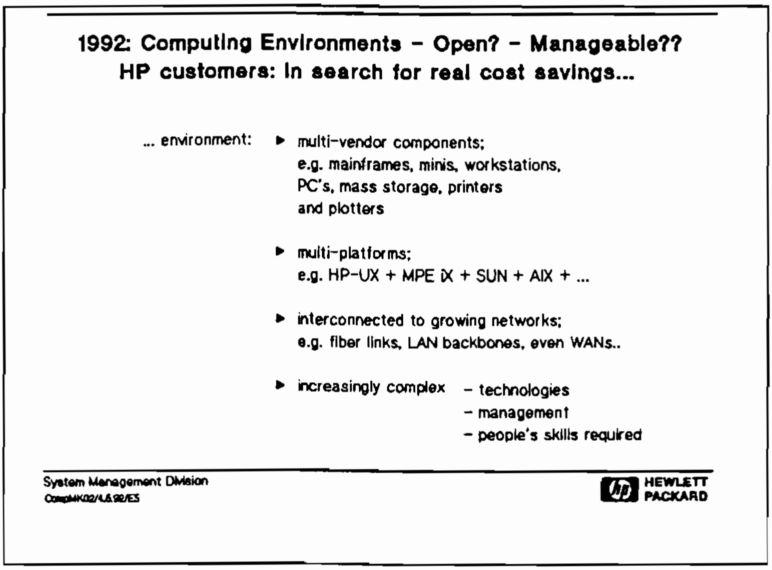
Hewlett-Packard GmbH
Herrenberger Str. 130
D 7030 Böblingen, Germany
Phone: +49 7031 142060



What is your environment?

System Management Division
CompMK01/4.5.92/ES

HEWLETT PACKARD



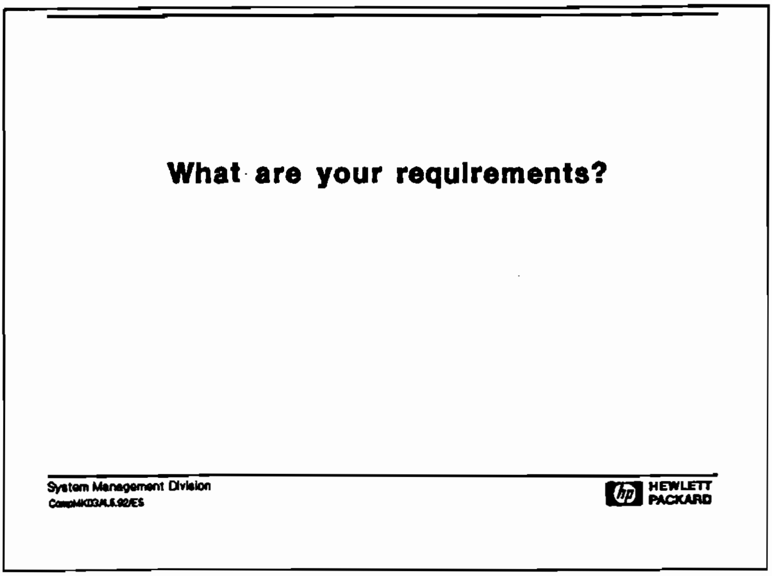
1992: Computing Environments – Open? – Manageable??
HP customers: In search for real cost savings...

... environment:

- multi-vendor components;
  e.g. mainframes, minis, workstations,
  PC's, mass storage, printers
  and plotters
- multi-platforms;
  e.g. HP-UX + MPE iX + SUN + AIX + ...
- interconnected to growing networks;
  e.g. fiber links, LAN backbones, even WANs..
- increasingly complex
  – technologies
  – management
  – people's skills required

System Management Division
CompMK02/4.5.92/ES

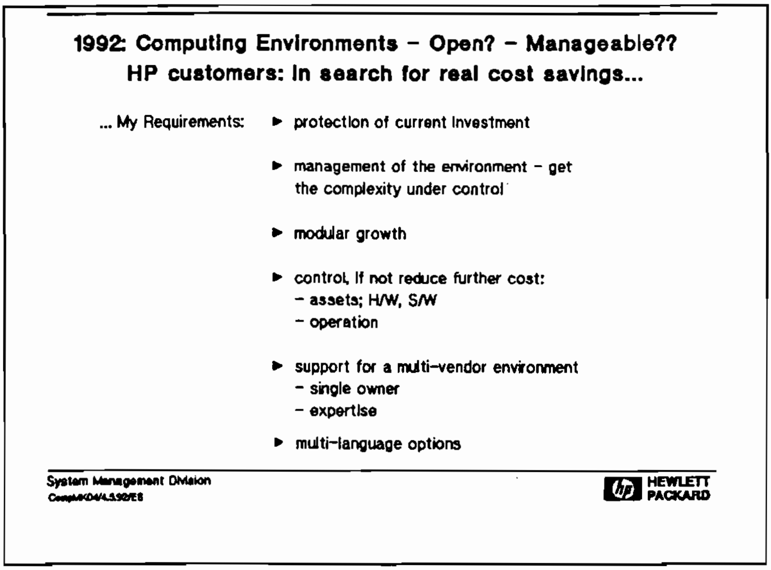
HEWLETT PACKARD



What are your requirements?

System Management Division
CompMK03/4.5.92/ES

HEWLETT PACKARD



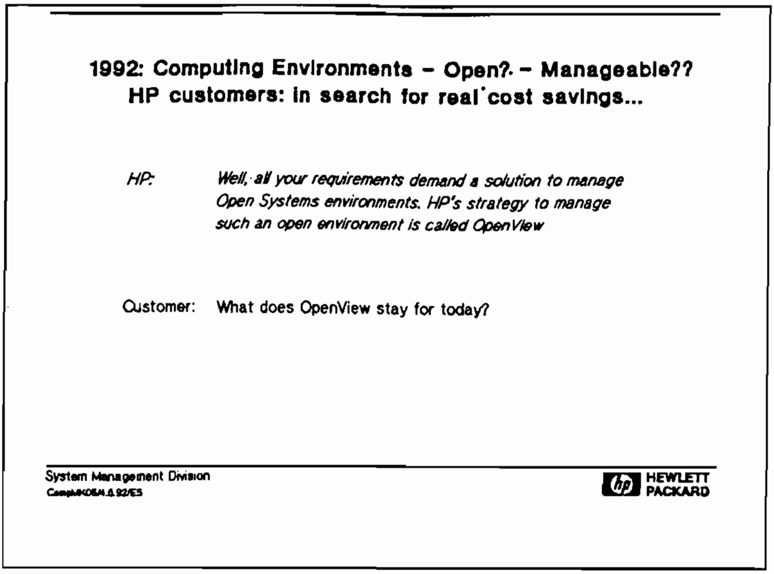
1992: Computing Environments – Open? – Manageable?? HP customers: In search for real cost savings...

... My Requirements:

- protection of current investment
- management of the environment – get the complexity under control
- modular growth
- control, if not reduce further cost:
  - assets; H/W, S/W
  - operation
- support for a multi-vendor environment
  - single owner
  - expertise
- multi-language options

System Management Division
CompMK04/4.5.92/ES

HEWLETT PACKARD



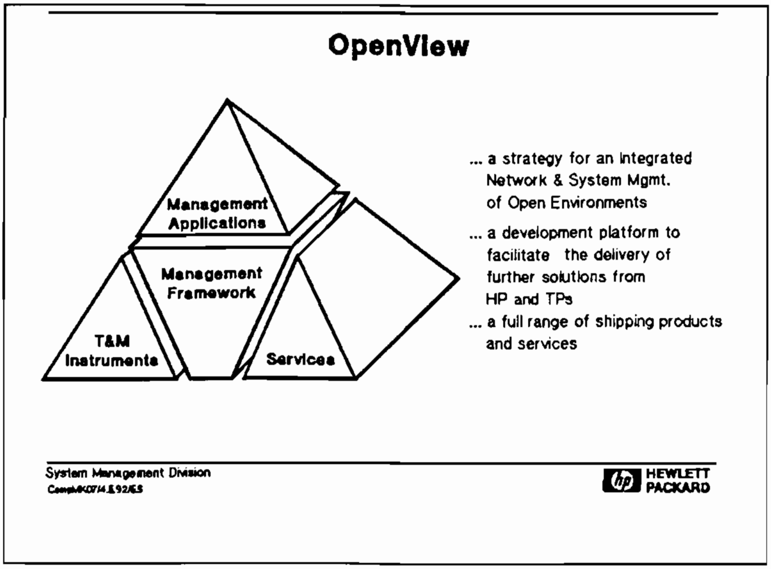
1992: Computing Environments – Open?- – Manageable??
HP customers: in search for real'cost savings...

HP: *Well, all your requirements demand a solution to manage Open Systems environments. HP's strategy to manage such an open environment is called OpenView*

Customer: What does OpenView stay for today?

System Management Division
CompMK05/4.6.92/ES

HEWLETT PACKARD



OpenView

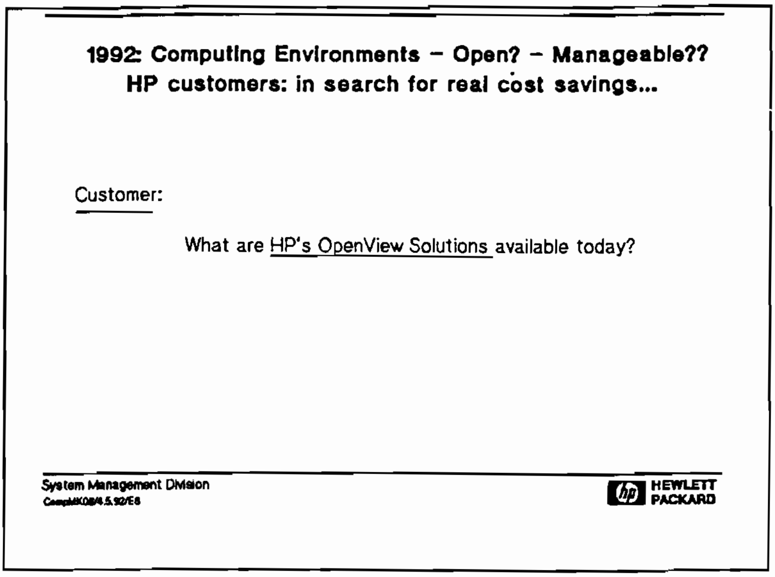
... a strategy for an integrated Network & System Mgmt. of Open Environments

... a development platform to facilitate the delivery of further solutions from HP and TPs

... a full range of shipping products and services

System Management Division
CompMK07/4.5.92/ES

HEWLETT PACKARD



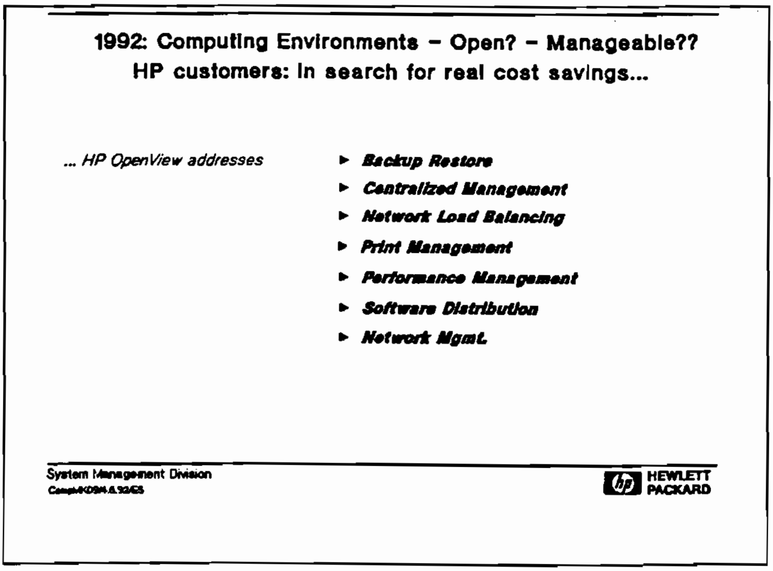
1992: Computing Environments – Open? – Manageable??
HP customers: in search for real cost savings...

Customer:

What are HP's OpenView Solutions available today?

System Management Division
CompMKO8/4.5.92/E6

HEWLETT PACKARD



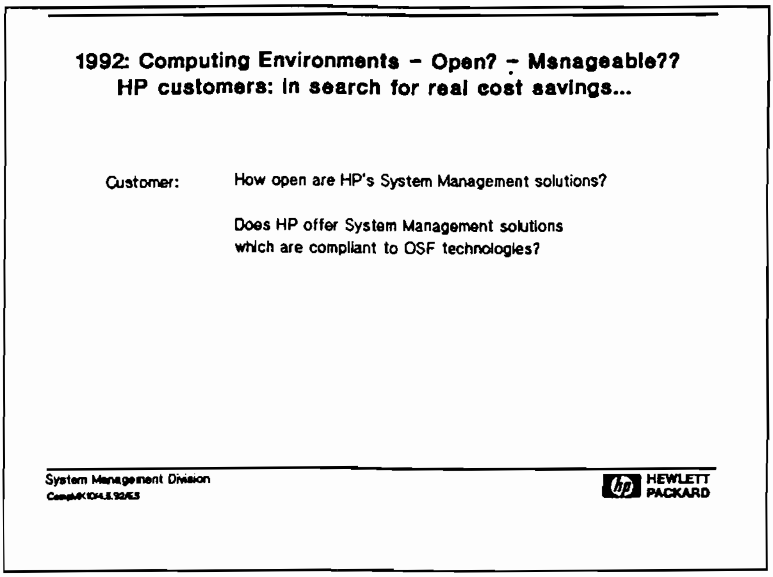
1992: Computing Environments – Open? – Manageable??
HP customers: in search for real cost savings...

... HP OpenView addresses

- ***Backup Restore***
- ***Centralized Management***
- ***Network Load Balancing***
- ***Print Management***
- ***Performance Management***
- ***Software Distribution***
- ***Network Mgmt.***

System Management Division
CompMKO9/4.5.92/E5

HEWLETT PACKARD



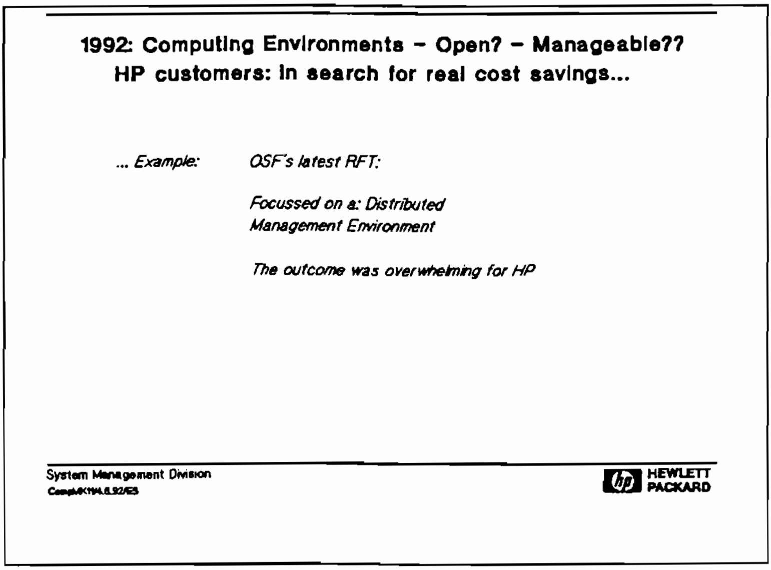
1992: Computing Environments – Open? – Manageable??
HP customers: In search for real cost savings...

Customer: How open are HP's System Management solutions?

Does HP offer System Management solutions
which are compliant to OSF technologies?

System Management Division
CompMK104.5.92/E3

HEWLETT
PACKARD



1992: Computing Environments – Open? – Manageable??
HP customers: In search for real cost savings...

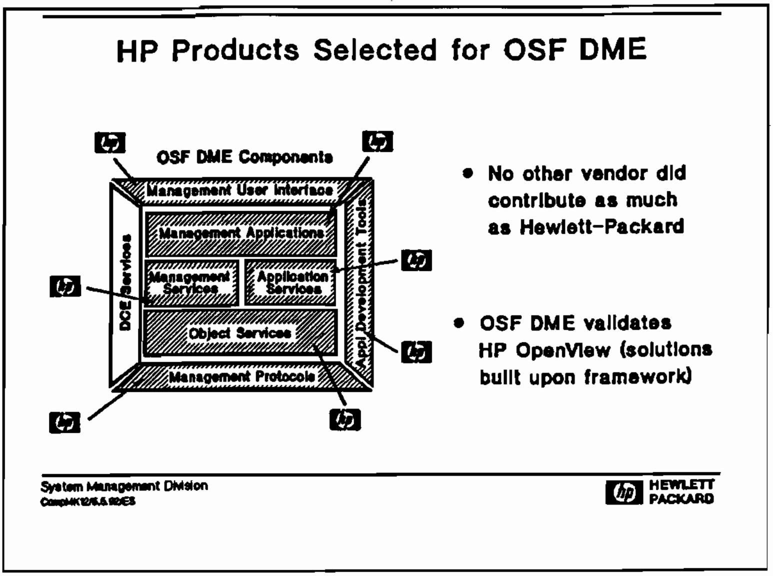
... Example: *OSF's latest RFT:*

Focussed on a: Distributed
Management Environment

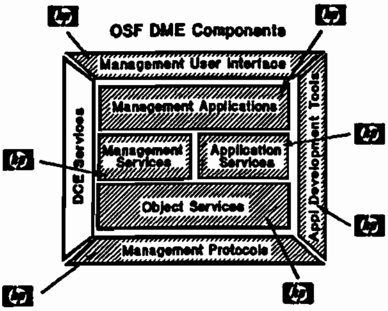
The outcome was overwhelming for HP

System Management Division
CompMK1W4.6.92/E3

HEWLETT
PACKARD



HP Products Selected for OSF DME

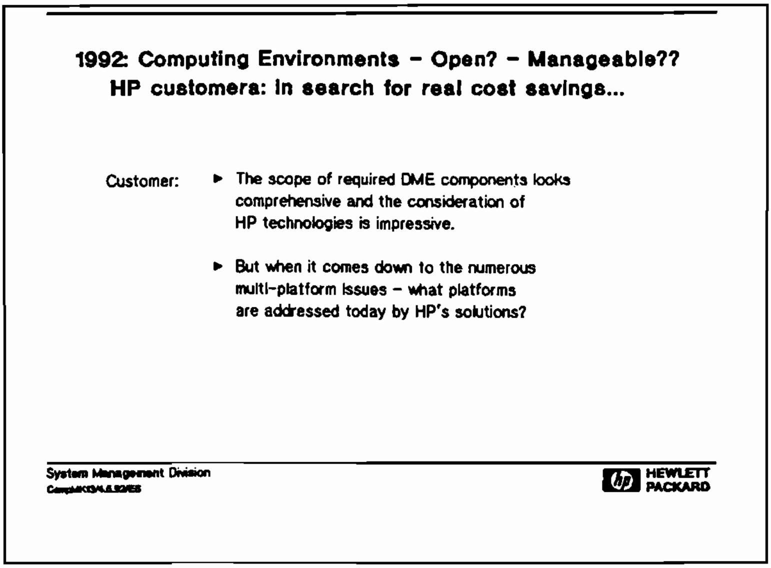


- No other vendor did contribute as much as Hewlett-Packard

- OSF DME validates HP OpenView (solutions built upon framework)

System Management Division
CompMK12/6.5.92/ES

HEWLETT PACKARD



1992: Computing Environments – Open? – Manageable?? HP customers: In search for real cost savings...

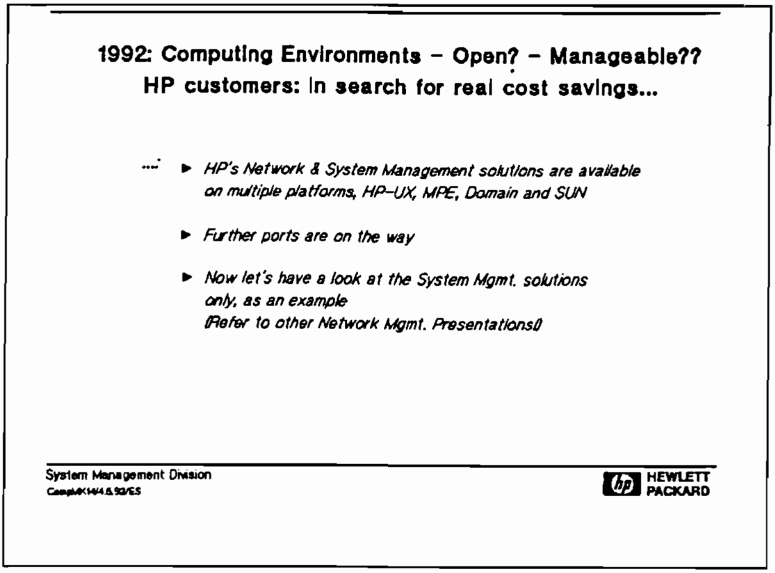
Customer:

- The scope of required DME components looks comprehensive and the consideration of HP technologies is impressive.

- But when it comes down to the numerous multi-platform issues – what platforms are addressed today by HP's solutions?

System Management Division
CompMK13/4.5.92/ES

HEWLETT PACKARD

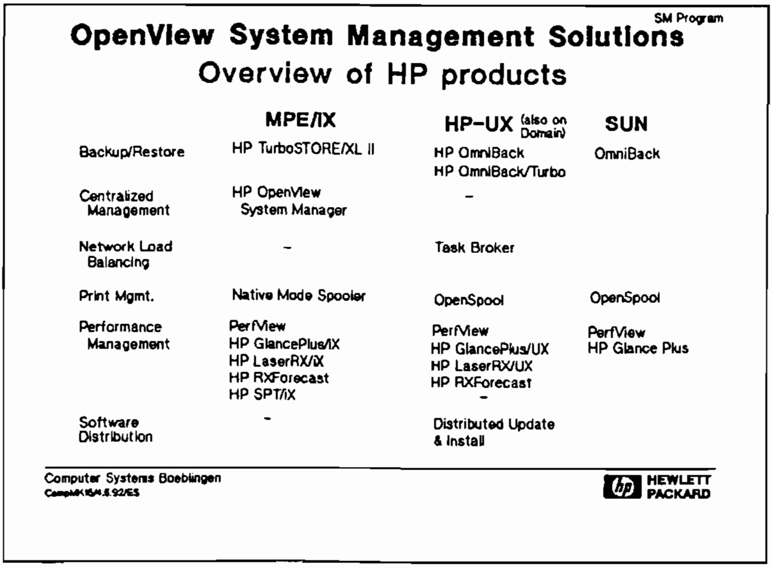


1992: Computing Environments – Open? – Manageable??
HP customers: In search for real cost savings...

- *HP's Network & System Management solutions are available on multiple platforms, HP-UX, MPE, Domain and SUN*
- *Further ports are on the way*
- *Now let's have a look at the System Mgmt. solutions only, as an example*
  (Refer to other Network Mgmt. Presentations!)

System Management Division
CompMK14/4.5.92/ES

HEWLETT PACKARD



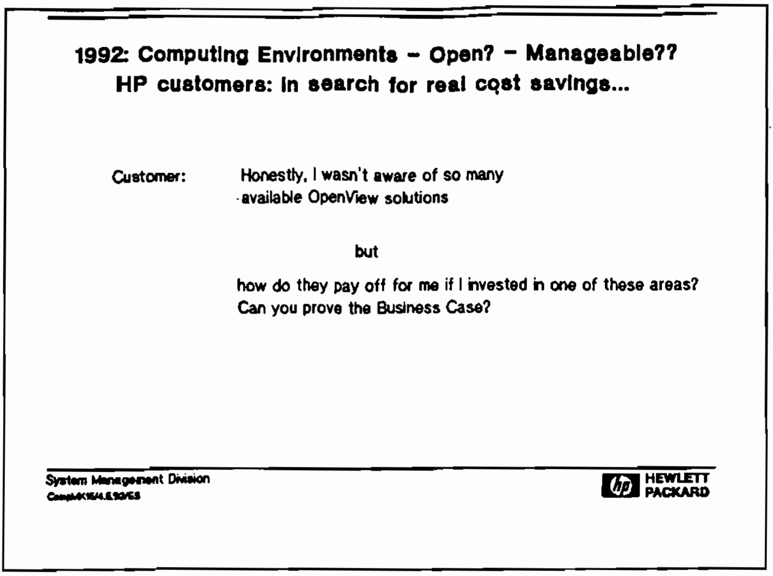
SM Program

OpenView System Management Solutions
Overview of HP products

	MPE/iX	HP-UX (also on Domain)	SUN
Backup/Restore	HP TurboSTORE/XL II	HP OmniBack HP OmniBack/Turbo	OmniBack
Centralized Management	HP OpenView System Manager	–	
Network Load Balancing	–	Task Broker	
Print Mgmt.	Native Mode Spooler	OpenSpool	OpenSpool
Performance Management	PerfView HP GlancePlus/iX HP LaserRX/iX HP RXForecast HP SPT/iX	PerfView HP GlancePlus/UX HP LaserRX/UX HP RXForecast –	PerfView HP Glance Plus
Software Distribution	–	Distributed Update & Install	

Computer Systems Boeblingen
CompMK15/4.5.92/ES

HEWLETT PACKARD



1992: Computing Environments – Open? – Manageable??
HP customers: In search for real cost savings...

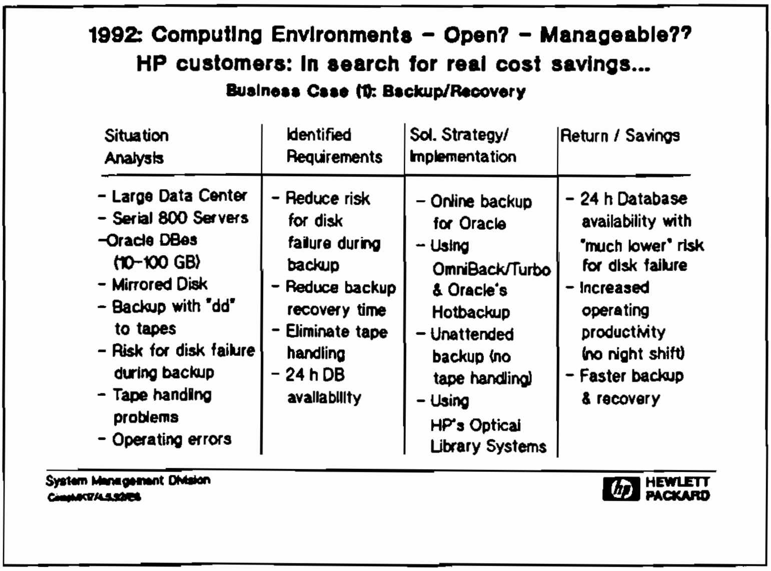
Customer: Honestly, I wasn't aware of so many available OpenView solutions

but

how do they pay off for me if I invested in one of these areas? Can you prove the Business Case?

System Management Division
CompMK15/4.5.93/ES

HEWLETT PACKARD

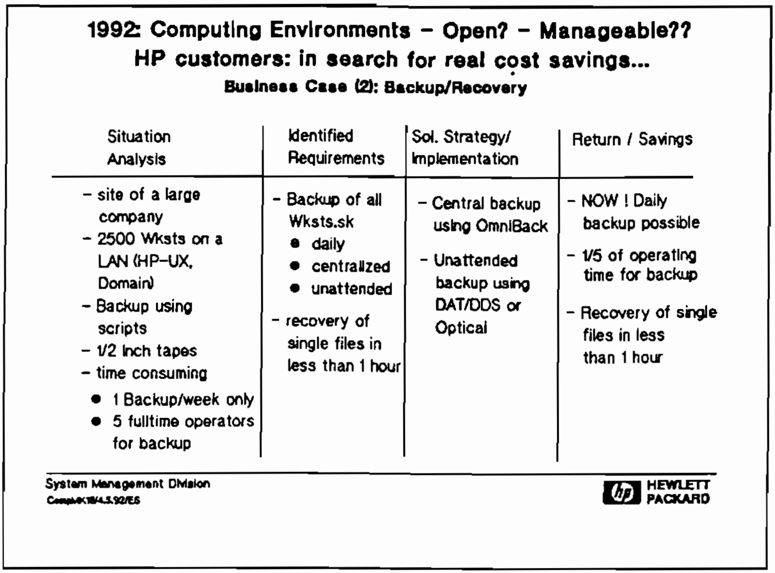


1992: Computing Environments – Open? – Manageable??
HP customers: In search for real cost savings...
Business Case (1): Backup/Recovery

Situation Analysis	Identified Requirements	Sol. Strategy/ Implementation	Return / Savings
- Large Data Center - Serial 800 Servers -Oracle DBes (10-100 GB) - Mirrored Disk - Backup with "dd" to tapes - Risk for disk failure during backup - Tape handling problems - Operating errors	- Reduce risk for disk failure during backup - Reduce backup recovery time - Eliminate tape handling - 24 h DB availability	- Online backup for Oracle - Using OmniBack/Turbo & Oracle's Hotbackup - Unattended backup (no tape handling) - Using HP's Optical Library Systems	- 24 h Database availability with "much lower" risk for disk failure - Increased operating productivity (no night shift) - Faster backup & recovery

System Management Division
CompMK17/4.5.92/ES

HEWLETT PACKARD



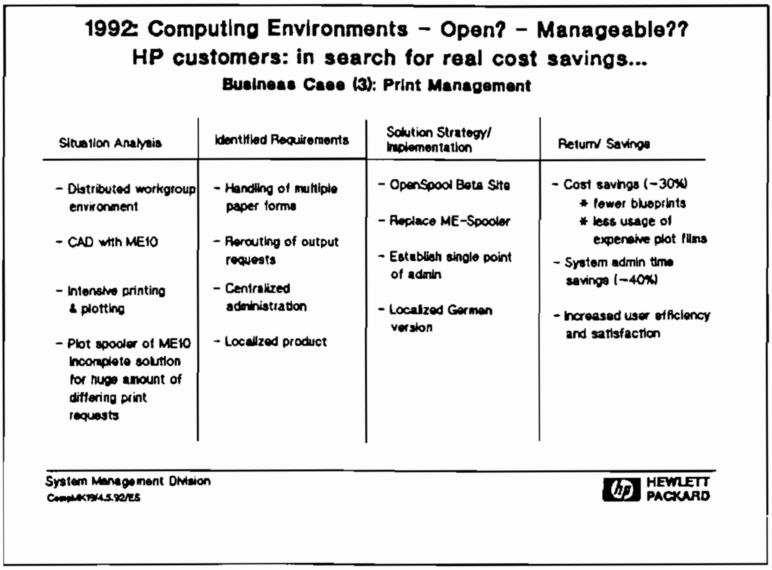
1992: Computing Environments – Open? – Manageable??
HP customers: in search for real cost savings...

Business Case (2): Backup/Recovery

Situation Analysis	Identified Requirements	Sol. Strategy/ Implementation	Return / Savings
- site of a large company - 2500 Wksts on a LAN (HP-UX, Domain) - Backup using scripts - 1/2 inch tapes - time consuming ● 1 Backup/week only ● 5 fulltime operators for backup	- Backup of all Wksts.sk ● daily ● centralized ● unattended - recovery of single files in less than 1 hour	- Central backup using OmniBack - Unattended backup using DAT/DDS or Optical	- NOW ! Daily backup possible - 1/5 of operating time for backup - Recovery of single files in less than 1 hour

System Management Division
CompMK19/4.5.92/ES

HEWLETT PACKARD



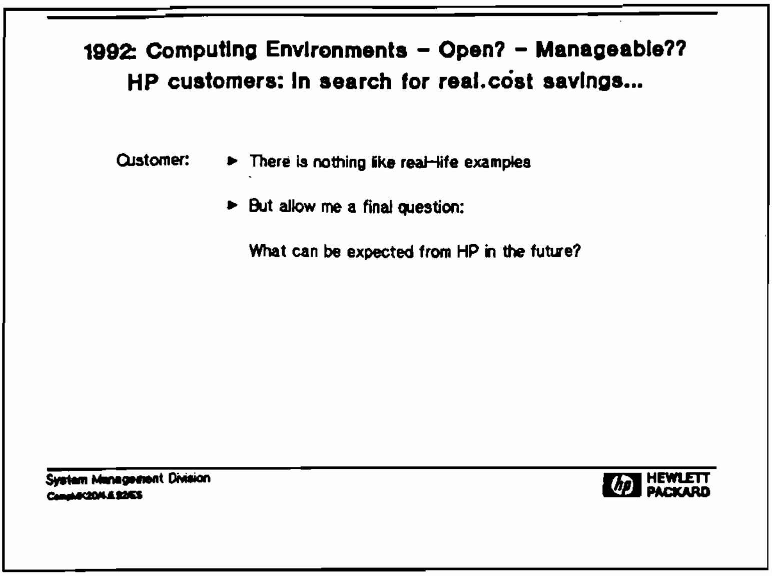
1992: Computing Environments – Open? – Manageable??
HP customers: in search for real cost savings...

Business Case (3): Print Management

Situation Analysis	Identified Requirements	Solution Strategy/ Implementation	Return/ Savings
- Distributed workgroup environment - CAD with ME10 - Intensive printing & plotting - Plot spooler of ME10 incomplete solution for huge amount of differing print requests	- Handling of multiple paper forms - Rerouting of output requests - Centralized administration - Localized product	- OpenSpool Beta Site - Replace ME-Spooler - Establish single point of admin - Localized German version	- Cost savings (-30%) * fewer blueprints * less usage of expensive plot films - System admin time savings (-40%) - Increased user efficiency and satisfaction

System Management Division
CompMK19/4.5.92/ES

HEWLETT PACKARD



1992: Computing Environments – Open? – Manageable?? HP customers: In search for real.cost savings...

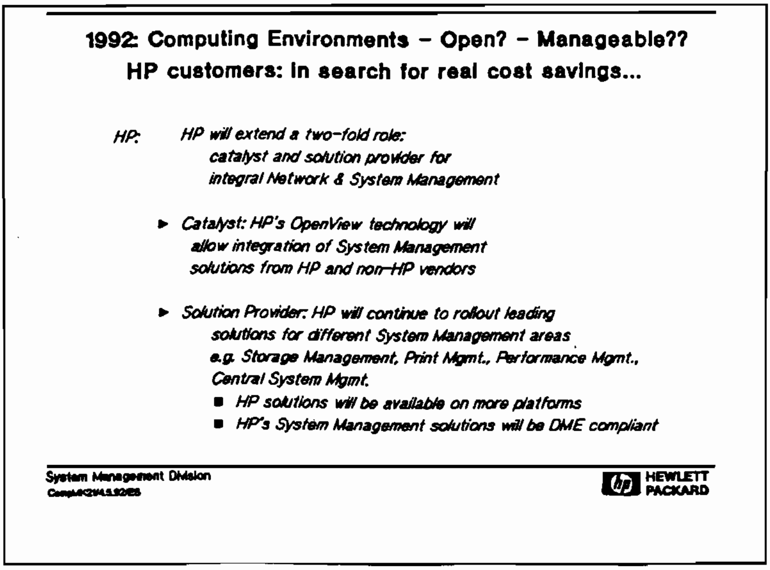
Customer:

- There is nothing like real-life examples
- But allow me a final question:

  What can be expected from HP in the future?

System Management Division
CompM/K20/4.5.92/ES

HEWLETT PACKARD



1992: Computing Environments – Open? – Manageable?? HP customers: In search for real cost savings...

HP: *HP will extend a two-fold role:*
catalyst and solution provider for
integral Network & System Management

- *Catalyst: HP's OpenView technology will allow integration of System Management solutions from HP and non-HP vendors*
- *Solution Provider: HP will continue to rollout leading solutions for different System Management areas e.g. Storage Management, Print Mgmt., Performance Mgmt., Central System Mgmt.*
  - *HP solutions will be available on more platforms*
  - *HP's System Management solutions will be DME compliant*

System Management Division
CompM/K21/4.5.92/ES

HEWLETT PACKARD

Paper # 2032

Harnessing CASE for Your Move to Open Systems

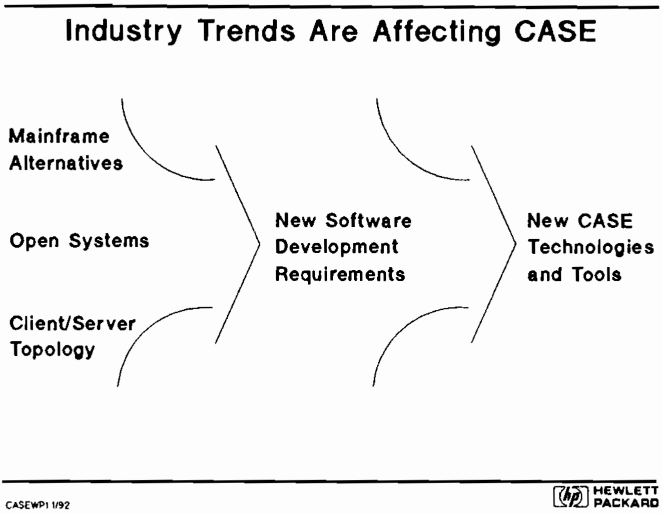
Casey Lemus
Hewlett Packard Company
19490 Homestead Road MS 41AK
Cupertino, California 95014
(408) 447-1358

Introduction

As business becomes increasingly driven by information, the ability to easily deploy and modify information systems can be a significant competitive advantage. Being able to deploy and modify applications faster can mean the ability to focus on new opportunities quickly, offer a better level of service, expand, consolidate, and keep pace with changing technologies. These days businesses are putting increasing emphasis on their software development efforts; evaluating their current procedures and looking into Computer Aided Software Engineering (CASE) to see what it has to offer.

Two important industry trends are turning out to have a uniquely synergistic effect on each other and CASE. First, the trend toward "downsizing" (also called "rightsizing" or "downcosting") that is; supplementing mainframe data centers with lower cost systems rather than continuing to add mainframes, and second, the growing utilization of open systems and interest in client/server environments. The synergy has resulted in a new make-up for data centers and in the availability of new CASE technologies that span the mainframe and open system (or "Unix") alternatives. See fig. 1.

Fig. 1



Industry Trends Are Affecting CASE

Mainframe Alternatives

Open Systems

Client/Server Topology

New Software Development Requirements

New CASE Technologies and Tools

CASEWP1 1/92

HEWLETT PACKARD

The benefits of this trend for CASE are the same benefits that CASE in general provides; increased developer productivity, better ability to maintain code, improved quality, more process control, etc. What's NEW is offering these benefits in a complex, multi-vendor environment.

Each computer user, each MIS department, each coding team has their own particular needs for software development. And how they choose to utilize CASE (or IF they choose to utilize CASE) is different for each organization. This paper will discuss different approaches and options for utilizing software development tools to introduce open systems into your information systems environment. This paper will refer specifically to the HP 9000 and its available toolsets as a "typical" open system to illustrate and describe these various approaches.

The HP 9000 in a Data Center Environment; Application Development and Maintenance

Within data center environments, HP 9000 systems can be used in many ways to complement mainframe or other proprietary systems; ranging from "coexistence" to off-loading of proprietary applications to creating entirely new applications. See fig. 2. The degrees of use include:

- Maintain/modify proprietary application from HP 9000

- Emulate proprietary application on HP 9000

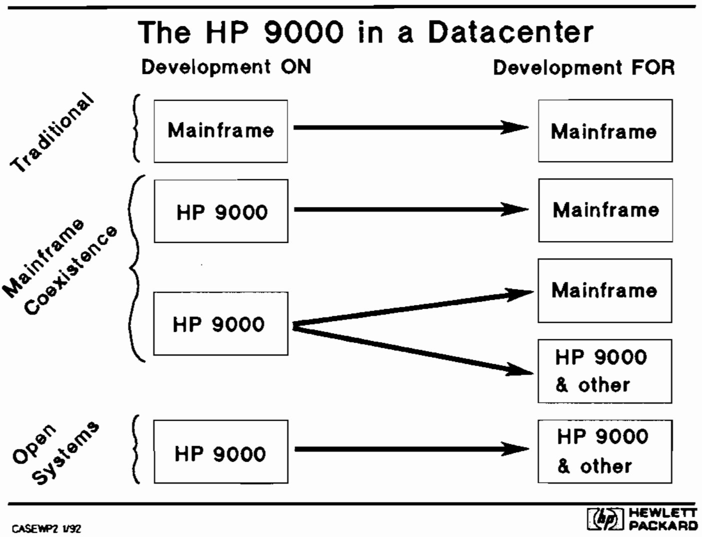
- Convert proprietary application to native HP-UX code

- Re-engineer application, then redevelop to run native on HP 9000

- Rewrite proprietary application to run on HP 9000

- Write new application on HP 9000 to take advantage of open systems technology

Fig. 2



The HP 9000 in a Datacenter

Which method makes sense? Is it better to take the "coexistence" approach or to select to start newly on the HP 9000? Many factors are involved, including the amount of time and resources available to make the transition and training of existing programmers. The movement to open systems covers a broad spectrum of activities ranging from fixing bugs on the legacy system to developing significant enhancements. At the far end of the spectrum is a complete rewrite of code. Whether you want to off-load development, execution, or both, a wide variety of tools exist for transitioning to HP 9000 systems, ranging from editors to re-engineering technologies.

Introducing the HP 9000 into a data center/development environment offers several benefits including cost savings, support for a client/server style of computing, support for standards, and the investment protection that an open system provides.

General Benefits of CASE

CASE itself offers compelling benefits that apply to development and maintenance and across functional areas. These include:

Features	Benefits
Improved quality and maintainability of software.	Lower maintenance costs Better "customer satisfaction."
Improved project team communication.	Ability to manage complex projects/products.
Consistent, repeatable methods and processes.	Process control and ability to maintain software over time.
Improved productivity.	Faster "time to market" and reduced costs.
Increased predictability of projects.	Improved ability to accurately budget and schedule projects.
Clarification of methods and	Ability to rapidly adapt information

processes for enterprise wide consistency.	systems to changes in organization, business focus, customer needs, company size, etc.
Generation of re-usable code for future software.	Reduced cost and faster "time to market" for future development efforts.

CASE has had some shortcomings in its history. Heavy marketing before the technology was mature led to unrealistic expectations and disenchantment. Hard to measure benefits has made the industry wary. Functionality lagging customer requirements and a general reputation for not delivering on its promises have made CASE a four letter word (small joke). But, just as its popularity drops, CASE is getting interesting. Many CASE vendors have now been in the business for 10-20 years, are experienced, and are on their second or third generation of products. Code generation and repository technologies are excitingly emerging as solid technology. New ways of integrating various tools are available and being used. And the development tools of the proprietary world are moving to the open world, supplementing Unix tools. There is also an acceleration of tool availability as tools to address new technologies (like client-server) are becoming available faster and faster.

Specific Benefits of CASE for the Data Center Developer

For the developer who has developed for a mainframe and now needs to develop for open systems in the mainframe "coexistence" scenario, this CASE phenomenon can eliminate the need for retraining. If the tool is adept, the developer need not learn all the intricacies of Unix. He can use a tool he is familiar with from the mainframe environment. As the developer's experience can be leveraged, so can the software that has already been developed for the mainframe. (Through application re-engineering for example.) From a business perspective, developing software on lower cost systems can free up valuable mainframe CPU resources and have a much lower ownership costs overall.

For the developer creating a new application on the HP

9000, this CASE phenomenon can make available integrated CASE tools that are unprecedented in the open systems/Unix world, making him much more productive and allowing him more rigor and control in the development process. Some of the independent tools that are emerging can also offer him added productivity in undertaking specific, complex coding tasks (like developing a standard graphical user interface).

The CASE Solution Portfolio

Not all CASE products do the same thing or even address the same span of activities. It's useful to classify CASE solutions as belonging to one of three types:

* Stand-alone Tools and Shells
* Open Integration
* Integrated CASE (I-CASE)

Classifying CASE Solutions

```
<--------------------- Flexibility in Tool Selection


Standalone Tools      Open              Integrated
and Shells         Integration             CASE


Integration/Structure ------------------------->
```

These types are distinguished by the level of structure offered by the CASE tools and the degree of integration between the tools. By understanding what each type offers, it will be possible to better evaluate a good CASE approach for a specific project.

* Tools and Shells: Independent tools and shells that developers choose and use individually. Includes 3GLs, 4GLs, editors, debuggers, etc.) HP and third parties offer a wide variety of tools and shells. Tools and shells are most appropriate for use by individual developers, or for small scale development efforts. Migration and emulation tools can also be considered standalone tools and shells.

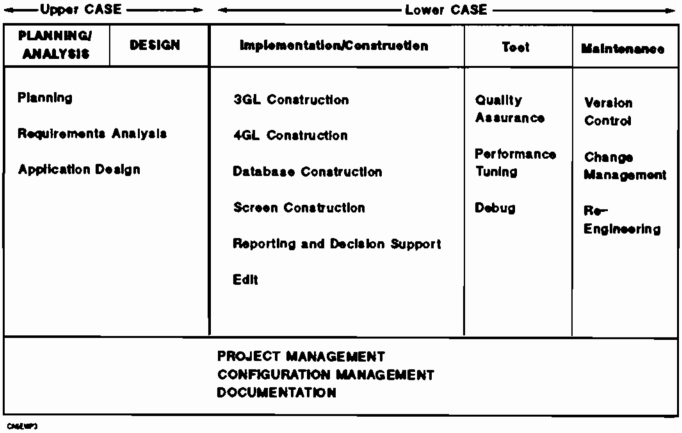
* Open Integration: Tools and shells that are loosely integrated to allow communication between them. An example of an integration framework that provides open integration is HP's SoftBench. Using SoftBench, tools can be encapsulated, giving them a common user interface and message passing capabilities. Open Integration is most appropriate for larger, more complex projects where developers still require a large degree of freedom, but must work together.

* Integrated CASE: Also called I-CASE, provides a comprehensive, structured CASE environment. I-CASE enforces rigorous structures and procedures to successfully complete large-scale, complex development efforts. I-CASE solutions include a data repository.

The activities that are part of the software development process are shown in fig. 3. They have been broken down by phases of the development life-cycle. Standalone CASE tools typically address a single phase or task. Open integration tools allow key standalone tools to be more integrated, pulling the lifecycle closer together. Integrated CASE type of solutions address all the phases and tasks in a single, integrated whole.

Fig. 3.

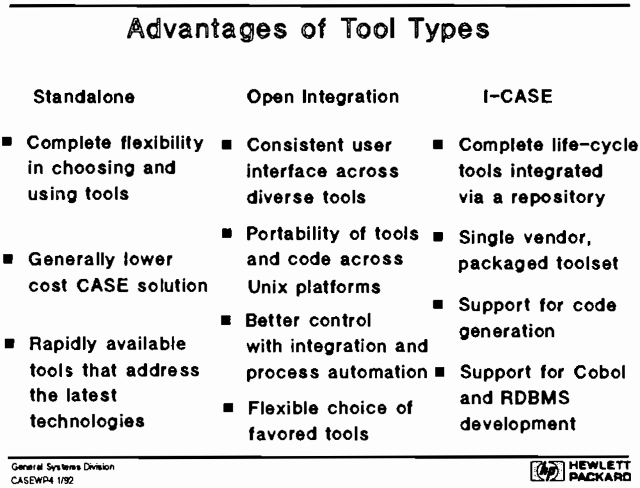
Software Development Activities



← Upper CASE →		← Lower CASE →		
PLANNING/ ANALYSIS	DESIGN	Implementation/Construction	Test	Maintenance
Planning Requirements Analysis Application Design		3GL Construction 4GL Construction Database Construction Screen Construction Reporting and Decision Support Edit	Quality Assurance Performance Tuning Debug	Version Control Change Management Re-Engineering
PROJECT MANAGEMENT CONFIGURATION MANAGEMENT DOCUMENTATION				

The advantages of each kind of tool are listed in figure 4.

Fig. 4



Advantages of Tool Types

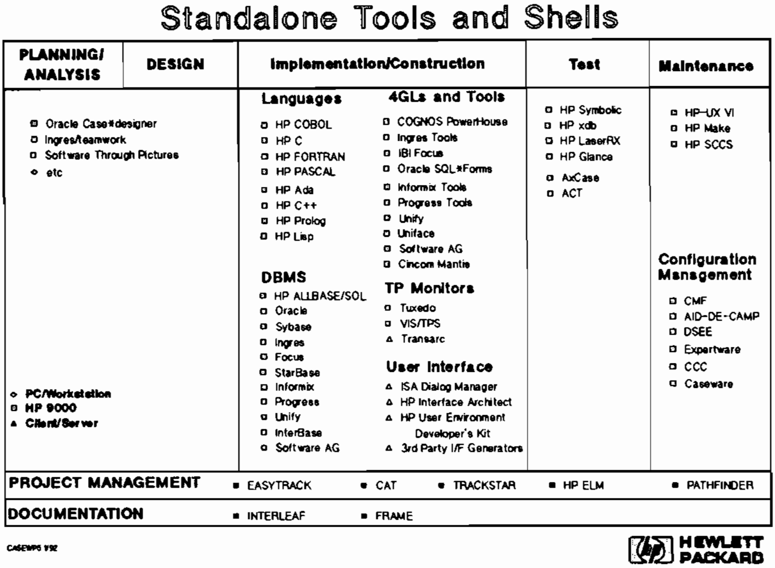
Standalone	Open Integration	I-CASE
■ Complete flexibility in choosing and using tools	■ Consistent user interface across diverse tools	■ Complete life-cycle tools integrated via a repository
■ Generally lower cost CASE solution	■ Portability of tools and code across Unix platforms	■ Single vendor, packaged toolset
■ Rapidly available tools that address the latest technologies	■ Better control with integration and process automation	■ Support for code generation
	■ Flexible choice of favored tools	■ Support for Cobol and RDBMS development

General Systems Division
CASEWP4 1/92

HEWLETT PACKARD

The following figures list examples of tools available for open systems development. An extensive offering of tools and shells is available for the HP 9000, see fig 5. below.

Fig. 5



Standalone Tools and Shells

PLANNING/ ANALYSIS	DESIGN	Implementation/Construction		Test	Maintenance
□ Oracle Case*designer □ Ingres/teamwork □ Software Through Pictures ◇ etc ◇ **PC/Workstation** □ **HP 9000** ▲ **Client/Server**		**Languages** □ HP COBOL □ HP C □ HP FORTRAN □ HP PASCAL □ HP Ada □ HP C++ □ HP Prolog □ HP Lisp **DBMS** □ HP ALLBASE/SOL □ Oracle □ Sybase □ Ingres □ Focus □ StarBase □ Informix □ Progress □ Unify □ InterBase □ Software AG	**4GLs and Tools** □ COGNOS PowerHouse □ Ingres Tools □ IBI Focus □ Oracle SQL*Forms □ Informix Tools □ Progress Tools □ Unify □ Uniface □ Software AG □ Cincom Mantis **TP Monitors** □ Tuxedo □ VIS/TPS △ Transarc **User Interface** △ ISA Dialog Manager △ HP Interface Architect △ HP User Environment Developer's Kit △ 3rd Party I/F Generators	□ HP Symbolic □ HP xdb □ HP LaserRX □ HP Glance □ AxCase □ ACT	□ HP-UX VI □ HP Make □ HP SCCS **Configuration Management** □ CMF □ AID-DE-CAMP □ DSEE □ Expertware □ CCC □ Caseware
PROJECT MANAGEMENT		■ EASYTRACK	■ CAT ■ TRACKSTAR	■ HP ELM	■ PATHFINDER
DOCUMENTATION		■ INTERLEAF	■ FRAME		

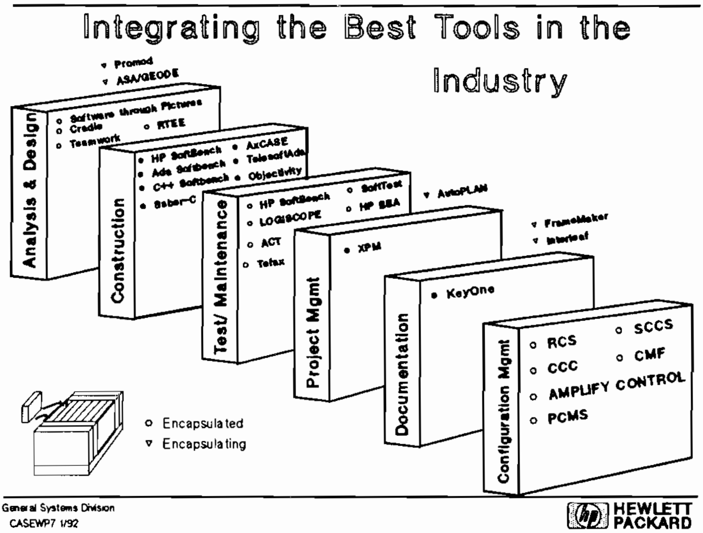
CASEWPS 1/92

HEWLETT PACKARD

Migration and emulation tools can also be categorized as standalone tools and shells. Migration tools may be used for a one time move of software from system to system (mainframe to HP 9000 for example). After the migration, other CASE tools could be used to maintain or modify the application. Emulation tools can be used indefinitely or until an application can be rewritten for the new environment (HP-UX). Ask your HP Sales Rep for a matrix of migration tools and services that are available for the HP 9000.

For the open integration category, the following fig. 7 lists tools that have been encapsulated within HP's SoftBench:

Fig. 7



Integrating the Best Tools in the Industry

▽ Promod
▽ ASA/GEODE

Analysis & Design
- ○ Software through Pictures
- ○ Cradle
- ○ RTEE
- ○ Teamwork

Construction
- ● HP SoftBench
- ● Ada Softbench
- ● C++ Softbench
- ● Saber-C
- ● AxCASE
- ● TeleSoft/Ada
- ● Objectivity

Test/ Maintenance
- ○ HP SoftBench
- ○ LOGISCOPE
- ○ ACT
- ○ Tefax
- ○ SoftTest
- ○ HP SEA

▽ AutoPLAN

Project Mgmt
- ● XPM

▽ FrameMaker
▽ Interleaf

Documentation
- ● KeyOne

Configuration Mgmt
- ○ RCS
- ○ CCC
- ○ AMPLIFY CONTROL
- ○ PCMS
- ○ SCCS
- ○ CMF

○ Encapsulated
▽ Encapsulating

General Systems Division
CASEWP7 1/92

HEWLETT PACKARD

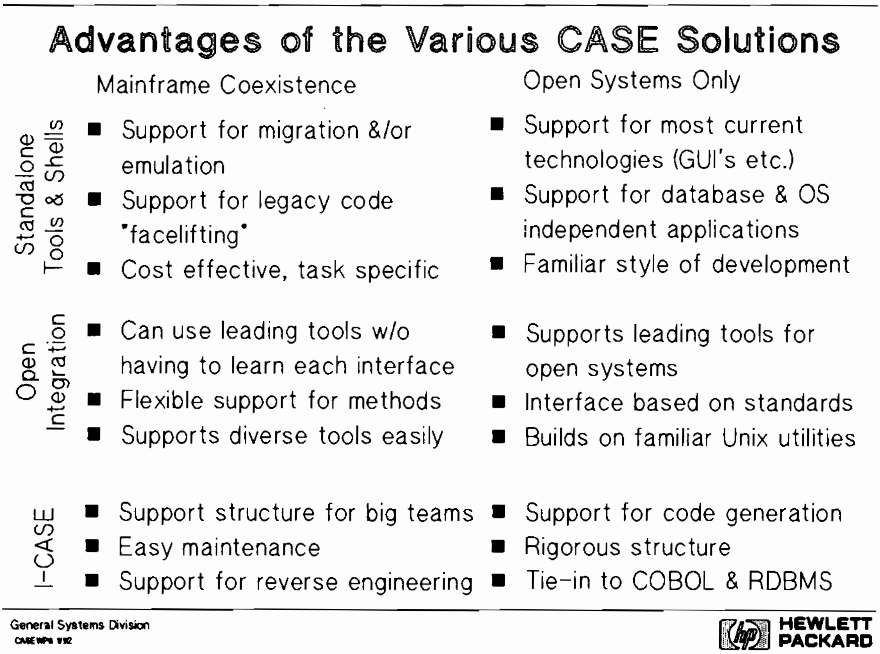
Because it takes advantage of the best Unix tools and of software development features within Unix itself, HP Softbench is especially well suited for open systems development - creating new applications for the HP 9000 and other open systems.

For integrated CASE solutions HP offers the products of three of the top I-CASE vendors in the industry: CGI, Softlab, and Texas Instruments (PacLan/X, Maestro II, and IEF, respectively). These international companies offer well-recognized I-CASE products, previously available only for IBM and proprietary markets. With HP and the HP 9000, they are now into the UNIX market.

Historically, I-CASE products have been available only in commercial mainframe environments (IBM, etc.). Consequently, they are very well suited for developing complex business applications, especially those related to MIS. And often, developers who have worked on a mainframe are familiar with these type of tools. These types of tools are now an excellent option for development activities which need to incorporate both mainframe and open systems (e.g., for the "mainframe coexistence" developer).

A developer working with mainframe coexistence has different needs than a developer creating new open systems applications. What each are apt to like about the various solutions is outlined in fig. 8.

Fig. 8



Advantages of the Various CASE Solutions

	Mainframe Coexistence	Open Systems Only
Standalone Tools & Shells	■ Support for migration &/or emulation ■ Support for legacy code "facelifting" ■ Cost effective, task specific	■ Support for most current technologies (GUI's etc.) ■ Support for database & OS independent applications ■ Familiar style of development
Open Integration	■ Can use leading tools w/o having to learn each interface ■ Flexible support for methods ■ Supports diverse tools easily	■ Supports leading tools for open systems ■ Interface based on standards ■ Builds on familiar Unix utilities
I-CASE	■ Support structure for big teams ■ Easy maintenance ■ Support for reverse engineering	■ Support for code generation ■ Rigorous structure ■ Tie-in to COBOL & RDBMS

General Systems Division
CASE WPs 9/92

HEWLETT PACKARD

Implementing the HP 9000 in a Data Center Environment

The potential for how the HP 9000 could be used in a data center environment was mentioned earlier. Below is a description of how one might go about doing that.

Method	Enabler
- Maintain proprietary application from HP 9000	- Tools, shells, I-CASE
- Emulate proprietary application on HP 9000	- Emulation tools
- Convert proprietary application to native HP-UX code	- Conversion/migration to - Database tools - 4GLs
- Re-engineer application, then redevelop to run native on HP 9000	- Re-engineering tools, I-CASE
- Rewrite proprietary application run on HP 9000	- Tools, shells, I-CASE
- Write new application on HP 9000 to take advantage of open system technology	- Tools, HP Softbench, I-CASE

Maintaining or Modifying Proprietary Applications From the HP 9000

Two situations exist for an HP 9000 system coexisting in a proprietary environment. One is to leave the proprietary application operating as is. Future development can be done on the HP 9000, with the applications accessing data on the legacy system. For example, tools exist that provide DB2 emulation on DOS and Unix as well as providing a linker to access mainframe DB2 data.

A second coexistence situation is using the HP 9000 as a maintenance station for proprietary source code. Various I-CASE products allow that. The HP 9000 can also be used for adding "client/server" to an existing application.

Emulating Proprietary Environments on the HP 9000

HP references several third-party tools for emulating

Is it a project that will be contained in one development group or will it span across groups, divisions, or geographical locations? Is there a need for central control or will developers have autonomy? More structure requirements and more complexity point to an I-CASE product. Conversely, simpler tasks or immediate productivity needs point to standalone tools or open integration products (e.g., HP Softbench).

* Is the emphasis on new code development, code maintenance, or porting existing code?

Since the majority of MIS resources are spent on code maintenance, tools for code development may be a minor consideration in comparison with tools for maintenance. I-CASE helps in both areas, and supports maintenance more than other products do. Because of its repository, an I-CASE product provides one point of change for maintenance procedures as well as integrated configuration management tools for propagating the "maintained" code. A standalone change management/ configuration management tool can be very helpful as a basic maintenance tool. It acts as a software inventory control system for versions and modules of an application. Open integration type of tools are especially popular for new code development (particularly in Unix environments) because they tend to be very strong in the construction phase.

* What is the expected life-span of the software being developed?

This is related to the maintenance question above. If the software is to last a long time, it will have to be maintained in a way that is structured and consistent over time. If the software is just a quick fix, there is no need to use expensive tools (although most developers have realized that there is no such thing as a quick fix).

* Does the organization prefer any particular methodology? Or, for that matter, are they already using one?

Certain tools support specific methodologies, and sometimes, organizing and training a development group around a specific methodology is the most complicated and expensive part of implementing CASE. Select a tool that supports a methodology that you're already using.

* Is there a CASE legacy? What tools does the organization currently use or prefer? Have they created any of their own tools?

Sometimes developers won't care about compatibility with their existing tools, but most probably will. If there are tools that they are already using, or if they've created their own, the open integration approach is strong at integrating diverse tools. Also, some of the I-CASE vendors do work with other individual tools to provide some integration, however, this is rare.

* What is the desired target platform? What hardware, operating system, language, database, TP monitor, etc. do they want the software that they're developing to run in?

The open integration category usually supports C and C++ development for UNIX. The I-CASE products support primarily COBOL development for many platforms. With the I-CASE products, you will need a code generator for each specific environment you want to target. With open integration, it's all done by hand, so no special code generator is needed, only the appropriate compiler.

* What is the desired development platform?

Developers may not have a preference here. They may not realize that they can develop on a platform other than the target system. On the other hand, they may prefer the same platform as their current system (e.g., a PC).

Facilitating Open Systems Application Development and Ensuring CASE Satisfaction

A survey conducted by Sentry Market Research in 1991 demonstrated that sites that are satisfied with CASE spend their money differently than sites who are dissatisfied. Specifically, sites that are satisfied with CASE spend more money on training and consulting, and less money on hardware than sites that are dissatisfied. Also noted in the survey is by spending more money and time for training and consulting services, satisfied sites have improved the likelihood of continued satisfaction and efficient CASE use.

In addition to the toolsets discussed above, vendors offer training and consulting. A sample of specific offerings from HP are:

- Consultation with local HP field representatives and service professionals. They understand emerging technologies and how open systems can be incorporated into your information systems applications.

- Seminars that help you analyze the existing software engineering process and recommend metrics and improvements.

- HP's Open Systems Environment program that offers high level seminars on the benefits of moving to open systems. In the course of the seminar the participants create a "roadmap" for doing this.

- Regional experts on HP 9000 coexistence and offloading in proprietary environments. These consultants are available for ensuring technical design feasibility.

- A variety of classes for developers, systems administrators, etc.

The Human Impact of CASE

Another thing to keep in mind as you evaluate CASE is how a commercial CASE solution will affect the developers. Depending upon how experienced the organization is and how involved with CASE the developers are, implementing CASE (especially I-CASE) can be disruptive. It may involve change, re-organization, re-training, etc. Getting the developers behind implementing CASE, essentially getting their "buy-in," can be the single most critical success factor for a project.

To assist in the transition process, HP and third party CASE vendors offer a variety of services, including training classes and on-site consultants.

Summary

In response to rising application development and maintenance costs, an increasing number of businesses are looking at CASE solutions.

In response to the quest for lower cost mainframe alternatives and the focus on open systems and client/server, an interesting mix of CASE tools has become available. These new CASE tools allow for some innovative ways of introducing open systems into existing information systems environments.

CASE (Computer Aided Software Engineering) is a set of tools and services that support all aspects of application development and maintenance, including analysis and design, implementation, and project management. Such solutions are useful to virtually any organization that develops and maintains applications, and offers real benefits, such as improved software quality and reduced development times.

There are three types of CASE solutions: standalone tools and shells, open integration, and Integrated CASE. Each with their own strengths and appeal, they make difficult tasks realistically achievable.

There are a variety of ways to use CASE tools to introduce open systems into your data center and take advantage of the benefits of lower cost, more flexibility, and better investment protection that open systems afford.

As an open systems provider, HP provides a portfolio of solutions in each tool category to make the HP 9000 a valuable data center asset. HP's focus is on tools to allow development on and development for open systems and heterogeneous environments.

System Management in Distributed Environments

Wolfgang Maier

HEWLETT PACKARD

Hewlett-Packard GmbH
Herrenberger Str. 130
D 7030 Böblingen, Germany
Phone: +49 7031 143041

1 Introduction

The availability of powerful PCs, workstations and minicomputers has changed the computer infrastructure of many companies during the last 5 years. A few years ago, the whole computing environment was concentrated on a few machines in the "glasshouse". Today, the network at most companies consists of different-brand, special-purpose computers (print server, file server, OLTP server, etc.) distributed around several locations.

During this same period end-user computing has become increasingly important [1]. The availability of powerful desktop computers and user-friendly software packages (spreadsheets, graphics, reporting, word-processing, etc.) has lead to a tremendous growth in end-user orientation. This development changed the role of computing dramatically [2]. Today, the desktop computer is used for almost all types of business. Mission- and business-critical applications require a high availability of the computing and network resources.

While there is a continuing strong trend toward decentralization of computing power, distributed client/server environments continue to become more complex and difficult to manage [3]. This increasing complexity conflicts with the requirements facing MIS management; namely, tighter cost control and increasing demands for improved services and productivity.

Centralization and automation are two major approaches which address these conflicting requirements. Instead of using different system consoles at several locations, the entire network is managed from a single, central point. Replicated tasks are automated and the system administrator focuses on pro-active maintenance and escalation management tasks. But, for this methodology to be successful several other aspects (management infrastructure, tools, processes etc.) need to be addressed. This paper describes the evolution of system management in distributed environments.

2 Challenges

Despite the decentralization of services, IS responsibility has remained a central function in most companies [1]. Within a distributed computing environment management and control are the major areas of responsibility for MIS today. Now, organizations try to seize the opportunity to more closely integrate IS and business planning. Assessment and evaluation of effectiveness of this increasingly complex infrastructure becomes a major challenge for MIS management. Even today the majority of MIS managers are faced with this question of effectiveness. What does MIS contribute to the success of the company? What are the metrics for efficiency and success?

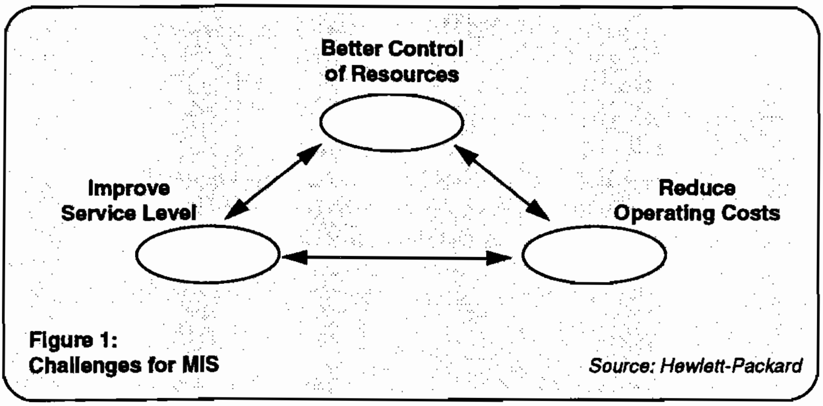


Figure 1:
Challenges for MIS

Source: Hewlett-Packard

These changes within the computing environment has lead to major challenges for MIS (Figure 1). Companies expect solutions from MIS which address these conflicting requirements.

Cost Reduction

In a distributed client/server computing environment, multiple systems from different vendors, positioned in different locations, are networked together. Today, this diversity requires that the administrator use different, inconsistent tools and management schemes [3]. In order to achieve the required level of proficiency necessary to manage such a diverse environment, much time and money has to be spent in training the administrators. With the increasing number of systems the administrative costs grow to a level that most companies find unacceptable.

Service Improvement

Computers are used for more and more tasks by an increasing number of people. This necessitates a very high availability of the systems. Users are directly affected by the quality and reliability of the environment. They expect an improved level of service, and system downtime has to be avoided or minimized [4]. It is no longer acceptable that a reconfiguration requires a system halt, or that applications are interrupted for a database backup.

Resource Control

Today, operators are overloaded with trivial routine tasks and often are confronted with irrelevant system messages. This causes errors in reactions and increases the likelihood of not recognizing important problems [5]. Thus, better control of the computing resources is required, including preventive maintenance and early detection of unusual conditions.

In summary, the increasing complexity of a distributed environment presents MIS with major challenges. There is no way that MIS can proceed as usual in solving these issues.

3 Vision

Changing technology will further improve price/performance ratio of computers, and the distribution of services. These improvements in computers and communications will lead to an open, standards based client/server environment with distributed information and processing.

An integrated management system (Figure 2) simplifies the administration of a distributed environment. The administrator can use the different applications in a consistent way, with the same management scheme, independent of the underlying hardware or operating system. The applications will share information in a common datastore. An integrated inventory system, and accounting and business planning tools are components of the solution. The environment is automated with a central monitoring and control station. A common graphical user interface facilitates the consistent management of resources and applications across distributed multi-vendor computing environments.

The integrated system will provide a consistent management scheme for an administrator. From a central point the operator can monitor and control all networked systems. The management application will follow the workflow of an administrator instead of forcing him to follow a computer internal scheme. The operational steps of the application will be mapped to more abstract human tasks. For example, adding a user today requires multiple steps using several different tools with different interfaces. The user has to be added to a mail system, a database, an operating system, etc. In an integrated management system the function *add user* will be presented as one single task, corresponding to the administrator's logical view of adding a user. Information sharing, the abstraction of tasks, and centralization will allow an implementation of application independent, global policies. These policies can be derived from business goals and will provide an implicit guidance for applications and for operators.

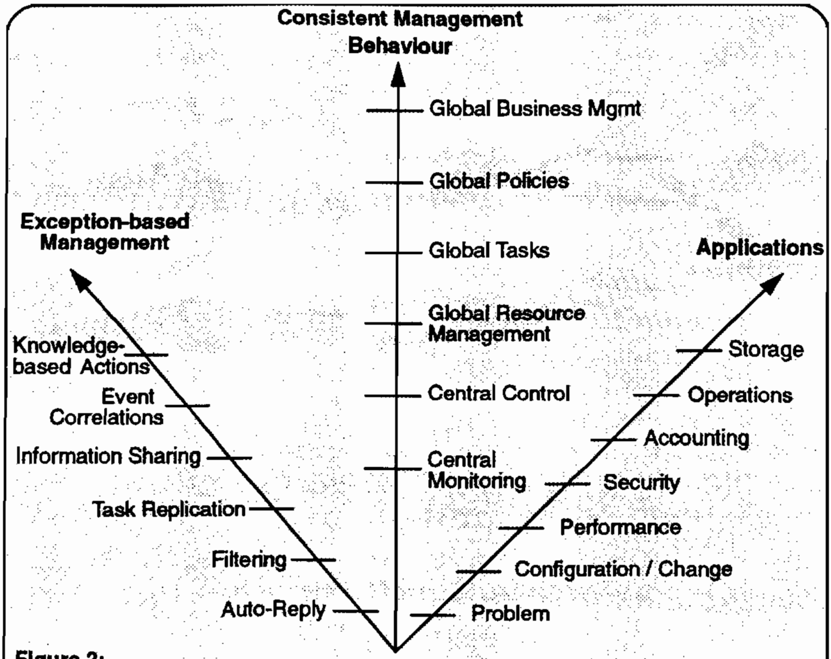


Figure 2:
The Integrated Management System

Source: Hewlett-Packard

The environment will be self-managing, notifying the operator only of exceptions. The system will provide sophisticated help for identifying problem causes, and guidance for problem resolution. Historical analysis of similar problems and correlation between different errors will be used to handle more complex situations. New technologies, like knowledge based systems, will be incorporated to support all those activities. This will lead to a kind of *self learning* system helping the administrator and operator to manage and control the more complex distributed environment.

In combination with the efficient utilization of people and processes, the integrated management system will provide the desired level of service and quality with improved efficiency and reduced costs. Therefore, it addresses each of the conflicting requirements facing MIS today.

4 Requirements

Ease of Use

Today, a high level of skill is required to perform the management tasks of a computing environment. Inconsistent tools, each with different behaviour, and machine oriented cryptic commands are the working environment of an administrator. There is a growing demand to reduce skills, training, time and costs required for the operation. Additionally, end-users should be shielded from system management activities in order to fully concentrate on the business tasks.

Multivendor Support

Standardized hardware platforms and operating system services eliminate the need to rely on just one vendor. In many organizations, there exists already a multivendor hardware and software environment today [6]. Several different and inconsistent tools are used to manage this heterogenous environment. There is a strong need for solutions which serve multiple heterogeneous platforms.

Automation

Automation has proven a valuable means in reducing the human error rate and improving productivity of operators in a mainframe environment [4, 5]. Facing a strong need to reduce costs and improve productivity and services, automation becomes even more important to MIS. Automatic reactions to planned events, scheduling and automatic execution of repetitive tasks, as well as automatic notification of potential and actual problems are some of the important areas for attention today.

Centralization

In many organizations dedicated operators are not available at remote computer sites. With dozens, hundreds, or even thousands of computers in a network, a central point of control is required. From this point, all operator actions can be performed on any system within the network. It would be some kind of a control panel for monitoring, control, and administration of a networked system.

Customization

Every company has its own individual policies and business goals. In order to apply those policies to the operational tasks of their computing environment, the management applications have to be adaptable. Customization of tools, according to specific needs, will provide an important differentiating factor for success of the business. People who know the business best should be able to adapt, tune, and maintain the environment [2]. This knowledge can be leveraged by the company as a whole, through distributing these customized applications throughout the network.

Process Definition

Before routine tasks can be automated, the underlying processes must be analyzed and documented. The different tasks, their commonalities and their relationships have to be identified and understood before processes can be automated [6]. The definition of repeatable processes is an important step in setting up policies, automating tasks, and controlling and evaluating these processes [5]. This leads to a better understanding of the whole business system.

Management Infrastructure

Central management and automation are important parts of the management processes, but they must be supported by an appropriate infrastructure. Such an infrastructure has to serve as a solid foundation unifying system and network management. It has to provide services for the development, use, and management of the heterogeneous distributed computing environment.

5 Current Status and Trends

Few organizations have established a distributed client/server environment. But there continues to be a strong trend toward distributed computing. Today, we are in the midst of a dramatic change. The following sections give an overview of the current situation.

5.1 Automation and Centralization

In the mainframe environment all major software and information technology (IT) suppliers have for several years offered tools for automated operations [7]. These automation tools help operators do their job, and allow a shift away from trivial and repetitive tasks. Recent surveys indicate that automation software packages are now well accepted by operations personnel [8]. However, very few automation tools exist for a distributed environment. Some network management tools provide automatic reactions to specific SNMP (Simple Network Management Protocol) events (e.g., when a threshold is exceeded). Other tools use external hardware to intercept console messages and react under certain conditions. Several vendors are planning to introduce distributed automation tools in the near future.

Centralization is inherent to the mainframe environment. Several central console facilities are available which intercept messages from one or multiple computers in the data center. In a distributed environment, the concept of a physical console as a focal point for the operator no longer exists. Applications don't use the console as a medium for event notification. In this area of central management, again, SNMP-based network management tools are the most advanced today. Within a year central system management solutions are expected.

5.2 System Management Applications

System management applications provide specific services to the user such as backup, performance monitoring, job management, etc. Until recently, these services were tied to a specific computer and its attached devices. Some links between applications running on different systems were provided, such as file transfer (for example in order to transport spooler requests to another machine for remote printing). But the focus of system management applications still was one specific computer system.

The trend toward distributed environments has changed the requirements for system management applications. Applications now have to serve a network of computers and support LAN-attached devices. The user requires services that provide a network-transparent view of the systems, and deal only with logical tasks. For example, the user selects a printer, but is not concerned with which system or LAN the printer is attached. The components of such upcoming applications can be distributed on multiple nodes. This allows the use of dedicated servers for specific purposes (file server, database server, print server, etc.) across a network, with a separation of the user interface from the application service.

Several customers have established a hierarchical structure of their computing environment. Some mainframe-class computers are located in the headquarters, satellite servers are distributed at the branch offices, and dedicated servers with PC and workstation clients serve the actual workgroups. Today there are no system management applications that serve all the platforms across these multiple tiers. This will change in the near future. There will be similar trends as in the area of relational databases. The applications will be available on multiple platforms, provide distributed services across a network, and establish gateways to other applications providing the same service.

5.3 Processes and People

Tools and technology alone do not solve the business problems for MIS. Important improvements can only be achieved by mapping this technology to business processes and continuously tailoring and optimizing them. Process improvements require an understanding of various tasks, their dependencies and relationships. In order to automate these tasks, an organization has to operate with defined procedures and plans and has to control these processes. Otherwise, problems requiring management attention may be forgotten or deferred with an unpredictable impact on the operation.

The Software Engineering Institute (SEI) of Carnegie Mellon University has developed a model for assessing the maturity of an organization's overall process quality. This model is called "Capability Maturity Model" (CMM) [9]. It characterizes the management process using a scale of five maturity levels, starting from level 1 (*Initial,* with inefficient processes and unpredictable results) up to level 5 (*Optimizing,* with well established quantified processes and iterative improvement cycles).

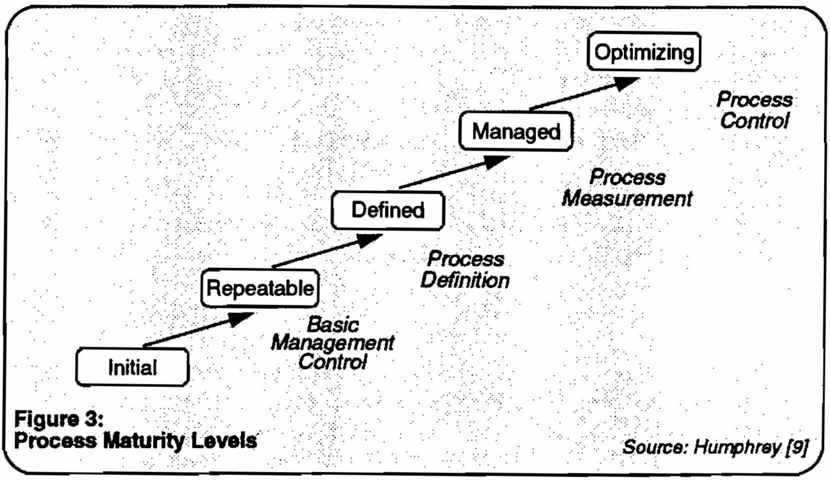


Figure 3:
Process Maturity Levels

Source: Humphrey [9]

A survey of the SEI covering the software development process showed 74% of the companies in level 1, 22% in level 2, 3% in level 3, and none above. According to SEI's estimations it takes about 18-36 months for a company to move to the next higher level of maturity on this scale [10].

People are involved in all processes of information management, such as planning, control, administration and operation. They also play an important role in the evolution toward defined processes and increased level of maturity. Today, most of the value of management and integration comes from people performing the tasks. The processes and knowledge for managing the environment are located in the heads of experts in most cases. On the other hand, human interaction speed, and reaction behaviour are limiting factors to the quality of service provided. Keeping in mind the control of costs and the demand for an improved service level, the solution is to relieve people from standard, repetitive processes as much as possible, and to rely on their expertise for handling of exceptions. This shift has already been completed in several industries facing similar problems (plant automation, process control, etc.).

5.4 Standards

There are a lot of ongoing activities regarding standardization and de facto standards in system management. Several ISO and ANSI standards are already available. IEEE™ and X/Open™ have defined several application programming interfaces (APIs) in the area of operating systems, connectivity, and communication services. Additionally, these bodies have defined generic models to describe the realm, the objects, and their relations. Open Software Foundation

(OSF™) provides a set of services and APIs achieving industry wide acceptance (e.g., OSF/Motif™, OSF/DCE™).

These standardization efforts lead to important improvements in the area of connectivity between dissimilar computers. Additionally, they are a prerequisite for different applications (e.g.,databases, mailsystems, etc.) to communicate and exchange data. Worldwide networks have been established with thousands of computers communicating and exchanging data.

Object Definitions

The key functions of system management applications are to configure, monitor, and maintain the appropriate elements they serve. In the case of a spooler application, a printqueue is created, print requests in the queue are monitored, and the priority of a print request may be changed or the request deleted. Today, every tool has a different set of commands. How does an operator from a central management station change the priority of a print request in a distributed heterogeneous environment? He has to know exactly which spooling tool is running on what computer and how the different tools are handled. Does the spooler application provide support for changing a priority of a print request? Is this attribute called priority, severity, urgency? What are the values of the priority? If it is a scale of 1 to 10, is 10 the highest or the lowest priority? Graphical interfaces provide some help, but they cannot hide different functions and behaviour.

One major step towards the solution of this issue is standardization of object definitions and a common availability of those definitions in a global repository, the Management Information Base (MIB). With this approach, the object *printqueue* is clearly defined. It includes a well known number of attributes (queue name, priority, etc.), their value types and ranges (e.g. priority: low, medium, high), the corresponding functions (e.g. delete, add, change) as well as the error conditions and alarms. If such a MIB with standardized object definitions is available and used by applications, it will be possible to initiate a specific function from a central location in a common way, despite using different tools on multiple platforms.

In the network management area, such a standard MIB definition exists for SNMP devices, the SNMP MIB II. For system management objects no such a standard exists. Several organizations and committees are working on this topic and the first results are expected within one year from IEEE, X/Open, OSF and others.

5.5 Management Infrastructure

Today, many different applications are available to manage and control the network of systems. Some of these individual applications follow a common look and feel, but they all have a different behaviour, use different services, and do not share common data. Using a common management infrastructure, the application services can be components of an integrated information management solution.

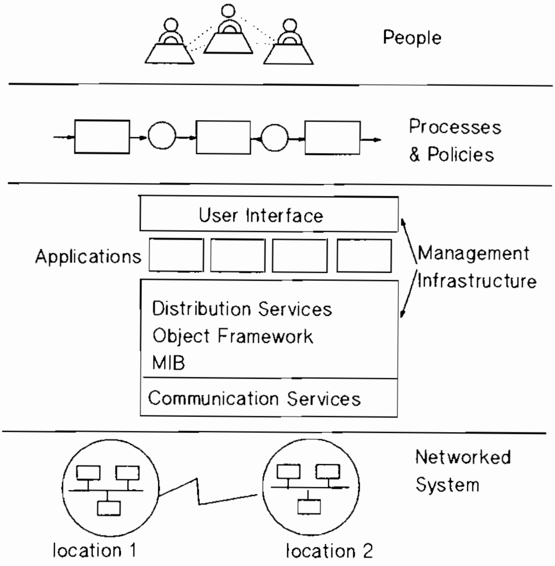


Figure 4:
Information System Model

Source: Hewlett-Packard

All major IT and software vendors have a strategic system management program (HP OpenView™, IBM SystemView™/NetView™, DEC EMA™, CA90s™, etc.). These programs are based on a common management infrastructure. The key elements of such an infrastructure are:

- a common datastore for information sharing
- a common user interface for consistent look & feel, behaviour and ease of use
- process interaction and communication services
- developer interfaces (application programming interfaces)

The vendor specific efforts will provide a solution within the realm of that supplier, but they do not address the issue in an open, heterogeneous world. OSF has addressed this area to provide an industry standard with its Distributed Management Environment (DME™) [3]. This environment includes all services required for such a management infrastructure. The resulting industry standard infrastructure for system and network management can be a solid basis for developing modular applications, where building blocks can be bound together to form an integrated solution.

6 Next Steps

Customers want an integrated management solution serving a distributed client/ server environment. A prerequisite for providing such an integrated management system is the availability of a common, industry wide accepted infrastructure. OSF Distributed Management Environment (DME) will provide such an infrastructure by mid/end of '93. This infrastructure will allow consistent presentation and behaviour as well as information sharing between several different applications. Using this infrastructure, tools and applications can be delivered in a modular and structured way building components that can be bound together to an integrated solution (plug & play principle).

A consistent presentation and behaviour will simplify the management of distributed systems and reduce costs of administration. A common available set of application programming interfaces (API) will increase the portability of the applications between different platforms. This will extend the interoperability of applications utilizing common management services.

Information sharing, using a common datastore helps MIS achieve greater control of the environments and costs. For example, a software distribution application can then store its data (e.g., which and where the software is installed) into the common database. This data can be a base for an inventory system to generate an overview of the computer related assets in the company. Today, both applications don't share data, and the information available from the software distribution package has to be added manually into the inventory system. Additionally, accounting and business planning tools can be built on top of such an infrastructure.

Open Software Foundation (OSF) has selected HP OpenView technologies for key components of the DME submission. This makes HP OpenView a de facto industry standard and the right strategic choice for multivendor system and network management. Today, HP OpenView already provides key elements of DME, including a developer's kit. Thus, HP OpenView products, built on this platform, provide standardized solutions for today's problems and a guaranteed migration path to the upcoming DME environment.

HP OpenView offers a broad portfolio of applications today, and allows for easy integration of management applications. A consistent user interface and the capability for central monitoring and control are steps towards a consistent management behaviour. Automated reactions and filtering of events are important first steps towards exception based management. Task oriented management stations will allow multiple operators and administrators to deal with specific tasks and applications, like performance monitoring, databases, networks, etc.

Hewlett Packard, together with its partners, is working on an integrated information management system, as described in section 3. The integration of different applications toward a consistent, and exception based management solution will be addressed in phases. Several parts are available today, including the HP OpenView infrastructure providing major parts of DME. HP provides a solution for today's problems and a guaranteed migration path to DME for system management solutions required by tomorrow's complex distributed computing environment.

7 Summary

Today, companies are using information technology for almost all types of business. The availability of powerful desktop and mini computers leads to decentralization of computing resources. The resulting growth in the number of computer systems and networks from multiple vendors increases the complexity of the environment and its operating expenses. This leads to major challenges for MIS as the responsible entity for the management of this environment. MIS managers are faced with questions of effectiveness, the requirements for cost reduction, and demands for improving the service level.

While looking for tools to manage the distributed client/server environment, many companies recognize that only an integrated solution can protect their investment and satisfy their requirements. An integrated management system based on a standardized infrastructure, with portable applications, sharing information based on standardized objects is the solution for the future. The environment will be automated with exception based operations and provide a consistent management scheme. HP OpenView has been confirmed by industry analysts as a leading platform today for integrated management [11, 12]. With the selection of core components of HP OpenView by OSF for the upcoming industry standard DME, HP OpenView is the superior strategic choice to manage multivendor distributed environments.

Acknowledgements

Acknowledgements to Kai-Uwe Jensen, Tjerk Feenstra, Ramesh Panchal, Larry Wilkinson, and Dieter Wagner for contributing ideas and comments to this paper.

Trademarks

CA90s is a registered trademark of Computer Associates International, Incorporated.

EMA is a registered trademark of Digital Equipment Corporation.

HP OpenView is a registered trademark of Hewlett-Packard Company.

IEEE is a registered trademark of The Institute of Electrical and Electronics Engineers, Incorporated.

NetView and SystemView are registered trademarks of International Business Machines Corporation.

OSF, Motif, DCE and DME are registered trademarks of the Open Software Foundation.

X/Open is a registered trademark of X/Open Company Limited.

References

[1] Clark Thomas D. Jr., *"Corporate Systems Management: An Overview and Research Perspective,"* Communications of the ACM, 2/92 Vol. 35, No. 2.

[2] Palermo Ann, *"Workgroup Technology: Tying Technology to Business Objectives,"* IDC White Paper, Computerworld March 23, 1992.

[3] Open Software Foundation, "OSF Distributed Management Environment Rationale," OSF September, 1991.

[4] Sentry Market Research, *"Automated Operations in the Data Center 1990"*

[5] Schulman J., *"Automated Systems Operations,"* Gartner Group, Inc., Software Management Strategies, June 5, 1989.

[6] Hayes Frank, *"Exploding Desktop Myths,"* UNIXWORLD 4/92, p38-44.

[7] Picardi Dr. Anthony C., *"1991 Data Center Software: User Requirements And Market Size"*, International Data Corporation, May 91.

[8] Holdsworth Ian, "Computers go it alone," Financial Times March 10, 1992.

[9] Humphrey, Watts S., *"Managing the Software Process,"* SEI Series in Software Engineering, Addison-Wesley, 8/90.

[10] Thaller, Georg Erwin, *"US-Uni Carnegie Mellon stellt die Entwicklung auf den Prüfstand"*, Computerwoche, 3/13/92.

[11] McCusker Tom, *"HP Opens Up Network Management,"* DATAMATION, February 15,1992, p22-26.

[12] Slater Derek, *"Buyers' Scorecard: LAN management software,"* Computerworld January 27, 1992, p96-97.

PAPER #2034
HOW TO MAXIMIZE UP TIME IN A UNIX ENVIRONMENT

CECILIA MARGEVICIUS
HEWLETT PACKARD
19490 Homestead Road, MS 41AK
Cupertino California 95014
408-447-7929

ABSTRACT

Maximizing computing capabilities is not only an expectation, it has also become a vehicle for impacting a business's competitiveness and revenue. The loss of computing capabilities can impact production, critical decision making, customer satisfaction and even human life. Increasing the availability within a computing environment involves solutions that not only minimize planned downtime, but also minimize unplanned downtime. Particularly as distributed computing and Open Systems Foundation Distributed Computing Environment (OSF DCE) have become a driving force, the various options for hardware and software resiliency become more critical for computing environments. This paper describes the various levels of availability solutions within the UNIX environment.

INTRODUCTION

The demand for highly available systems and data is rapidly increasing as businesses are basing their operation and competitiveness on mission critical applications - applications that are integral in the day to day operation of an organization. Because of market pressures and user expectations, the need for higher availability is greater than ever before. The loss of computing capabilities can impact production, revenue, critical decision making, and customer satisfaction.

HIGH AVAILABILITY REQUIREMENTS

Availability has become a requirement in the industry due to the major shift from batch to on-line computing. There is an increased recognition of the cost of downtime as it relates to an enterprise's competitive position, business opportunities, and financial bottom line. Mission critical environments cannot tolerate data inaccessibility or service interruptions. An outage translates directly into reduced productivity, financial loss, lack of a competitive offering and lack of service. In order to maximize the uptime of your system, understanding the underlying causes of system unavailability allows you to plan for and implement your environment to accommodate your availability requirements. There are several key requirements which today's computing environments must incorporate to address the availability needs for an organization's business. The three most important are:

1) data availability and integrity
2) minimal planned downtime
3) minimal unplanned downtime

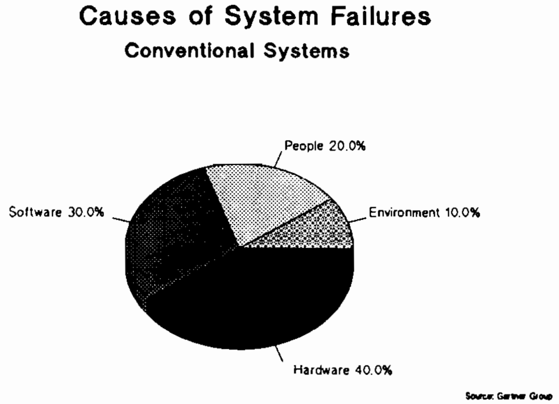
How a system performs on each of these metrics determines how well the system can meet availabilty needs.

The first requirement, data availability and integrity, is the most critical as it refers to the accuracy, correctness, consistency and validity of the data. In environments where computer systems are constantly being relied upon to provide information for making critical business decisions, data can never be permanently lost. Once the data is guaranteed intact, the second and third requirements together determine the system's availability. Planned downtime is the time a system is unavailable for predetermined periods to perform tasks like system maintenance, software or hardware upgrades and disk backups. Unplanned downtime is the time a system is unavailable due to unanticipated events like hardware failures, operating system software failures, application software failures, environmental situations and operator errors. In general, the computing resources must be designed so that neither planned nor unplanned downtime will negatively impact essential business operations.

High availability requires systems designed to tolerate unexpected conditions: to detect a fault, report it, and then continue service while the faulty component is repaired off-line. A highly available system must also tolerate interruptions of services caused by power outages, software updates and operation errors.

WHAT CAUSES DOWNTIME?

In developing a strategy to address availability, it is useful to understand why systems fail. Figure 1 illustrates the results of a Gartner Group study on downtime for conventional systems.


Causes of System Failures

Conventional Systems

Source: Gartner Group

As shown in the figure, downtime is affected by four major areas: hardware, software, people and environment. This emphasizes the point that when considering the total availability of a system, there is more to consider than just the hardware's availability. The remainder of this paper focuses on how current and emerging technologies can minimize the impact of failures in the four areas.

MINIMIZING THE IMPACT OF HARDWARE FAILURES

There are various methods for minimizing or eliminating the impact of hardware failures and thereby increasing hardware availability:

(1) the use of hardware that is less prone to failure
(2) replication of hardware

Using Hardware Less Prone to Failure

The computing industry has implemented several strategies for increasing the hardware availability. One key strategy is to reduce the number of components that make up computer hardware. VLSI (Very Large Scale Integration) permits hundreds of thousands of components to be integrated on a singe chip, which dramatically increases the reliability due to the reduced number of parts that can fail. Technology is moving more towards single chip implementations to further decrease the probability for error with decreased number of parts. Also, processors provide error correcting codes to detect and isolate errors and subsequently recover and restore information that would have been lost before. This is done through redundant encoding of information. In addition, comprehensive integrated hardware testing strategies account for a major component of improved reliability. Extensive hardware testing improves infant mortality detection and provides a more comprehensive coverage of system related hardware problems.

To increase the reliability of system memory, techniques such as memory scrubbing and dynamic memory deallocation are being utilized. Memory scrubbing is a process which works in the background to correct single bit memory errors before causing a failure. Dynamic memory deallocation is the deconfiguration of sections of memory which have experienced repetitive single bit errors.

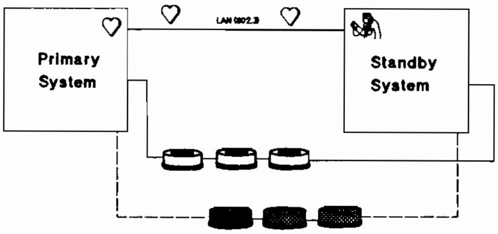
Tools also exist from vendors which provide a proactive approach to hardware problems. Proactive tools detect and resolve hardware problems before they result in unplanned downtime through the utilization of automatic analysis software.

As increasing numbers of disk drives are being connected to computer systems to accomodate the steadily growing storage requirements, disk drive failures and resultant system halts cannot be tolerated within mission critical envionments. Failures within the disk subsystem can generally impact operations for hours, therefore including drives which have a low mean time between failure and have a well acknowledged quality history often can accommodate most users need for reliability.

Replication of Hardware
Spare modules can be installed or configured in advance, so when one module fails, the second can replace it almost instantly. The failed module can be repaired off-line while the system continues to deliver service

Multiprocessing (MP) systems can be designed for concurrent processor maintenance. This allows you to repair a processor while the rest of the system continues to handle its workload. If the system detects a failure of one of the CPU boards, it brings itself down and then automatically reboots with the faulty CPU deconfigured. Since MP systems have additional hardware cards, the reliability typically is less than for a uniprocessor system. As the number of components increases, so does the probability for failure. MP systems are generally solutions to accommodate higher performance requirements. System interruptions can be further reduced with solutions such as graceful degradation of performance and on-line replacement of failed cards. These can effectively address the reliability of MP systems.

When a system fails, it is critical to resume operations as quick as possible to minimize the impact of unavailability. Mission critical operations, in particular, require near continuous operation. Solutions which combine software with multiple systems exist to monitor the state of the mission critical systems. The monitoring of the system is implemented through state of health messages which are transmitted across the network. In the event the monitoring system does not receive a message within a predefined period of time, the monitoring system automatically takes over the operations of the failed system.



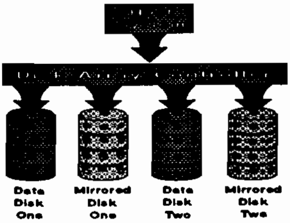
There is a brief period of time in which the processes are interrupted and data uncommitted to disk is lost after a failure has occurred. Implementations vary from hot standby systems which monitor a mission critical system to systems which add the workload of the failed system on the remaining system. The key attributes to consider in a processor recovery product is the automation, the transparency to the user and applications, data integrity and the duration of the recovery.

Environments in which any inability to access data would have an impact on operations, solutions such as disk arrays and disk mirroring exist.

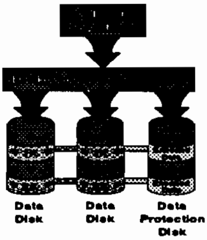
Disk arrays offer a data protection option which guards against potential loss due to a drive failure. This is accomplished by dedicating an entire disk drive or sections of all drives to storage of encoded information. This allows the array controller to recreate data from a faulty drive with no loss in performance. Other arrays duplicate or mirror the disk drives to provide two copies of the data in case one copy is lost.

Disk arrays are available in several modes. The variation in mode has an impact on performance and also the level of availability. There are several levels of arrays available RAID Level One, Three and Five.

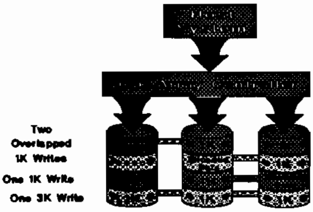
Disk Arrays - RAID 1



Disk Arrays - RAID 3



Disk Arrays - RAID 5



Implementations of RAID (Redundant Arrays of Inexpensive Disks) Level One are designed to provide a high data availability disk array solution through disk mirroring. Disk mirroring is where a duplicate copy of the data in one disk drive is also stored on another disk drive. In the event of a disk drive failure, the array controller will automatically switch all system I/O activity to the surviving drive.

RAID Level Three is designed to provide a high transfer rate, high availability disk array solution. It uses a separate data protection disk drive to store encoded data. The data protection drive stores an encoded form of the data from the data disks to allow reconstruction of the data in the event of a failure. Should a data disk fail, the array controller reconstructs missing information from the failed drive using the encoded form of the data. The failed disk drive can be replaced on-line and the array controller will automatically rebuild the data. Within commercial computing environments, RAID Level 3 is typically the optimal solution to address both availability and performance.

RAID Level Five is designed to provide a high transfer rate, moderate I/O rate, high availability disk array solution. It executes reads and writes to the disk drives either in parallel or independently of each other, depending on the size of the transfer from the host system and the block depth specified by the array controller. Meaning concurrent reads and writes can occur for short transfers, but large transfers still require all drives. The data is spread across all disk drives in the array, but the encoded data protection is not stored on a single dedicated drive. In the event of a disk drive failure, the missing data is recalculated from the coded or checksum information.

Disk Arrays

- Low Cost Data Protection
- Data Protection disk
- Online "Hot" disk drive replacement and disk rebuilds (based on vendor)
- Selectable operating modes
- Application Transparency

Disk Mirroring

- No single point of failure
  - separate controllers
  - separate power supplies
- On-Line Backup (based on vendor)
- Selective mirror of data (based on vendor)
- Application Transparency

Software disk mirroring is another option to reduce or eliminate the impact of a disk or interface card failure. Disk mirroring creates a pair or triplet of disk sections which are identical copies of each other. The disk sections are normally on separate disks, but appear as a single section to applications.

Mirroring disks provides two distinct benefits:

1) In environments where no data can afford to be lost due to disk failure, a mirrored disk increases data availability and integrity. Even if one disk fails, the application can still access and store data on the remaining disk(s) without interruption.

2) Disk backup, repair and replacement can be done to one disk without interrupting applications. When a disk is removed from a mirrored pair to perform one of these tasks, the other disk is still available to applications.

MINIMIZING THE IMPACT OF SOFTWARE FAILURES

Software reliability depends on whether or not the software contains failure producing faults or the use made of the software causes it to encounter those faults. In other words, does the code contain bugs or does invalid or unexpected input to the code cause a failure to occur?

A key indicator of reliability for individual Unix systems is the longevity of the specific Unix port. Some of Unix's poor reputation for reliability arises from the fact that many systems in the Unix market are relatively young ports and each needs to be broken-in over time. A key reason for this is that generic Unix is coded to go into "panic conditions" (to decide to crash) more often than necessary, even when alternative resolutions are provided and/or the operating system is tailored closely to individual hardware platforms.

Also, Unix has had to be configured by a system administrator who knows his users' needs for system resources. System administrators can control resources such as disk space utilization by using disk quotas. This provides a mechanism for controlling and reporting user utilization of disk space. Soft (quota) and hard (limit) bounds may be set by the system administrator on the number of files and blocks an individual user may utilize on each file system.

Testing is critical for reliable Unix systems. All vendors check for standards compliance (ex. SVVS), but these do not adequately check reliability. The utilization of reliability tests which include one-to-one, one-to-many, many-to-many combinations of subsystems, various resource usages for system components, multi-node tests using different combinations and load levels on each node are required to effectively test the system. Typically, system reliability testing is measured in Continuous Hours of

Operation (CHO) with various stress and volumes imposed on the system. Testing needs to be done for system and component limits, resource exhaustion and multiple simultaneous operations. Some vendors utilize a FURPS (Functionality, Usability, Reliability, Performance, Supportability) policy as a criteria for releasing new versions of Unix. Based on the defined FURPS, the code must be certified based on defined Breadth, Depth, Reliability and Defect Density. Breadth is the percentage of internal and external entry points/interfaces and parameters executed/used. Depth is the percentage of all branches/instructions executed. Reliability is the continuous hours of component stress testing passed. Defect density is the maximum number of known open defects.

The system can be designed so that each software module either operates correctly or stops immediately (fail-fast). To make a module fail-fast, there are at least two techniques: self-checking or comparison. Self-checking is where a program does simple sanity checks on inputs, outputs, and data structures. If an inconsistency is discovered, an exception is raised and either the state is failed or repaired. Comparison or sometimes referred to as N-Version Programming (NVP) is where two or more modules of different design run the same computation. A comparator examines their results and declares a fault if the outputs are not identical.

Another approach is to design into the software alternate ways of achieving the program results when faced with failures or other unexpected difficulties. For example, when data may not be accessible from a primary storage location, check an alternate location. Another may be when data cannot be transmitted over a primary network path, the utilization of an alternate network would allow the application to remain operational. The key to software resiliency is building in techniques for dealing with all types of errors. Unfortunately, the major stumbling block in this approach is recognizing all the types of errors or failures so that the design can accommodate them.

The utilization of transactional locking and logging techniques, such as a journaled file system, aids in quick recovery, enhanced performance and data integrity. With a journaled file system every system call that modifies data does so as a transaction. The data is locked as it is referenced and the changes are recorded in a disk log before allowing the data to be written to the disk location. In the case of a system failure, the data is restored to a consistent state by applying the changes contained in the log.

Another very powerful and emerging tool is a transaction monitor. Particularly with the proliferation of distributed computing resources, the need for transaction processing is fundamental to computing because it provides data integrity in the face of concurrency and system failures. By distributing computing resources, an enterprise achieves greater configuration and increased flexibility in applications and data placement.

Technology such as Transarc's Encina transaction monitor plays a key role in the Distributed Computing Environment (DCE). Encina's Transaction Processing (TP) strategy is to provide a set of standards-based distribution services for simplifying the construction of reliable, distributed systems, and to provide the integrity guarantees required for mission critical enterprise computing. The strategy expands on the DCE framework to include servies which support distributed transaction processing with full data integrity, security, and recoverability.

In distributed transaction processing, two or more databases might be involved in a global transaction. In a global transaction, multiple sub-transactions, executing on multiple systems within a network, can collectively be treated as a single coordinated transaction. Such an environment requires data integrity be maintained from the multiple, coordinated transactions. The commitment protocol, called two-phase commit, is the synchronization method to guarantee that multiple, related transactions are either committed or rolled back depending on wether they are successful. Encina's TP services offer two-phase commit through the X/OPEN XA interface.

These and more standards-based technologies are emerging to provide greater availability of distributed data and applications. Also, teçhnologies such as a distributed file system, naming services, and network management tools will allow identification, isolation, and repair of transient and permanent distributed system faults.

MINIMIZING THE IMPACT OF HUMAN ERROR

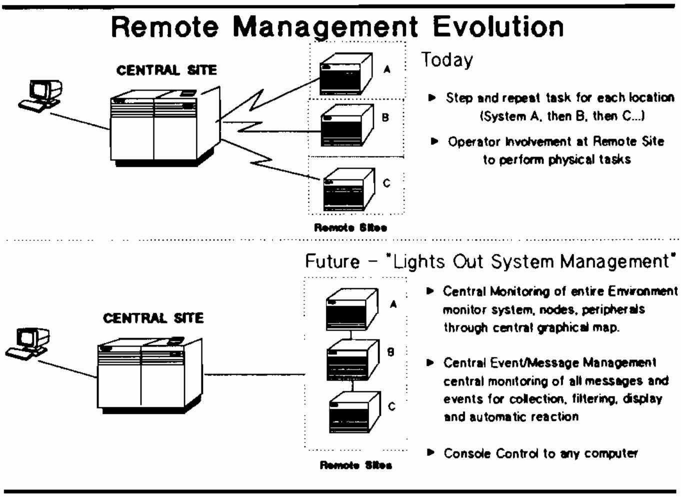
Since operator error account for much of the system outage, taking steps to minimize operator intervention can significantly reduce both planned and unplanned downtime. Vendors, such as HP are shipping systems as an integrated package which includes interface cards, the operating system and the networking software pre-installed. This type of solution both provides for quick and easy installation as well as ensures accurate initial configuration. Also, features such as autoconfiguration reduces the operator involvement required to configure devices during system startup.

Administrators are involved in the installation and update of software. Typically, the task is a planned activity which may result in downtime. Minimizing the duration of the downtime can be accomplished through software distribution tools which perform installation and update of system and application software. An administrator can simultaneously distribute software to multiple nodes from a single point, eliminating the need for tedious step-and-repeat operations which are prone to error. The ability to perform upgrades on-line or temporarily remove the system from the network minimizes or eliminates the downtime. Solutions such as the Software Distribution Utilities from OSF will aid in minimizing operator involvement and facilitate the move toward 'light-out' computing.

Sites are evolving to "lights out" configurations. "Lights out" Data Centers run companies with little or no human intervention. The Boole and Babbage "Getting Started in Automating Computer Operations" report states that "lights-out operations is a concept and aspiration for most progressive data centers today. The source of the idea for lights-out operations was inspired by telecommunications. Fifty years ago, a single human being handled a switch board... now the entire process has been automated... The same evolution is occurring now in computer operations."

The Gartner Group has stated that "automated operations will be an inevitable path for large shops by the 1990s. Many dark computer rooms are in operation today. Segregating the systems in dark computer rooms can reduce operating expenses while making more efficient use of computer room space. Equipment can be run at stable, cool temperatures, which provides more reliability and cost-effectiveness."

Over time, remote operations will be reduced, and eventually eliminated. In some large installations today, robotics eliminates many of the manual procedures.



Remote Management Evolution

Today

- Step and repeat task for each location (System A, then B, then C...)
- Operator Involvement at Remote Site to perform physical tasks

Future - "Lights Out System Management"

- Central Monitoring of entire Environment monitor system, nodes, peripherals through central graphical map.
- Central Event/Message Management central monitoring of all messages and events for collection, filtering, display and automatic reaction
- Console Control to any computer

Systems and network management tools are evolving to provide solutions for automation and efficient utilization of resources within distributed environments. Backup utilities, such as HP Omniback, provides centralized, unattended network backup which utilizes sophisticated scheduling and journaling to automate and track backup. Tools to manage the performance of a distributed environment are evolving to provide "management by exception" techniques. The objective is to alert administrators of situations which may lead to problems or failures. System defined rules are utilized so that once thresholds are exceeded either an automatic action is taken or an administrator is alerted.

Rapid detection and resolution of network error conditions is an important capability which any high availability system must have. Network management products exist which provide the ability to configure, troubleshoot and monitor the performance of the network. Automating the management of the network and including facilities which automatically alert you of problems or automatically react based on defined thresholds can significantly enhance the system availability.

Infonetics, a market research firm, calculates that for every hour a LAN is inoperable, it costs a Fortune 1000 company more than $30,000 in lost productivity. As standards emerge to facilitate expansion and to support multivendor environments, systems become increasingly difficult to manage. Networks are problematic because any change in the environment can severely affect performance. As long as network managers operate well within a given envelope of tolerance, everything will seem fine. But once the network crosses a certain threshold, things deteriorate very quickly. Tools exist to isolate and kill network bugs using high level protocol analyzers and instruments to monitor performance across enterprise-wide networks and recognize potential problems before they become failures.

OSF's Distributed Management Environment (DME) will provide technologies for efficient management of distributed environments. Included in DME are Network Management tools, such as HP OpenView, which provides an easy-to-use set of tools for verifying network configuration, isolating faults, and tuning performance of TCP/IP networks. DME also includes tools for print management, Software Distribution Utilities (SDU), software licencing tools (NetLS) plus enabling technologies for object and event services, protocols and application services.

MINIMIZING THE IMPACT OF ENVIRONMENTAL SITUATIONS

Unplanned events such as power outages, earthquakes, tornadoes, and fires clearly cannot be eliminated, however the impact on operations can be minimized.

Volatile storage, typically RAM, is where portions of objects reside when they are being accessed. The contents of volatile storage are lost if the system crashes or if power is lost. To address unforeseen circumstances such as power outages, some systems, like those available from Hewlett Packard, provide Powerfail/Battery Backup within the processor. Powerfail/Battery Backup gives you the capability to recover quickly and transparently from power failures and brownouts. When power to a system is cut off, volatile memory is maintained by batteries. If power is restored during a period of time, the system resumes exactly where it left off when the power outage occurred.

Uninterruptible Power Supplies (UPS) also protect the hardware and data from power outages. UPS is a separate power system which provides continuous availability of the system when blackouts (total loss of utility power), brownouts (short term decreases in voltage levels), power surges (short term increase in voltage) or power spikes (dramatic increase in voltage) occur.

Services are also available to provide system service when disasters such as earthquakes, hurricanes, tornadoes, etc occur. The services vary based on the urgency to resume critical operations and also the associated cost for a disaster recovery plan. There are 'hot-site' services which provide a fully configured backup data center that enables you to get critical computer operations up and running within a matter of hours after a disaster. A 'cold-site' service provides a building wired for computers; users bring their own equipment which typically results in a few days before critical applications are back in operation.

The emergence of distributed computing technolgy will provide tools to design replicated data sites. Using transaction management (Transarc's Encina), distributed file systems, and network management tools, solutions will evolve to provide highly available distributed

PAPER # 2035

Comparing Multiprocessing Architectures

systems. Data will be replicated over wide area networks, allowing operations to continue nearly uninterrupted from various locations within a global network.

SUMMARY

Computer reliability and availability has and will continue to evolve to meet the needs of the computing environment. Minimizing planned and unplanned downtime in a UNIX environment is the same as a non-UNIX environment. Software errors are reduced through increased software quality, human interaction is reduced through methods which automate systems management, hardware reliability is increasing with new design techniques to minimize the number of components and work around typical errors, and facilities are available to plan for disaster recovery. Also with the emergence of DCE technology, distributed high availability solutions will facilitate increased availabiltiy.

let users reach high performance levels without waiting but will also have the power needed for the most demanding environments.

Definition of Multiprocessing

Multiprocessing is a design where many CPUs are connected to provide additional computing power to allow many tasks to run in parallel. This is more than multitasking. Multitasking systems allow many jobs or processes to be running on a system, rather than running each job serially to completion. However, that does not mean that at any one moment that all jobs are running. For example, figure 2 shows how a single CPU system parses out CPU time to multiple jobs and switches between running jobs. If the CPU switches between jobs quickly, it gives a user the appearance that many jobs are indeed running simultaneously, even though in actuality only one job is running at a time. For multiprocessing systems, figure 2 shows that each CPU continues to switch between running jobs, but now jobs are truly running in parallel with jobs running on each CPU (of course each individual CPU continues to multitask).

Multiprocessing also means more than having a multiuser or timeshare system. As with multitasking, a system with a single CPU can switch between jobs for different users and, if it has the performance to switch quickly enough, gives each user the appearance that they have a dedicated computer at the other end of their display. A multiprocessor system essentially provides more CPU resources that can thus be shared simultaneously by many users without a drop in response time.

Another common misconception with MP systems is that they are Fault Tolerant systems. While Fault Tolerant systems do have multiple CPUs as figure 3 shows for the HP 9000 Series 1200 systems, they also have fully redundant hardware for system buses, memory, I/O and peripherals and power supply. This redundancy protects Fault Tolerant systems from unplanned downtime due to any one hardware component going down. However, MP systems can take advantage of the additional CPUs to ensure continued operation if one CPU fails. For example, if a CPU failure in the HP 9000 Series 890S MP Corporate Business Server system causes the system to go down, the HP 890S will bring itself down and then

HP 9000 Series 1200 Fault Tolerant System

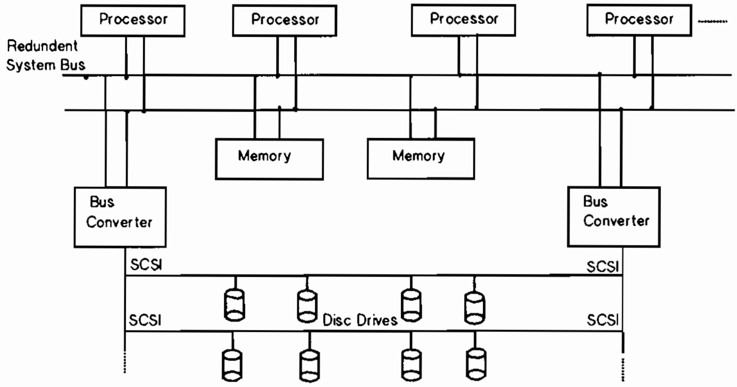


Figure 3: HP 9000 Series 1200 System diagram

Figure 3

boot up again with the faulty CPU reconfigured out of the system. The HP 890S will continue to run at a reduced performance level until the CPU board can be repaired or replaced.

Building an MP system requires more than just adding additional CPU boards to a system. In actuality, there is a continuum of multiprocessing systems from loosely coupled machines on a network to multiple CPUs in one system acting as if there was only one CPU in the machine. The key measure of MP systems is how transparent the implementation is to users and programmers and also how well the additional power of the extra CPUs is utilized.

Types of Multiprocessing Systems

Several terms have been suggested to help describe different types of computer systems and provide a useful framework for understanding multiprocessing systems in relation to other types of systems. Computers can be grouped into 3 classes:

SISD (single instruction, single data stream). A SISD machine is a typical single CPU system where one CPU executes one instruction and works on one piece of data at any one time.

SIMD (single instruction, multiple data stream). One example of a SIMD machine is a vector processor, where an array of data is input into several registers and then the same operation (eg. multiply by pi) is performed on all the data simultaneously. SIMD machines are mostly used for scientific and engineering applications where intense numeric computations and simulations are needed.

MIMD (multiple instructions, multiple data stream). This is where many CPUs execute different instructions on different pieces of data. However, MIMD systems differ on how instructions and data are shared between each CPU and how much each CPU interacts with the other CPUs.

Multiprocessing computer systems are MIMD machines with the CPUs in one system sharing all the system resources such as memory, I/O and buses. There are two main types of multiprocessing systems in use today, asymmetric and symmetric.

Figure 4 shows an asymmetric multiprocessing system found in several graphic workstations. Even though there are two processors in the system, they are not equal. The main processor runs the operating system and user programs, accesses the main memory, and also controls the actions of the attached graphics processor. The attached graphics processor handles the displaying and updating of the graphical display, but does not actually modify any of the data in the main memory nor does it execute any user programs. The attached processor does improve the performance of the overall system, because the main processor does not need to spend its resources on the graphical display, but it does not provide a user with the full power of the two processors.

Asymmetric multiprocessing systems are relatively easy to implement because less modifications need to be done to an operating system, memory bus structures and I/O. However, asymmetric systems have a potential performance bottleneck with the main processor. Figure

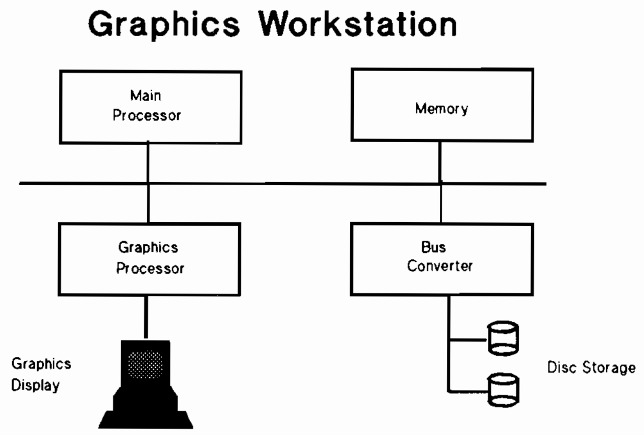

Graphics Workstation

Figure 4: A graphics workstation is an example of asymmetric multiprocessing with specialized processors

Figure 4

5 shows an expanded asymmetric system where there is also an attached vector processor and floating point processor, in addition to a graphics processor. The three attached processors are controlled by the main processor and can thus sit idle if the main processor becomes busy. Asymmetric processors are best suited for special tasks such as heavy numeric calculations or simulations where the system can be tailored for the specific task at hand.

In symmetric multiprocessing, all processors are equal and are not specialized for specific tasks. Figure 6 shows the HP 890S symmetric multiprocessing system where each processor can access memory, I/O and other parts of the system. The operating system and hardware components need to maintain data consistency, job scheduling and message passing between processors. There are two main implementations of symmetric multiprocessing: master/slave and fully symmetric. While the hardware may be symmetric, the differences between these two implementations are mainly in how the

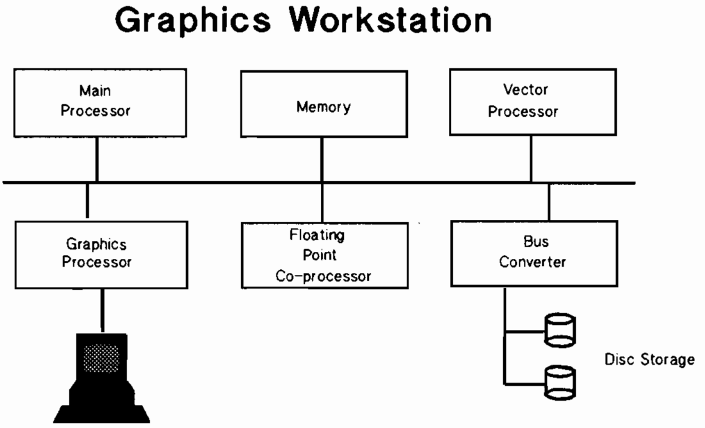


Figure 5: Attached processors in asymmetric systems may not be fully utilized due to specialization and to contention for the main processor.

Figure 5

operating system handles two main tasks: job scheduling and kernel access.

The job scheduling algorithm determines how jobs are parsed out and scheduled to each processor and is a major factor in overall multiprocessor performance. Figure 7 shows the job scheduling for master/slave systems, where one processor (master) assigns jobs to the other processors (slaves). In fully symmetric systems, each processor receives the next job that is waiting in a global system queue.

To show how the two scheduling routines work, think of waiting for the next available cashier at a store. In a master/slave store, one of the cashiers has the responsibility of assigning customers to the next available cashier. If the master cashier is busy when one of the slave cashiers signals that they are ready for another customer, there will be a delay before a customer is sent to the slave cashier, slowing down the overall throughput. Or this store might modify the

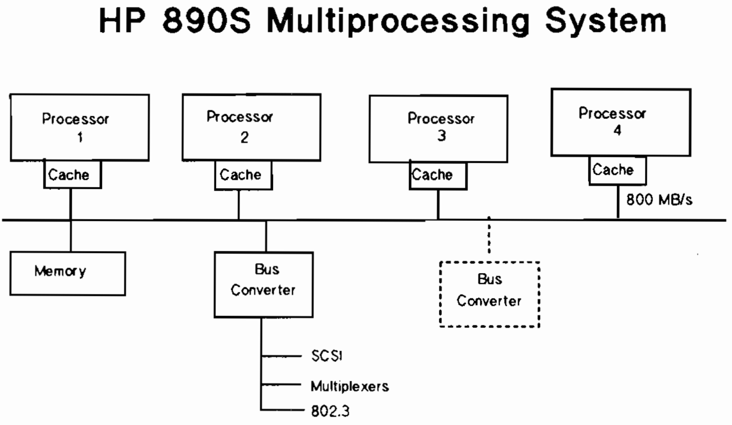


Figure 6: The HP 890S is a fully symmetric multiprocessing system with all processors equal to each other, and can access all parts of the system.

Figure 6

routine slightly and have the master cashier assign an incoming customer straight away to one cashier. However, if for some reason there is a holdup at one of the cashiers (because of some pricing difficulties), then all the other customers in line for that cashier are stuck, even though other cashier lines continue to move quickly. Either way, overall throughput is lowered as the master cashier (processor) must spend time assigning and possibly reassigning customers (jobs) to the other slave cashiers (processors). Also, individual customers (jobs) may find that they may have to wait much longer than other customers before they are serviced by a cashier (processor), just because they were assigned to a slow line to begin with.

In contrast, in a store that implements a fully symmetric system, no one cashier assigns customers to the other cashiers. All incoming customers line up in one queue and when they get to the head of the line, they go immediately to the next available cashier. The advantage is that no extra time nor resources are spent allocating customers (jobs) to specific cashiers

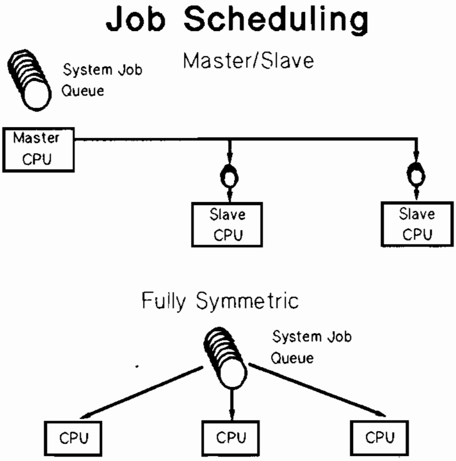


Figure 7: Master/Slave MP systems depend on the Master CPU to schedule all jobs. Fully symmetric MP systems all CPUs dynamically access the system job queue, leading to improved load balancing.

Figure 7

(processors) and that any queued customer (job) is more likely to be serviced in a timely fashion.

Kernel access, or the lack of access, can also hinder multiprocessor performance. As figure 8 shows, some implementations of master/slave systems limit the execution of the UNIX kernel (the operating system core) and access to I/O to only the master processor, further pinching the master processor bottleneck.

Examples of master/slave multiprocessing can be seen in systems from SUN and Solbourne. These type of systems are best suited to environments that have little I/O activity or have intensive numerical tasks that can be easily scheduled. In a heavy multiuser environment, users of master/slave symmetric multiprocessing systems (while gaining some improvement) will fail to see the extra performance promised from additional processors as the extra processors will spend more and more time waiting for kernel or I/O requests to be filled by the master processor.

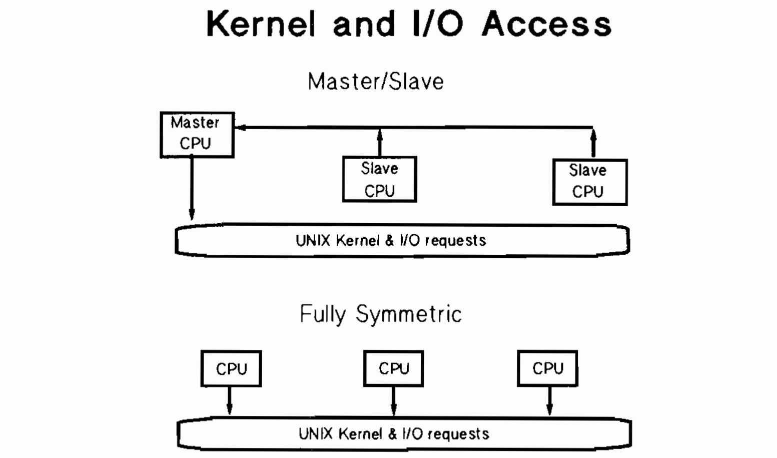


Figure 8: Slave CPUs in Master/Slave systems must wait for the master CPU to service all kernel and I/O request. CPUs in fully symmetric systems all have full access to the Kernel and I/O

Figure 8

Fully symmetric multiprocessing systems, such as the HP 890S, are more difficult to implement, but are better suited to heavy multiuser or I/O intensive environments, such as OLTP and RDBMS applications. Each processor is an equal to other processors and so can access I/O or the UNIX kernel, instead of sending all such requests to a master processor and then waiting for service. In general, SMP systems provide better scaling for commercial environments, such as Online Transaction Processing (OLTP) applications. Mixed workloads would also work best on a SMP system, because they would be most likely to have a high amount of I/O and kernel calls.

Application Fit with Multiprocessing

Applications that appear to the UNIX system as just one large process would most likely not run faster on a multiprocessing system. Such applications are called single threaded and typically can not be split up among multiple processors. Large batch jobs, common in commercial environments, is one example of a single threaded application. In order to be split up over multiple processors, a single threaded application needs to be either "broken up" explicitly into multiple threads or processes, or implicitly broken up with the help of special parallel compilers. However, general purpose parallel compilers are still not available for the commercial marketplace. For today, large batch applications would run best on SMP systems where each individual processor has high performance, rather than a SMP system using many lower performance processors.

The balance between batch jobs and mixed applications (such as OLTP) is one of the reasons that the HP 890S multiprocessing system was implemented with high performance uniprocessors, rather than with several more lower performance uniprocessors. The HP 890S offers the fastest combined batch and OLTP system performance due to its fast individual processors.

RDBMS applications can also be tuned explicitly for MP systems for increase OLTP performance. HP has and continues to work closely with key industry leading RDBMS solution providers to optimize their applications specifically for the HP 890S MP system.

Multiprocessing Challenges

A fully symmetric multiprocessing system needs to overcome several hurdles without compromising the performance potential of additional processors. The main challenges are ensuring data integrity, job scheduling, I/O and kernel access, and application transparency. The fully symmetric multiprocessing implementation of the HP 890S will be used as an example of how these multiprocessing design challenges have been meet.

Ensuring Data Integrity

As seen in figure 6, the HP 890S is a tightly coupled system with all processors sharing the same memory. It is very likely that different jobs running on different processors will need to access the same memory locations, and thus the system must ensure that data integrity is maintained at all times with little or no contention. The HP 890S solves this problem in hardware with a "snoopy" cache protocol. All processors have their own local cache and memory controllers, and manipulate data in memory through their caches. Processors can either share cache lines, or mark lines as private. Each memory controller listens to every transaction on the System Memory Bus (SMB) (or "snoops," leading to the snoopy term) and act as needed to maintain data integrity.

For example, if processor A and B share a cache line, but now processor A needs to write to the shared line, A broadcasts the write and marks the cache line as private. In the meantime, processor B picks up the write request and thus marks its copy of the cache line as invalid, leaving the modified line on processor A as the only valid copy. The HP 890S also avoids another potential bottleneck due to the extra traffic on the SMB. The SMB has a bandwidth of 800 MB/s, ensuring that bus contention will not occur between the processors and thus will not decrease performance.

Job Scheduling

A fully symmetric MP system has already been shown in figure 7 to use a single job queue for all processors. The next waiting job is sent to the next available processor. While a single job queue is an efficient MP scheduling method in general because of its inherent dynamic load balancing, it can be made more efficient by recognizing that not all jobs finish after one run on a processor. Often jobs or processes are scheduled to run for some length of time and are then swapped out and placed back in the job queue while another job is swapped in. However, a process builds up local data structures on the processor that it runs on (such as data into cache). If that process is later scheduled to run on another processor, then the second processor must spend time rebuilding the same data structures that are resident on the first processor. The HP 890S removes this potential source of inefficiency by using a heuristic scheduling algorithm where each process develops an affinity to one particular processor, but can move to another processor if needed for load balancing. To understand this heuristic scheduling method, imagine a process is handled the same way as a person who constantly checks in and out of a hotel. If the hotel kept checking this person into the same room, then some of the person's baggage could remain in the room when the person wasn't in (assuming of course that everyone else has the courtesy to leave it alone), making it very easy and quick for this person (and the hotel) to check back in. If it was necessary to move to another room, or if this person was not coming back, only then would the effort be spent to pack up all this person's belongings and move them.

Kernel Access

In order to maximize performance, the HP 890S allows multiple processes to be executed in the UNIX kernel at the same time. Without this capability, it is very likely that processors could sit idle while they wait for access to the kernel. The HP-UX kernel data structures are broken up into several pieces with synchronization variables called semaphores used to protect each block of data structures. A processor must lock the appropriate semaphore before they can access the desired kernel section. If another processor previously locked the semaphore, then the requesting processor must wait until the semaphore is unlocked. Kernel contention is reduced by having each semaphore lock only a small part of the kernel. The kernel semaphores also ensure that no I/O collisions occur between different processors. This is in contrast to master/slave systems where the UNIX kernel only runs on the master processor, thus there is no need to add semaphores to the kernel. This is one reason why master/slave systems are easier to implement.

Application Transparency

Multiprocessor systems should ideally be transparent to applications, meaning that an application can run on single processor or multiprocessor systems without any modification. For most applications, this will be the case for the HP 890S. The HP-UX system call interfaces for the multiprocessor systems are the same as for a single processor system. Applications that are structured as a set of cooperating processes may experience some problems in a multiprocessor system because processes may not execute in the same order as in a single processor system. These applications should be tested to identify any potential timing problems.

Future Multiprocessing Trends

Many system vendors will continue to refine their multiprocessing offerings. Several vendors such as HP will incorporate advanced technology for improved multiprocessor performance.

HP will continue to enhance its multiprocessing offering. Presently HP offers a 4-way multiprocessor system with the HP 890S Corporate Business Server. The HP 890S will be increased into an 8-way and then later a 16-way multiprocessing system. In order to support these higher levels of multiprocessing, HP will break up the HP-UX kernel into smaller pieces to allow for increased simultaneous access by many more CPUs. Each kernel piece and data structure will continue to be protected with semaphores.

HP-UX will also support multiple threads in the kernel. Threads are special light-weight UNIX processes. Traditional UNIX processes (called a task) can not be broken down into smaller units. If a UNIX system needs to start a new task, it does so with some overhead. With threads, a new process can be started with a thread with less overhead (by doing things such as sharing the same address space and common memory), leading to increased performance. Also, tasks can be broken down into several related threads, leading to a higher level of granularity. Having this increased granularity will lead to improved multiprocessor performance because of parallelization inherent with threads. HP-UX will also implement a communication mechanism called ports as a way for threads to know about each other, to talk to one another and to synchronize related threads. Support for threads will be added first for user applications and then for the HP-UX kernel. With these enhancements, the HP 890S will continue to offer significant performance increases over the next few years.

Massively Parallel Systems

Much attention has been given to research on massively parallel multiprocessor systems. Figure 9 shows that while the hardware implementation is straight forward and the potential performance gains quite large, the software needed to split up processes, coordinate them and maintain data consistency is not straight forward.

SMP systems like the HP 890S are extensions of single processor systems with a common bus to easily share and maintain consistency of a common memory. An SMP software environment is very similar, if not identical, to a single processor software environment, with the benefit of existing applications running without modification. In contrast, applications will still have to be written specifically for massively parallel systems. Applications such as complex numeric simulations (eg. airflow analysis for the space shuttle) that can easily be broken down into individual and well defined subprocesses will continue to be the best fit for massively parallel systems. For commercial applications, special parallel compilers and development tools that would automatically break up applications while maintaining data consistency still need to be developed. While progress is being made in this area, general purpose parallel compilers are still estimated to be years away.

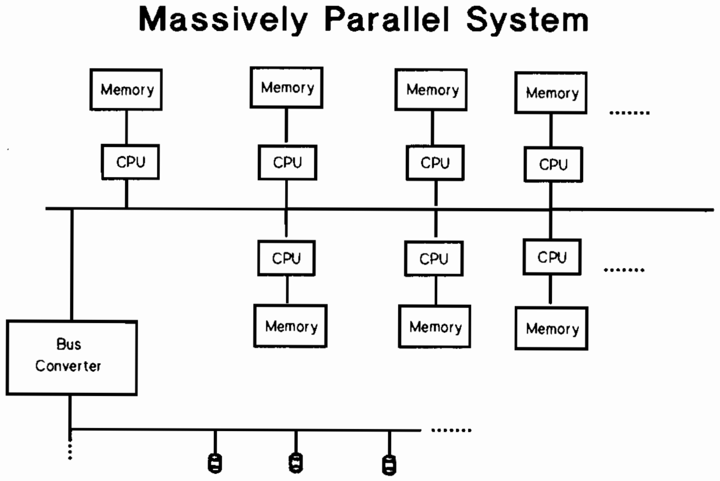


Figure 9: Some massively parallel systems consist of several CPUs each with their private memory, connected by a high speed bus. Special parallel compilers must be available before general purpose applications will run on these systems

Figure 9

Distributed Computing

In many respects, there are similarities between massively parallel multiprocessing systems and computer environments made up of multiple computers connected by a high-speed network, often called a multicomputer. Figure 10 shows that a multicomputer consists of distributed systems (with their own private memories) connected together, executing processes in parallel, and using messages to synchronize and maintain data consistency. In order to provide a standards-based framework for multicomputers (or distributed computing), the Open Software Foundation (OSF) has developed the Distributed Computer Environment (DCE) specification. DCE provides a common framework for sending process requests over a network and also a distributed file system that can be shared by many computers. With DCE, a large application can be written into several parts that can then be parsed out to several computers on the network using Remote Procedure Calls (RPC). Each computer works in parallel on its

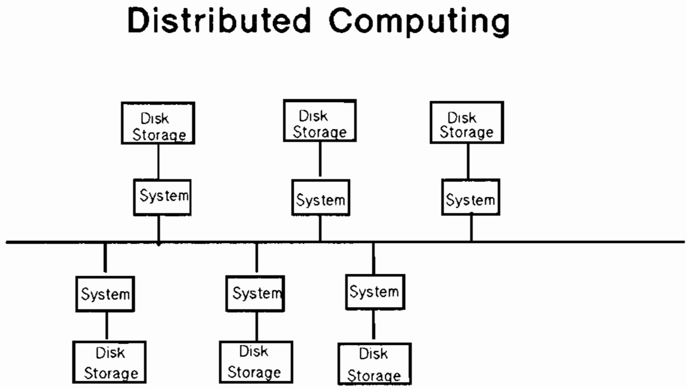


Figure 10: Multicomputers, or distributed systems, are built from general purpose Computers connected by network, Technologies such as DSF's DCE provide a standards-based framework for building distributed applications.

Figure 10

portion, with the result that the overall application executes much faster than it would on one individual system. Multicomputer systems have the advantage of using general purpose computers which are typically very cost effective and can continue to execute non-distributed applications as well as distributed ones. Of course, a SMP system could be one of the systems in a multicomputer environment.

Summary

Multiprocessing systems provide the extra levels of performance needed for demanding environments. While there are several types of multiprocessing implementations, fully symmetric multiprocessing systems offer the most balanced performance for commercial and mixed workload environments. SMP systems most fully utilize the extra performance available with additional processors. Systems such as the HP 890S provide SMP transparently to existing applications that were originally written for single processor systems. Standards development groups such as OSF will provide operating system technologies that will improve multiprocessing performance. While massively parallel systems will continue to be advanced, SMP systems with multicomputer networks will remain cost effective solutions for meeting general purpose computing needs.

PAPER #2036

"WHERE ARE YOUR HP-UX SYSTEM RESOURCES GOING?"

Marty Poniatowski
Hewlett-Packard Company
Stamford, CT
(203) 325-5613

"Where are Your HP-UX System Resources Going?"

Where Are Your HP-UX Resources Going?

A Way To Gain Insight Into The Way System Resources Are Used

With a finite amount of networking, I/O, memory, and CPU resources available on your HP-UX system you should have a good idea how these resources are being used. You should also be able to do this **quickly** and **easily**. This paper covers the way you can gain this insight into your system resources without spending a lot of time doing so.

I'll take a three step approach to determining how your system resources are being used:

1. Some standard HP-UX commands that give you information about system resources.
2. Analyzing accounting files that are automatically produced for you daily and monthly
3. Using the performance monitoring tool **HP GlancePlus/UX.**

The following three sections give details on these three steps.

#1 STANDARD HP-UX COMMANDS

To begin with, let's look at some commands you can issue from the HP-UX prompt to give you some information about your system. The commands I'll cover are:

iostat **ps**
vmstat **sar**
netstat **timex**
landiag

I/O and CPU Statistics with "iostat"

The **iostat** command gives you an indication of the level of effort the CPU is putting into I/O, and the amount of I/O taking place among your disks and terminals. The following figure shows the **iostat -t** command and associated outputs from an HP-UX system. The "#" shown is the HP-UX prompt:

```
# iostat -t
tty       cpu            /dev/dsk/0s0      /dev/dsk/5s0      /dev/dsk/cdrom
tin tout  us  ni  sy  id  bps  sps  msps   bps  sps  msps   bps  sps  msps
0   99    35  2   5   58   1   0.1  19.3    1   0.9  14.9    0   0.0  215.8
```

Here are descriptions of the reports you will receive with **iostat** for terminals, the CPU, and mounted file systems:

1. For every **terminal** you have connected (tty) you will see a "tin" and "tout" which represent the number of characters read from your terminal and the number of characters written to your terminal respectively. The "-t" option produces this terminal report.

2. For your **CPU** you will have the percentage of time spent in user mode ("us"), the percentage of time spent running user processors at a low priority called nice ("ni"), the percentage of time spent in system mode ("sy"), and the percentage of time the CPU is idle ("id").

3. For every **locally mounted file system** you will receive information on the kilobytes transferred per second ("bps"), number of seeks per second ("sps") and milliseconds per average seek ("msps"). For disks that are nfs mounted or disks on client nodes of your server you will not receive a report. Only locally mounted file systems appear with **iostat.**

When viewing the output of **iostat** there are some parameters to take note of.

Firstly, take a look at the amount of time your CPU is spending in the four categories shown. I have worked on systems that have poor performance that the administrator assumed to be a result of a slow CPU when in fact the "id" number is very high indicating the CPU is in fact idle most of the time. If the CPU is mostly idle the chances are the bottleneck is not the CPU but I/O, memory, or networking. If the CPU is indeed busy most of the time ("id" is very low) then see if any process are running "nice" (check the "ni" number). It may be that there are some background processes that consume a lot of CPU time that can in the future be run "nice".

Secondly, compare the milliseconds per average seek "msps" for all of the disks you have mounted. If you have three identical disks mounted yet the "msps" for one of the disks is substantially higher than another then you may be overworking one of your disks while the others remain mostly idle. If this is the case then you may want to more evenly distribute the workload among your disks so that you get as close to the same number of accesses per disk as possible. Note that a slower disk will always have a higher "msps" than a faster disk so put your most often accessed information on your faster disks. The "msps" for a disk maybe 10 or 15 milliseconds and 200 milliseconds for a mounted CD ROM. In the example above the CD ROM is the last disk shown with a high msps.

<u>Virtual Memory Statistics with "vmstat"</u>

vmstat provides virtual memory statistics. There is information provided on the status of processes, virtual memory, paging activity, faults, and the breakdown of the percentage of CPU time, all over the last five seconds. The following diagram shows the output of **vmstat 5 2:**

```
# vmstat 5 2:

procs   memory      page                    faults      cpu
r b w   avm  free   re at pi po fr de sr   in sy cs   us sy id
0 0 0   503  177    0  0  0  0  0  0  3    25 80 18   28  8 64
0 0 0   655  158    0  0  0  0  0  0  4    46 60 19   25 10 65
```

You will get more out of the **vmstat** command than you want. Here is a brief description of the categories of information produced by **vmstat:**

1. You will see processes described as falling into one of three categories; runnable "r", blocked on I/O or short term resources "b", or swapped "w".

2. Next you will see information about memory. "avm" is the number of virtual memory pages owned by processes that have run within the last twenty seconds. If this number is roughly the size of physical memory minus your kernel then you are near paging. The "fre" column indicates the number of pages on the system's free list. It doesn't mean the process is done running and these pages won't ever need to be accessed again, it just means they have not be accessed recently and have been marked free. I suggest you ignore this column.

3. Next is paging activity. Only the first field is useful.

   "re" - Pages that were reclaimed. These pages made it to the free list but were later referenced and had to be salvaged.

4. Next comes the number of faults in three categories; interrupts per second which usually come from hardware ("in"), system calls per second ("sy"), and context switches per second ("cs").

5. The final output shown is CPU usage percentage for user ("us"), system ("sy"), and idle ("id"). This is not as complete as the **iostat** output in that nice entries are not shown as with **iostat.**

As with **iostat**, **vmstat** produces a lot of data which is, for the most part, not useful. There is, however, some useful information from **vmstat** including the following three numbers I always look at.

You want to verify the runnable processes ("r") is higher than the blocked ("b") and runnable but swapped ("w") processes. If two many processes are blocked and swapped your users will get a slower response time.

Next you want verify that memory pages owned ("avm") is less than total memory minus your kernel. If memory pages owned exceeds this value you'll be doing a lot of swapping.

Total Memory - [(avm)(8k) + (kernel size)] = Free Memory

An example of this for a 24 MB system with an 8 MB kernel would be:
24 MB - [(748)(8k) + (8 MB)] = 14 MB of Free Memory

Finally, check to see that "re" is a low number. If you are reclaiming pages which were thought to be free by the system then you are wasting valuable time salvaging these.

Network Statistics With "netstat"

netstat provides information related to network statistics. Since network band width has as much to do with performance as the CPU and memory in some networks, you want to get an idea of the level of network traffic you have. The following diagram shows the output of **netstat -i**:

```
netstat -i

Name  Mtu  Network    Address          Ipkts   Ierrs  Opkts   Oerrs  Coll
lan0  1497 151.150    a4410.esr.hp.com 242194  120    107607  0      18126
```

netstat doesn't provide as much extraneous information as **iostat** and **vmstat**. Put another way, most of what you get from "netstat" is useful. Here is a description of the nine fields in the **netstat** figure:

1. The first field gives you the Name of your network interface (Name), in this case "lan0".

2. This is the "maximum transmission unit" (MTU) which is the maximum packet size sent by the interface card.

3. This is the network address (Network) of the LAN to which the interface card is connected (151.150).

4. Even though this field is labeled "Address" you will see the host name of your system here. This is the symbolic name of your system as it appears in the /etc/hosts file.

Start of statistical information:

5. This field shows the number of packets received (Ipkts) by the interface card, in this case lan0.

6. This is the number of errors detected (Ierrs) on incoming packets by the interface card.

7. This is the number of packets transmitted (Opkts) by the interface card

8. This is the number of errors detected (Oerrs) during the transmission of packets by the interface card.

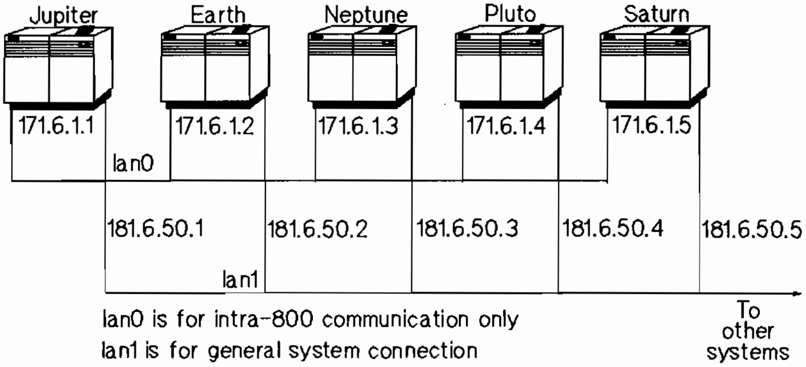
9. The final field is the number of collisions (Collis) that resulted from packet traffic.

Since **netstat** provides cumulative data since the node was last powered up you have a long elapsed time over which data was accumulated. You can also specify an interval to report statistics, however, I prefer to use the statistics accumulated since the system was last powered up.

Using **netstat** is intuitive. You want to verify that the collisions (Coll) is low in comparison to the total number of packets received (Ipkts) and transmitted (Opkts). Every collision your lan interface encounters slows down the network. You will get varying opinions on what is too many collisions, however, if your collisions are less than five percent of "Ipkts" and "Opkts" combined then you're in great shape. If this number is high then you may want to consider segmenting your network in some way such as installing a bridge between portions of the network that don't share a lot of data.

As a rule-of-thumb if you reduce the number of packets you are receiving and transmitting ("Ipkts" and "Opkts") then you will have less overall network traffic and fewer collisions. Keep this in mind as you plan your network or upgrades to your systems. You may want to have two lan cards in systems that are in constant communication so that these systems have a "private" lan over which to communicate and therefore do not adversely effect the performance of other systems on the network.

One technique that is widely used to reduce lan traffic is to configure systems with two lan cards. This has been done by Gartner Group (Stamford, CT) on their Series 800 systems as shown in the following diagram.



Network Statistics With "landiag"

landiag provides additional information related to network statistics. When you run **landiag** a menu appears that gives you the option to perform various functions, one of which is display information related to the lan interface. The following figure shows the output of **landiag** when this option is selected.

```
                    LAN INTERFACE STATUS DISPLAY
                      Tues, Mar 17, 1992  15:47:20

Device file                           = /dev/lan
Select code                           = 21
Current sate                          = active
LAN Interface address, hex            = 0x080009118D57
Number of multicast addresses         = 2
Frames received                       = 12868474
Frames transmitted                    = 107641
Undelivered received frames           = 0
Untransmitted frames                  = 1
CRC errors received                   = 125
Transmit collisions                   = 18129
One transmit collision                = 10777
More transmit collisions              = 7352
Excess retries                        = 0
Deferred transmissions                = 10256
Carrier lost when transmitting        = 0
No heartbeat after transmission       = 0
Frame alignment errors                = 117
Late transmit collisions              = 1
Frames lost                           = 17
Unknown protocol                      = 3167
Bad control field                     = 0

LAN Interface test mode. LAN Interface device file = /dev/lan

clear          =  Clear statistics registers
display        =  Display LAN intfc status and registers
end            =  End LAN interface diagnostic
menu           =  Display this menu
name           =  Name of LAN Interface device file
quit           =  Terminate diagnostic, return to shell
reset          =  Reset LAN Interface to execute self-test

Enter Command:
```

The first observation you probably made when you looked at this figure is that **landiag** displays much of the same information as **netstat** such as the amount of information transmitted and received. **landiag**, however, provides two categories of information that are more detailed than **netstat**:

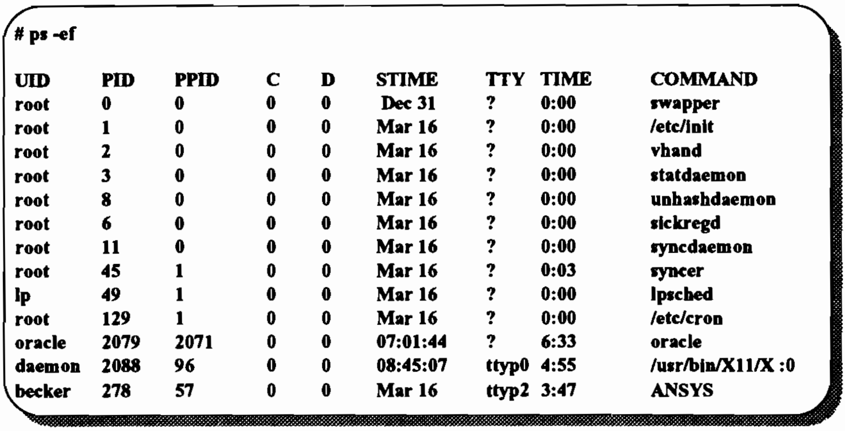
1. There is more detailed information related to the lan interface itself. The **Device file** as it exists in the HP-UX file system is shown, the **Select code**, the **Current state** of the interface which is active or inactive (which is a quick way to see if your interface is up and running), and the **Lan interface address, hex** which you would need to know to create a client kernel or code software.

2. There is also much more detailed error information. Although any error slows down your network having more detailed information on the type of errors and collisions may be helpful in troubleshooting a problem. If, for instance, you have a great deal of **One transmit collision** your network is not necessarily overworked because there will always be collisions

on a network. If, however, you encounter many **Excess retries** you may indeed have an overworked network or defective lan interface.

Check Processes With "ps"

The answer to "What is my system doing?" is **ps -ef**. This command provides information about every running process on your system. If, for instance, you wanted to know if nfs is running you would simply type **ps -ef** and look for nfs daemons. Although **ps** will tell you every process that is running on your system, it doesn't provide a good summary of the level of system resources being consumed. The other commands we have covered to this point are superior resource assessment commands. On the other hand, I would guess **ps** is the most often issued system administration related command because it gives a snapshot of running process. There are a number of options you can use with ps, I normally use "e" and "f" which provides information about every running process and lists this information in full. The following figure is a partial **ps -ef** listing:4/27/92



```
# ps -ef

UID      PID    PPID   C    D    STIME      TTY    TIME     COMMAND
root     0      0      0    0    Dec 31     ?      0:00     swapper
root     1      0      0    0    Mar 16     ?      0:00     /etc/init
root     2      0      0    0    Mar 16     ?      0:00     vhand
root     3      0      0    0    Mar 16     ?      0:00     statdaemon
root     8      0      0    0    Mar 16     ?      0:00     unhashdaemon
root     6      0      0    0    Mar 16     ?      0:00     sickregd
root     11     0      0    0    Mar 16     ?      0:00     syncdaemon
root     45     1      0    0    Mar 16     ?      0:03     syncer
lp       49     1      0    0    Mar 16     ?      0:00     lpsched
root     129    1      0    0    Mar 16     ?      0:00     /etc/cron
oracle   2079   2071   0    0    07:01:44   ?      6:33     oracle
daemon   2088   96     0    0    08:45:07   ttyp0  4:55     /usr/bin/X11/X :0
becker   278    57     0    0    Mar 16     ttyp2  3:47     ANSYS
```

Here is a brief description of the headings shown in the previous figure.

UID - The user ID of the process owner.
PID - The process ID (you can use this to kill the process).
PPID - The process ID of the parent process.
C - Process utilization for scheduling.
STIME - Start time of the process.
TTY - The controlling terminal for the process.
TIME - The cumulative execution time for the process.
COMMAND - The command name and arguments.

ps gives a quick profile of the processes running on your system. If you issue the **ps** command and find some processes running that are hung you can issue the **"kill"** command. To kill the last process shown above for instance you would type:

# kill 278

sar For System Activity Reporter (Series 800 only)

sar is another HP-UX command used to gather information about the activity taking place on your system. Since there are many useful options to sar I'll give a brief description of three that are often used:

sar -u Report CPU utilization with headings %usr, %sys, %wio idle with some processes waiting for block I/O, %idle. This is similar to the "iostat" and "vmstat" cpu reports.

sar -b Reports buffer cache activity. A database application such as Oracle would recommend you use this option to see the effectiveness of buffer cache use.

sar -w Report system swapping activity.

timex To Analyze A Command (Series 800 Only)

If you have a specific command you want to find out more about you can use **timex**. **timex** reports the elapsed time, user time, and system time spent in the execution of a command you specify.

#2 HP-UX ACCOUNTING

HP-UX accounting supplies a lot of valuable information such as such as the amount of cpu time and memory consumed by each user, the time each user was logged into the system, and other such resource related information. Using the built-in accounting capability of HP-UX you can produce one report daily and one report monthly which contain a summary of the ways in which system resources are being used.

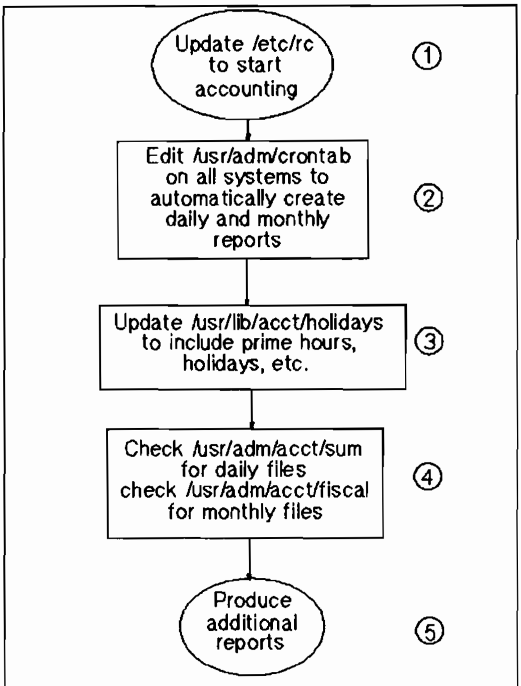
Unlike the first approach I covered (HP-UX commands), and the third approach I'll later cover (HP GlancePlus/UX) which give you information about your system in real time, HP-UX accounting provides you summary reports which you can review at you leisure.

I'll first described the way in which you can automatically produce one daily and one monthly accounting report and then describe the contents and meaning of all the entries in these accounting reports. These reports contain a great deal of useful information about the use of system resources; however, HP-UX accounting can go far beyond these reports should you require additional capability. The following list summarizes some of the functions HP-UX accounting can perform:

- Reports that summarize system performance and resource consumption (described in this paper).
- Automatically charge fees to users based on the resources they use.
- Reports of connect sessions for users showing login and logout information.
- Report disk space usage.

The Steps Required To Produce Daily and Monthly Accounting Reports

I'll assume you would like to have one report generated daily and one report generated monthly that has in it a lot of useful information (the specifics of what is included in these reports is later covered). The following figure shows the flow of automatically producing these two reports.



Update /etc/rc to start accounting
①
Edit /usr/adm/crontab on all systems to automatically create daily and monthly reports
②
Update /usr/lib/acct/holidays to include prime hours, holidays, etc.
③
Check /usr/adm/acct/sum for daily files check /usr/adm/acct/fiscal for monthly files
④
Produce additional reports
⑤

Flow of Creating Basic Daily and Monthly Accounting Reports

1) The /etc/rc file has in it a number of startup functions such as setting the name of the computer, setting the time zone, starting some network functions, and turning on accounting. To automatically start accounting when the system boots add the following line to/etc/rc:

```
/bin/su - adm -c /usr/lib/acct/startup
```

Only this line is required to start accounting in the /etc/rc file.

2) Making the above entry in /etc/rc starts system accounting; that is, accounting data is captured but not processed. In order to process the accounting data in the form of one report daily and one report monthly, you would create **cron** entries. **cron** is a program that runs other programs at the specified time. **cron** reads files which specify the operation to be performed and the date and time it is to be performed. Since we want to create accounting reports on a daily and monthly basis cron would be activated to create these files.

The format of entries in the **cron** file are as follows:

minute hour day month weekday username command

minute - the minute of the hour from 0-59
hour - the hour of the day from 0-23
day - the day of the month from 0-31
month - the month of the year from 1-12
weekday - the day of the week from 1 (Monday) - 7 (Sunday)

In order to create the appropriate **cron** entries for accounting you would login as "adm". The user "adm" has the appropriate access rights to accounting files so you would typically login as adm when issuing accounting commands. After logging in as "adm" you would edit the file /usr/adm/crontab creating the entries shown below. After making these entries you would issue the "crontab" command as follows:

```
crontab /usr/adm/crontab
```

This creates the file /usr/spool/cron/crontabs/adm with the following entries:

```
0  4  *  *  1-6  /usr/lib/acct/runacct  2> /usr/adm/acct/nite/fd2log

0  2  *  *  4  /usr/lib/acct/dodisk

5  *  *  *  *  /usr/lib/acct/ckpacct

15  5 1  *  *  /usr/lib/acct/monacct
```

The first line of /usr/spool/cron/crontabs/adm invokes the **runacct** shellscript at 4:00am every Monday through Saturday. Error messages from runnact are redirected to /usr/adm/acct/nite/fd2log. This daily accounting summary (and the monthly accounting summary created with the last line in the cron tab file) are in almost the same format. The contents and meaning of all entries in this file are later discussed in detail. **runacct** places the resulting daily report files in /usr/adm/acct/sum. The name of the daily report files are "rprtmmdd" where **mm** is the month and **dd** is the day of the month for which there port was created.

The second line of /usr/spool/cron/crontabs/adm runs the **dodisk** script at 2:00 am every Monday thru Saturday. This script should not be run on diskless nodes. **dodisk** collects disk usage information which is used by runacct when the daily accounting reports are created.

The third line of /usr/spool/cron/crontabs/adm insures that **pacct**, the process accounting file, does not get too large by running **ckpacct** at five minutes after every hour.

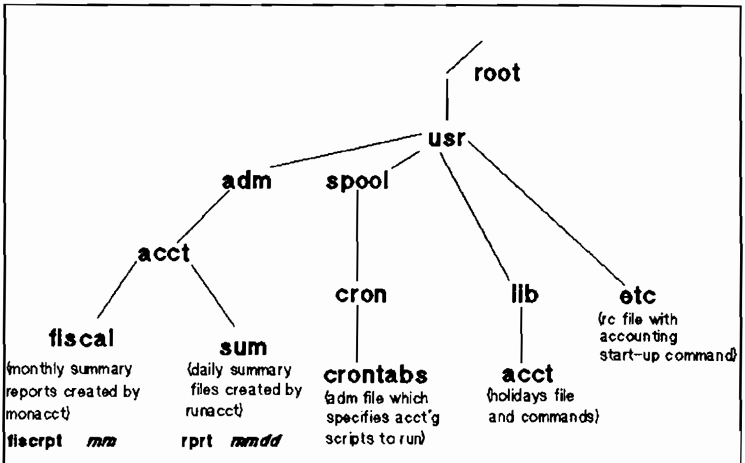
The fourth line of /usr/spool/cron/crontabs/adm runs the **monacct** script which merges all accounting data gathered over the course of the month into a single file. This script will be executed at 5:15am on the first day of every month. The monthly account summary created by **monacct** is in almost the same format as the output of the **runacct** script created in the first line. One important note about **monacct** is that after **monacct** creates monthly summary files and reports and places them in/usr/adm/acct/fiscal it removes the old daily accounting files from/usr/adm/acct/sum! If, therefore, you decide to manually produce a monthly report in the middle of a month keep in mind your daily accounting files will be deleted. The monthly accounting file produced at the end of the month will reflect only accounting information gathered in the second half of the month. The report files in/usr/adm/acct/fiscal are in the format fiscrptmm where mm is the month for which the report was generated.

3)The terms prime and non-prime are used in the categories of information described above. Prime time is the time during which the HP-UX system is most heavily used. You might describe the prime time as your working hours such as 8:00AM to 4:30PM. Non-prime time are the remaining hours in a day. Prime time, by default, is in effect on only weekdays and non-holidays. Non-prime time is in effect the balance of the time.

The system distinguishes between prime and non-prime hours by viewing the entries in "/usr/lib/acct/holidays". The following figure shows a sample holidays file.

```
* Prime/Nonprime Table for HP-UX  Accounting System

* Curr Prime           Non-Prime
* Year Start                Start

  1992  0800                1700

*Day of    Calendar    Company
*The Year  Date        Holiday

2          Jan 2       day after New Year
51         Feb 18      President's Day
83         Mar 25      Spring Holiday
147        May 27      Memorial Day
185        Jul 4       Independence Day
245        Sep 2       Labor Day
333        Nov 24      Thanksgiving Day
334        Nov 29      day after Thanksgiving
359        Dec 25      Christmas
```

4) The following figure shows the location of some accounting files in the HP-UX hierarchy. In order to insure the daily and monthly accounting files are created you would look in the /usr/adm/acct/sum directory for daily files and the /usr/adm/acct/fiscal directory for monthly reports.



root
usr
adm
spool
acct
cron
lib
etc
(rc file with accounting start-up command)
fiscal
(monthly summary reports created by monacct)
fiscrpt mm
sum
(daily summary files created by runacct)
rprt mmdd
crontabs
(adm file which specifies acct'g scripts to run)
acct
(holidays file and commands)

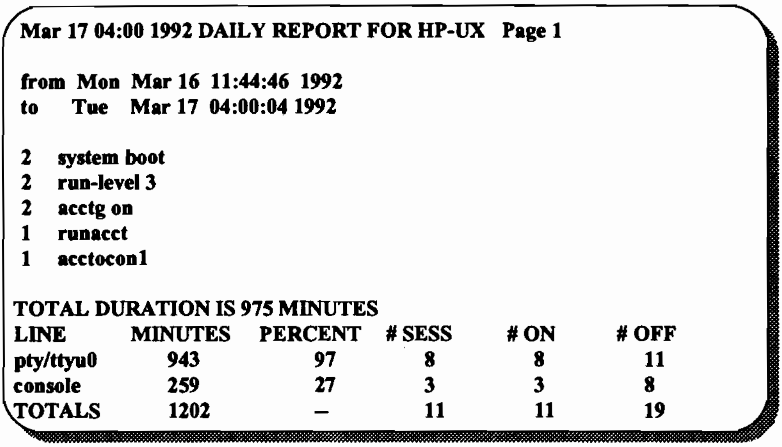
Some Accounting Files in the HP-UX Hierarchy

5) In this paper I cover one summary accounting report. If you would like to produce additional reports such as disk usage, connect sessions, or process accounting then there are additional steps required. The summary report covered here, however, contains a great deal of useful information that give insight to system administrators in any kind of HP-UX environment.

One Report Tells A Lot - Heading and Line Usage

The daily and monthly accounting reports prepared as a result of performing the above steps are almost identical in terms of the information each provides. The difference is the time period for which the information is reported. The daily report produces information about what has taken place over the last day. The monthly report produces data about what has taken place over the last month.

One minor difference in the daily and monthly accounting reports in HP-UX is that the daily report has a heading and line use section as shown in the following figure. This is a partial Daily Report With Heading and Line Usage.



```
Mar 17 04:00 1992 DAILY REPORT FOR HP-UX   Page 1

from  Mon  Mar 16  11:44:46  1992
to    Tue  Mar 17  04:00:04 1992

2   system boot
2   run-level 3
2   acctg on
1   runacct
1   acctocon1

TOTAL DURATION IS 975 MINUTES
LINE        MINUTES   PERCENT   # SESS    # ON    # OFF
pty/ttyu0     943        97        8        8       11
console       259        27        3        3        8
TOTALS       1202        –        11       11       19
```

The following bullet list describes some of the entries under the line use section.

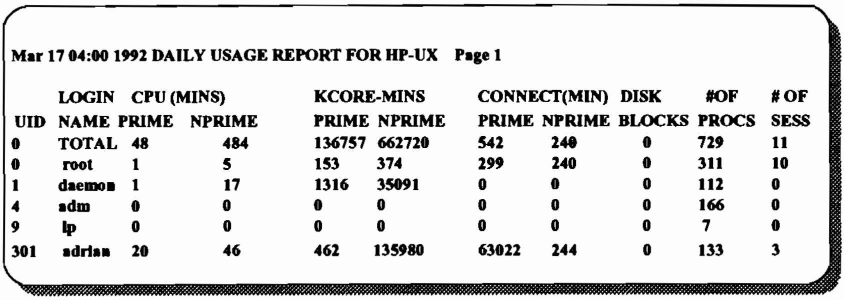
LINE - The terminal line or access port used in the report.
MINUTES - The total number of minutes the line was used since the last daily report was produced.
PERCENT - The percentage of "TOTAL DURATION" (shown in figure 4) that the line was in use:

PERCENT = (MINUTES / TOTAL DURATION) * 100

SESS - The number of times a login took place on the port.
ON - Identical to # SESS.
OFF - The number of times a user logged off as well as the number of interrupts on this line.

User Related Accounting Information

The following figure is a partial Daily Usage Report showing User Related information:



```
Mar 17 04:00 1992 DAILY USAGE REPORT FOR HP-UX   Page 1

     LOGIN   CPU (MINS)           KCORE-MINS         CONNECT(MIN)  DISK      #OF    # OF
UID  NAME   PRIME  NPRIME         PRIME   NPRIME     PRIME  NPRIME BLOCKS    PROCS  SESS
0    TOTAL  48      484           136757  662720     542    240      0       729    11
0     root  1       5             153     374        299    240      0       311    10
1    daemon 1       17            1316    35091      0      0        0       112    0
4    adm    0       0             0       0          0      0        0       166    0
9     lp    0       0             0       0          0      0        0       7      0
301   adrian 20     46            462     135980     63022  244      0       133    3
```

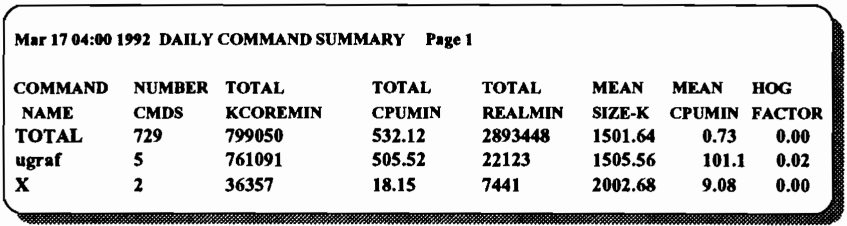
The following bullet describes the **user related** information that will automatically appear in both daily and monthly accounting report files when accounting is enabled:

UID - The user identification number of the user from whom the accounting information was gathered.
LOGIN NAME - The login name of the user from whom the accounting information was gathered.
CPU (MINS) - The number of prime and non-prime CPU minutes used by each user (prime vs. non-prime time was earlier described as defined in the/usr/lib/acct/holidays file)
KCORE-MINS - The amount of memory used by a user times the amount of time the memory was in use.
CONNECT (MINS) - The number of minutes each user was logged into the system.
DISK BLOCKS - The total number of disk blocks allocated for each user.
OF PROCS - The total number of processes spawned by each user.
OF SESS - The number of times each user logged in.

There are some categories of information which should be of interest to every HP-UX system administrator. All these categories of information are produced in the daily and monthly accounting reports.

Command Related Accounting Information

The following figure is a partial Daily Usage Report showing Command Related Information:



```
Mar 17 04:00 1992  DAILY COMMAND SUMMARY    Page 1

COMMAND   NUMBER  TOTAL        TOTAL     TOTAL      MEAN      MEAN    HOG
 NAME     CMDS    KCOREMIN     CPUMIN    REALMIN    SIZE-K    CPUMIN  FACTOR
TOTAL     729     799050       532.12    2893448    1501.64     0.73    0.00
ugraf     5       761091       505.52    22123      1505.56    101.1    0.02
X         2       36357        18.15     7441       2002.68     9.08    0.00
```

In addition to the user related information shown above, the same accounting report provides the following **command related** information:

COMMAND NAME - The command name for which execution statistics have been generated. Shell procedures are grouped under the name "sh".
NUMBER CMDS - The number of times that the command was invoked.
TOTAL KCOREMIN - The kcore minutes accumulated for the command. kcore minutes are the amount of memory used by a command times the amount of time the memory was in use.
TOTAL CPU-MIN - The CPU time used by the command.
TOTAL REAL-MIN - The accumulated real time in seconds used by the command.
MEAN SIZE-K - The average amount of memory in kilobytes consumed by the command.
MEAN CPU-MIN - The average CPU time consumed by the command is the total cpu time consumed by the command divided by the number of times the command was executed.

MEAN CPU-MIN = TOTAL CPU-MIN / NUMBER COMMANDS

HOG FACTOR - The hog factor of a command is the total cpu time consumed by the command divided by the minutes consumed by the command.

HOG FACTOR - TOTAL CPU-MIN / TOTAL REAL MIN

CHARS TRNSFD - The total number of characters transferred by the command.
BLOCKS READ - The number of file system blocks read and/or written as a result of executing this command.

Figure 6 shows the second part of a monthly accounting report which contains all of the system related information described in the bullet list above.

Last Login Information

The last piece of information provided in the accounting report is the date each user had logged in. If a user had not logged for several weeks or months you may want to consider archiving that users files or removing the user from the system. The format of last login is as follows:

yy-mm-dd username

An entry of **"91-11-15 libadmin"** would indicate the user libadmin had last logged in on November 15, 1991.

Accounting Command Summary

This paper provides an overview of accounting. You may later want to perform additional accounting functions or need to troubleshoot some aspect of accounting that isn't working properly. The following figure shows some of the commands in /usr/lib/acct and a brief description of each.

		Summary of Accounting Commands in /usr/lib/acct
Command	Type	Description
acctdusg	Program	Computes disk usage by login
accton	Program	Turns on process accounting
acctwtmp	Program	Adds boot record to wtmp file
chargefee	sh script	Charges specific users
ckpacct	sh script	Checks size of pacct file and restarts it if too big
diskusg	Program	Generate disk accounting data by user
dodisk	sh script	Takes a snapshot disk usage
fwtmp	Program	Fixes dates in wtmp when date command has changed them
holidays	Text	List of holidays
monacct	sh script	Produces monthly reports
nulladm	sh script	Reintializes files and insures correct ownership
prctmp	sh script	Print session record file
prdaily	sh script	Prints previous day's accounting summaries
prtacct	sh script	Prints the account records from tacct files
remove	sh script	Cleans up the /usr/adm/acct/sum director
runacct	sh script	Summarizes daily data
shutacct	Program	Adds shutdown record to wtmp file and turns off accounting
startup	sh script	Executed at boot time to enable accounting
turnacct	sh script	Turns on accounting to the file /usr/adm/pacct
wtmpfix	Program	Recognizes and repairs a bad wtmp file
acctcms	Program	Make command usage records - called by RUNACCT
acctcon1	Program	Make connect time records - called by RUNACCT
acctcon2	Program	Make connect time records - called by RUNACCT
acctprc1	Program	Make process records - called by RUNACCT
acctprc2	Program	Make process records - called by RUNACCT
acctmerg	Program	Merge processed records - called by RUNACCT
acctdisk	Program	Make disk usage records - called by DODISK
acctdusg	Program	Make disk usage records - called by DODISK

From "UNIX System Administration Handbook: E. Nemeth.

Accounting Provides A Good Overview

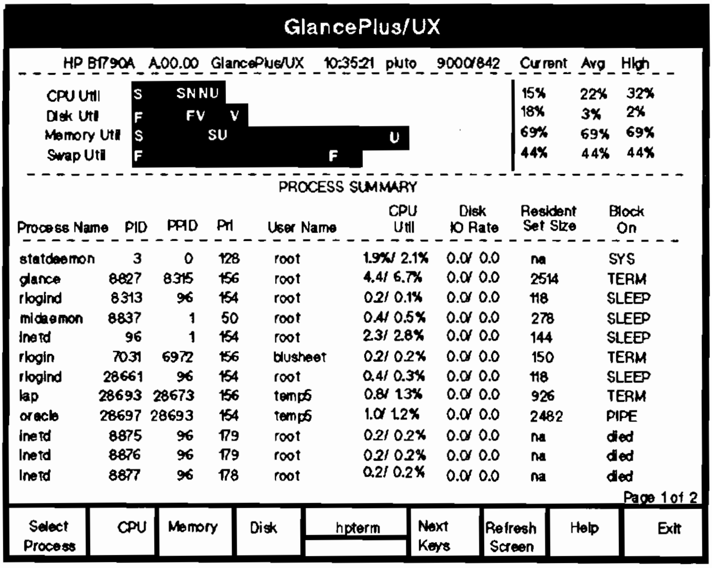
Just as some of the HP-UX commands earlier covered provided more information than you could use, the accounting files produced also provide a lot of information.

#3 HP GLANCEPLUS/UX

The first two techniques to get a better understanding of what your system is doing require you to deal with reports that contain some useful data as well as some data you don't care about. In the first case (issuing HP-UX commands) you get the advantage of obtaining data about what is taking place that very second on your system. Unfortunately you can't always issue additional commands to probe deeper into an area, such as a process, that you want to know more about. In the case of HP-UX accounting you get a lot of useful data gathered over time, however, there are a lot of reports generated so you spend a lot of time managing reports and pouring over them to extract the information which may be helpful to you.

A tool which can help get useful data in real time, allow you to investigate a specific process, and not inundate you with reports is HP GlancePlus/UX (GlancePlus).

The following figure shows one of the twelve interactive screens of GlancePlus.



```
                              GlancePlus/UX
      HP B1790A  A.00.00  GlancePlus/UX  10:35:21  pluto  9000/842    Current  Avg   High
-----------------------------------------------------------------------------------------
   CPU Util     S      SNNU                                          | 15%      22%   32%
   Disk Util    F       FV    V                                      | 18%      3%    2%
   Memory Util  S          SU                     U                  | 69%      69%   69%
   Swap Util    F                           F                        | 44%      44%   44%
-----------------------------------------------------------------------------------------
                                 PROCESS SUMMARY
                                                      CPU        Disk     Resident   Block
Process Name   PID    PPID   Pri   User Name          Util       IO Rate  Set Size   On
-----------------------------------------------------------------------------------------
statdaemon        3      0   128   root          1.9%/ 2.1%    0.0/ 0.0   na         SYS
glance         8827   8315   156   root          4.4/ 6.7%     0.0/ 0.0   2514       TERM
rlogind        8313     96   154   root          0.2/ 0.1%     0.0/ 0.0   118        SLEEP
midaemon       8837      1    50   root          0.4/ 0.5%     0.0/ 0.0   278        SLEEP
inetd            96      1   154   root          2.3/ 2.8%     0.0/ 0.0   144        SLEEP
rlogin         7031   6972   156   blusheet      0.2/ 0.2%     0.0/ 0.0   150        TERM
rlogind       28661     96   154   root          0.4/ 0.3%     0.0/ 0.0   118        SLEEP
lap           28693  28673   156   temp5         0.8/ 1.3%     0.0/ 0.0   926        TERM
oracle        28697  28693   154   temp5         1.0/ 1.2%     0.0/ 0.0   2482       PIPE
inetd          8875     96   179   root          0.2/ 0.2%     0.0/ 0.0   na         died
inetd          8876     96   179   root          0.2/ 0.2%     0.0/ 0.0   na         died
inetd          8877     96   178   root          0.2/ 0.2%     0.0/ 0.0   na         died
                                                                               Page 1 of 2
```

Two features of this screen are worth noticing immediately:

1. Four histograms at the top of the screen give you a graphical representation of your CPU, Disk, Memory, and Swap Utilization that is much easier to assimilate than a column of numbers.

2. The "Process Summary" has columns similar to **ps -ef** which many system administrators are familiar and comfortable with. GlancePlus, however, gives you the additional capability of filtering out processes that are using very little system resources by specifying thresholds.

Since most administrators have a greater appreciation of what is taking place on the system when it is in the form of a graph rather than numbers the GlancePlus interface is pleasant to work with.

Using GlancePlus you can take a close look at your system in all of the following areas:

- Global Summary of your system (shown in the figure)
- CPU Detail
- Memory Detail
- Swap Detail
- Disk Detail
- LAN Detail
- NFS Detail
- Diskless Server Resource Utilization
- Individual Process Information

Since the Global Summary shown in the figure tells you where your system resources are going at the highest level, I'll give a description of the lines on this figure. Keep in mind that the information shown on this screen can be updated at any interval you choose. If your system is running in a "steady state" mode you may want to have a long interval since you don't expect things to much change. On the other hand you may have a dynamic environment and want to see your histograms and other information updated every few seconds. In either case, you can change the update interval to suit your needs.

Line 1 provides the product and version number of GlancePlus, the time, name of your system, and system type.

Line 3 provides information about the overall state of the CPU. This tends to be the single most important piece of information administrators want to know about their system - **Is my CPU overworked?**

The CPU Utilization bar is divided into three parts:

1. "S" indicates the amount of time spent on "system" activities such as context switching and system calls.

2. "N" indicates the amount of time spent running "nice" user processes (those run at a low priority).

3. "U" indicates the amount of time spent running user processes.

The far right of line 3 shows the percent of CPU utilization. If your system is "CPU Bound" you will consistently see this number near 100%. You get statistics for Current, Average (since analysis was begun), and High.

Line 4 shows Disk Utilization for the busiest mounted disk. This bar indicates the percentage of File System and Virtual Memory disk I/O over the update interval. This bar is divided into two parts:

1. "F" indicates the amount of file system activity of user reads, and writes and other non-paging activities.

2. "V" indicates the percentage of disk I/O devoted to paging Virtual memory.

The Current, Avg, and High statistics have the same meaning as in the CPU Utilization description.

Line 5 show the system memory utilization. This bar is divided into two parts:

1. "S" indicates the amount of memory devoted to system use.
2. "U" indicates the amount of memory devoted to user programs and data.

The Current, Avg, and High statistics have the same meaning as in the CPU Utilization description.

Line 6 shows swap space information which is divided into two parts:

1) "F" indicates fixed local swap.

2) "D" indicates dynamic local swap.

All three of these areas; CPU, Memory, and Disk, may be further analyzed by using the F2, F3, and F4 function keys respectively. When you select one of these keys you move from the "Global Summary" screen to a screen that provides more in-depth functions in the specific area you have selected.

The bottom part of the screen shows the active processes running on your system. One of the most useful functions of GlancePlus is that you can select the process you want to view by the level of system resources it is consuming. You can, for instance, ask to view only the processes that are using greater than 10% CPU Utilization.

Although this a brief overview of GlancePlus I think you can see that it combines all of the best features of the HP-UX commands earlier covered and goes a lot further.

What should I look for when using GlancePlus?

Since GlancePlus provides a graphical representation of the way in which your system resources are being used the answer is simple - look to see which of the four bars have a high "Avg" utilization. You can then probe further into the process(es) that is causing this high utilization. If, for instance, you find your memory is consistently 99% utilized then you can select the F3 function key and have GlancePlus walk you through investigating which of your applications and users are memory hogs.

Similarly, you may be surprised to find that GlancePlus is showing your CPU or another aspect of your system resources to have a low utilization when you expected it to be high. Many slow systems are assumed to be CPU Bound. I have seen GlancePlus used to determine a system is in fact memory bound resulting in a memory upgrade instead of a CPU upgrade.

The difference between using GlancePlus to determine the level of CPU resources being used and the first two approaches is that GlancePlus takes out a lot of the guesswork involved. If you are going to justify a system upgrade of some type to management it is easier to do this with the hard and fast data GlancePlus provides than the detective work you may need to do with HP-UX commands and HP-UX accounting.

\GlancePlus is useful for providing this data. There are GlancePlus screens to help you identify the following three most common types of system bottlenecks:

1. **CPU Bottleneck**

   Use the "Global Screen" and "CPU Detail Screen" to identify these common CPU bottleneck symptoms:

   - Low CPU idle time
   - High capacity in User mode
   - Many processes blocked on priority (PRI)

2. **Memory Bottleneck**

   Use the "Global Screen", "Memory Screen", and "Tables Screen" to identify these common Memory bottleneck symptoms:

   - High swapping activity
   - High paging activity
   - Little or no free memory available
   - High CPU usage in System mode.

3. **Disk Bottleneck**

   Use "Global Screen", "Disk I/O Screen", and others to identify these common Disk Bottleneck symptoms:

   - High disk activity
   - High idle CPU time waiting for I/O requests to complete
   - Long disk queues.

The best approach to take to understanding where your system resources are going is to become familiar with all three techniques. You can then determine which information is most useful to you. The most important aspect of this process is to regularly issue commands and review accounting data so that small system utilization problems don't turn into catastrophes and adversely affect all your users.

Cooperative Computing Through Open Systems

Mikael Edholm
Marketing Program Manager
Hewlett-Packard Company

PAPER #2038

Open Distributed Computing

The objective of computer networks is to provide a mechanism for access and manipulation of information. In most organizations this information is dispersed across multiple locations. Data is stored in various formats, in different databases. Applications reside on a multitude of computer systems made by a variety of computer vendors. While one computer system may provide a certain organizational unit with its desired information, the same information may not be accessible by another unit. The sharing of information on an enterprise wide scale have in many organizations become almost impossible.

In order to efficiently operate a business, each individual within the enterprise should have the possibility to access all the information necessary, wherever located and whenever needed. Information, for design, purchasing, manufacturing, sales and accounting purposes, to name just a few, need to be accessed by the individuals concerned. Common resources, such as computing power and printing facilities must be shareable, scalable and widely accessible by all users as well as by all interdependent applications.

Only a truly open distributed computing system can provide this transparent access to information, in the manner required.

Only a seamlessly integrated network will enable users and applications alike to share the available resources.

Why Open Distributed Computing?

There are five key benefits provided by an open distributed computing system;

Flexibility, i.e. the capability to optimize the use of available resources. Shared resources that are transparently accessible reduce the overall cost of a system by limiting the use of duplicated, and often idle, resources. The basic technology enabling this flexibility is a Client/Server architecture. Shared resources are located in servers, dedicated resources act as clients of the servers. The network provides the infrastructure.

Moves, adds and changes to the system are handled by scalable resources. Moving an application to a larger or smaller system, from a different vendor, implies easy **portability** of application software. This is enabled by the use of standardized Application Programming Interfaces, (API). APIs provide software developers with common environments for application integration, allowing applications to be moved from one system to another with minimum effort.

Interoperability between clients and servers is guaranteed by adherence to standards. Common networking protocols enable transparent communications, also when the networked systems are provided by multiple computer vendors. Depending on the complexity of the network, these protocols should be able to address communication services ranging from simple terminal access to interprocess communications.

Shared information and resources are beneficial to a business only if reliably available. A networked system needs careful management to enhance productivity. **Centralized and distributed management** capabilities must be an integral part of any open distributed computing system. Each component of the system, whether a computer, a peripheral, a bridge or a router, must become a manageable object. An integrated management system can then provide network, systems and applications management capabilities, maximizing network uptime.

While open distributed computing undoubtedly offers great benefits over existing environments, there is still a large number of existing systems installed. Legacy systems need to become integrated with the new applications environments, to **protect investments** already made. Again, adoption of, and adherence to, standards will facilitate this move.

Open Distributed Computing Environments

Based on organizational needs, an enterprise typically contain one or more of four distinct computing environments. Two of these environments provide access to information. They are typically transaction oriented administrative applications environments. Common applications include ordering, invoicing and funds transfers. These environments can be described as **"Host Access"** and **"Global Access"** environments, respectively.

The other two computing environments support the activities of so called "knowledge workers". Typical applications include project management, design and development. These environments can be described as **"Resource Sharing"** and **"Cooperating Applications"** environments, respectively.

Most enterprises have host access and resource sharing environments implemented today. As the information technology needs evolve over time, so do the associated computing environments. Many organizations are already seeing the evolution from host to global access take place. Likewise, many organizations are beginning to distribute the resource sharing capabilities from their local area networks into global environments. In doing so, they are often adding network services that enable applications to interact and cooperate. In effect, they are implementing the basic fundamentals of the cooperating applications environment.

To fully understand the implications of the environments described above, it is necessary to examine them in a little more detail:

Host Access

Here, access to the processing power of local and remote hosts is the key. Terminals are attached to local servers through terminal controllers, remote hosts are linked via modems. Desktop computers, such as PCs and workstations access hosts through terminal emulation, in effect becoming terminals of the hosts. Peripheral resources are mainly centralized, attached to the hosts. Access is allowed on a time sharing basis, controlled by the host.

Local processing power exist at the desktop level, but no sharing of data occurs. To a limited extent, desktop users with frequent need of peripherals may be provided with individual resources. A printer may, for example, be added to a technical writers PC.

A typical application example would be an electronic mail system, storing and forwarding messages between employees. The mail application resides on a local or remote host, but is accessible from terminals and desktops alike. To the end user, application access is possible, but cumbersome. It is necessary to know the location of data and applications, as well as the means of accessing it. Elaborate log on procedures predominate, and productivity is highly dependent on the management of the centralized resources.

Global Access

Although common today, the centralized "Host Access" environment is rapidly being replaced by a distributed system, a global access environment. Each island of information, each individual local area network, is becoming part of a greater whole.

In a global access environment, data can be interactively manipulated without geographical constraints. It can also be stored and forwarded at a later, more convenient time, if multiple time zones or similar concerns are of importance. As more resources become available to the end users, and their applications, the total power of the networked system is greatly increased.

The move from host access to global access environments is taken in steps. First comes the communications infrastructure, consisting of bridges, routers, modems and associated remote access points. Based on the choice of infrastructure, a limited number of transport mechanisms must be put in place. The choices made regarding the deployment of transport mechanisms will influence the network services required, as will the applications development. Standardized application programming interfaces are used to ensure easy portability of applications to alternative systems. A geographical and resource view of the network is made possible by centralizing the network management functions to a single point on the extended LAN. Network management applications can then correctly balance the utilization of resources in this dynamic environment.

Moves, additions and changes to the networked system will frequently occur. A high degree of computing platform scalability is imperative. Adopting a client/server architecture will help in minimizing the growing pains of the network, as will a strict adherence to defined standards.

The end user perspective of a typical global access environment can be described using a sales statistics application as an example. The objective of the application is to produce a monthly sales report, based on multiple sources of statistical data.

In a simple computing environment, the end user may handle manual transfer of data, from local storage to spreadsheet application, and further on to the mail system. A typical user would download his Lotus 1-2-3 spreadsheet and enter data that has been stored on a local disk. Processing the data locally will produce an updated spreadsheet, perhaps with some graphic illustrations of current trends. The spreadsheet is then added to a memo, in a word processing application. Finally, the report is sent through the electronic mail application.

In a more developed environment, user agents handle such tasks on a pre-determined basis, requiring minimal end user interaction. Agents can be instructed to gather data from specific databases on certain dates, to process the information, and to produce and mail reports on other given dates. Still, the environment is subject to certain restrictions. Commonality of data representation is necessary. Location of information needs to be determined beforehand.

Resource Sharing

Physically, resources in a LAN environment are distributed over a limited geographical area. A workgroup with individual PCs can share output through nearby printers and plotters. Mini or mainframe servers provide centralized storage of large volumes of data. Complex application programs require external communications facilities. The network topology determines the use of enablers such as media, access methods and physical network components. Transparent access is facilitated by the introduction of common user interfaces.

The common user interface will typically provide the capability of displaying multiple applications simultaneously. A server may provide access to an electronic mail application, storing and forwarding messages that are displayed in one window on a client, i.e. a PC or a terminal screen. Another window may display a spreadsheet application, downloaded from a server seen as a networked drive. A third window may display locally available data, such as sales statistics, or production numbers stored on a local disk drive.

Through the use of a network operating system, time sharing of resources is possible. Printouts can be queued to handle multiple users, files can be shared by several users and applications. A design team, for example, can simultaneously work on different aspects of the same design, entering and deleting data in a common database, and print or plot the results on resources attached to a central server.

This type of environment is an improvement over the mere data access made possible by the virtual terminal and messaging facilities provided by the previously described environments. Multiple applications can run concurrently. Expensive resources can be shared, reducing idle time, and improving productivity. However, the resource sharing described is still subject to certain restrictions. Data needs to have a common representation, end users or application developers still need to know the location of the available data. There is also a minimal amount of integrated applications, requiring end users to manually switch between applications and combine the data into meaningful information.

Cooperating Applications

The fourth environment takes the client/server model one step further, by combining key elements of the previous stage with new capabilities. Task broking, network computing systems, enhanced file manipulation support and concurrent commit and recovery functionality split and distribute applications across the networked system.

To the end user, the network is completely transparent. There is no longer a need to know where data is stored, how it is stored or how to access the data. Even manipulation of data is highly automated, especially in the case of routine tasks.

The windowing environment remains, user agents handle most of the application work. Distributed databases are seamlessly integrated through equally distributed directory services. Remote procedure calls, transaction processing monitors and protocols handle the application logic in real time, independent of specific end user commands.

A cooperating applications environment is now possible. Take a travel reservation system as an example. A booking agent may want to arrange a trip from San Francisco to New York for a corporate employee. The employee already has an individual preference profile entered into a local database. Another database may contain corporate travel guidelines, providing spending limits, and corporate agreements with various travel related organizations. The travel reservation system also has access to car rental reservation databases, airline reservation databases, and hotel reservation databases.

The booking agent will see a windows based entry system on the terminal screen. By entering the employee name, destination and time, the booking application will automatically query the airline reservation system, and chose the appropriate flight. The booking application then queries the hotel reservation database, and reserves a room for the given period. If the hotel is further from the intended destination than would have been preferred, the booking application also queries a car rental reservation system for the availability of a car, the size of which is determined by the corporate guidelines in yet another database.

Having made all the necessary reservations, a report is generated, with a complete itinerary. The report is then automatically dispatched through the electronic mail network to the employee, while simultaneously tickets are being printed at the booking agents counter.

In this example, several applications cooperate, share information, and synchronize activities to perform a complex sequence of tasks. Nowhere along the line does the end user need to know how and where data was accessed. The on-line transaction capabilities of the system, combined with the distributed applications and logic takes care of business, without human involvement after the initial request is entered.

A truly open distributed computing system is at work.

Open Distributed Computing Architecture

The creation of an open distributed computing environment has to be based on a conceptual framework, or architecture. From an organizational perspective, business applications are the key components, the center of activities. Applications that perform critical tasks do so in interaction with their immediate environment. An application may "talk" to a user at a desktop, "ask" a data base for information, "tell" a processor to compute a formula or "request" access to other applications via a communications link. When operational, applications have "dialogues" with users, data bases, systems or other applications.

In a traditional computing environment, applications dialogues were conducted through the use of detailed, system specific calls. Each computer system required a different type of dialogue. Applications were custom made, efficient but hard to modify, and difficult to adapt to changing surroundings.

In an open distributed computing environment, applications have generic dialogues. The way a service is requested is defined by standard application programming interfaces, APIs. APIs provide sets of common macro commands that translate generic application service requests into system specific commands. Applications developed for use with standard APIs become highly portable, from one system to another, from one user interface to another, from one data base to another, etc.

As can be gathered from the above, there are four major classes of APIs; user interface APIs, data base APIs, operating system APIs and network APIs. The first three enable the "open" in open distributed computing. The fourth is the key to extending the open into a distributed environment.

Beyond the APIs lie the services. A service is what a resource can provide to an application. A user interface service can for example provide windows for a graphical user environment. A data base service can be a facility for definition and manipulation of data in a relational data base. Operating systems services can provide language independent system calls. Again, these services provide the "open" environment. The distributed functionality lies within the network service domain.

To better understand how applications are distributed, and conduct dialogues, it is necessary to take a closer look at a subset of the application integration architecture, the network architecture.

Network Architecture

The enabling technology behind a distributed computing environment is best described within a modular network architecture. Five building blocks make up this model;

At the base, the **communications infrastructure** block defines the available media, access methods, data link protocols and the corresponding routing algorithms. Above the infrastructure, **transport mechanisms** handle tasks such as end-to-end reliability, session establishment and termination, syntax encoding and basic application integration services. The **services** block defines what functionality the network can provide the applications. Virtual terminal services, messaging, file services provide base level capabilities. Directory services and interprocess communications protocols handle the more advanced capabilities. The **management** block interacts with the previous three blocks, to provide fault detection, isolation and correction, as well as operations support mechanisms.

Surrounding these four blocks, and shielding the applications from the underlying functionality, is the **application programming interface** block. The common APIs enable standardized access to each high level service, as well as each underlying level of a layered protocol architecture.

Communications Infrastructure

Going back to the first block, the communications infrastructure, it can easily be seen that a communications infrastructure need to handle several tasks. Applications typically require support for local area networks, LANs, wide area networks, WANs and combinations thereof. They may also require a wide range of speeds, access methods and media. Basic infrastructure technologies can be defined by the possible throughput of data and by the latency of data. Throughput is typically measured in MBytes/s; latency, the time between request and reply, is measured in ms.

Current networking technology provides wide area networking capabilities with sufficient throughput for store-and-forward applications, such as electronic mail and electronic data interchange, EDI. On a typical WAN, approximately one tenth of a Mbyte/s throughput can be achieved. The latency is typically above one second.

Existing local area networks provide better throughput, and better latency. Throughput can be up to one Mbyte/s and the latency might come down to 1 ms. This is adequate for most types of file transfer operations, the dominating application environment on a LAN.

In the future, these networking environments must be enhanced. The requirements of emerging applications for imaging, animation and remote data base access are far beyond today's limitations. WANs will need to handle throughput speeds up to 100 MBytes/s, with a latency of less than a second. LANs need approximately the same throughput speeds, but with a latency coming down towards 0.1 ms, if an application or an end user is to have a satisfactory dialogue over the networked system.

Transport Mechanisms

The reliability of the communication dialogue is handled by the second block of the network architecture; the transport mechanism. Two types of standard networking transport protocols exist; de jure standards which are based on adherence to internationally agreed upon committee developments and de facto standards which are based upon individual developments submitted to the public domain, or on market presence by dominant vendors.

Personal Computer networks typically use de facto standards, for lack of equivalent de jure standards. De facto standards are mainly SPX/IP, used by Novell NetWare systems, Xerox Network System, XNS, which exist primarily in engineering applications environments and TCP/IP, primarily existing in mixed PC and LAN environments.

LANs typically use TCP/IP, or to some extent XNS as the dominant multivendor de facto standards. For IBM environments, SNA is obviously dominant. The de jure protocols are all based on Open Systems Interconnection, (OSI), a rapidly emerging consensus solution to network transport mechanisms.

In the wide area environment, TCP/IP and SNA make up the bulk of de facto standards, again with an emerging presence of de jure standards based on OSI.

Network Services

The Network Service block can be defined along similar lines, de jure and de facto. An additional dimension will classify their complexity. As computing environments grow in complexity, the services required will change. Thus, host access environments depend on virtual terminal services while global access environments add store and forward messaging services to the basic terminal connectivity. The resource sharing environments are heavily dependent on file manipulation services. The cooperating applications environment incorporates all of the former services and while adding powerful interprocess communication services.

Starting with the basic services, virtual terminal capabilities can be implemented in several ways; based on the choice of underlying transport mechanisms. If TCP/IP is used, Telnet is the service of choice. If OSI is used VTP is the corresponding service. For IBM environments, 3270 protocols are frequently used. Messaging and file services are divided along the same lines, store and forward protocols such as OSI's X.400 are frequently used in the wide area environmen while de facto standards tend to dominate in local area networks.

Standardized messaging services are provided by the X.400 store-and-forward protocol. Enhanced addressing functionality is provided by the X.500 distributed directory services protocol. On top of the X.400 standard, two information structures are added; Electronic Data Interchange (EDI), based on the de jure EDIFACT definitions and Office Document Architecture and Interchange Format (ODA/ODIF). EDI is the standardized method of computer to computer exchange of business documents like purchase orders and invoices. EDI replaces paper transactions and considerably shortens the ordering, billing and payment cycles in a variety of corporate environments. ODA provides standardized compound document transfer, such as integrated text and graphics, and at a later stage, images. The integration of these services and information structures, combined with access capabilities to LAN based electronic mail systems enable the design and deployment of large scale messaging backbones.

The local area network environment will be enhanced by the implementation of the distributed computing environment (DCE), where file services, directory services and security enablers are the key components. Engineering environments in particular will benefit from the integrated technologies of DCE.

Transaction Processing services is the underlying networking technology for a wide range of service industries. Standardized transaction processing is intended for use with applications such as automatic teller machines, airline reservations systems, hotel reservations and car rentals, to name just a few examples. The key protocol is the OSI TP, supplemented by remote database access (RDA), remote procedure calls (RPC) and synchronization services such as concurrent commit and recovery (CCR).

In summary; the emergence of distributed applications are pushing network service technologies towards integration and enhancement of existing services, as well as towards development of certain new areas of functionality. These changes are primarily driven by the needs of three key application areas; Messaging, Transaction Processing and the Distributed Computing Environment.

Management

The management block interacts with all three previously mentioned blocks. Open distributed computing management can be structured into three levels, depending on the view of the network. In its simplest form, management consists only of interconnection management, providing a geographical view of the network. The second step adds the possibility of managing resources on the network. At the top, an applications view add management of the interaction between applications across the network.

There are five areas of management functionality at each of the three levels;

- **Performance management**, which deals with the measurement of network throughput, and system uptime.

- **Fault management**, which deals with the identification, isolation and correction of network errors.

- **Configuration management**, which deals with the structure and dynamics of the network.

- **Accounting and billing management**, which deals with the financial aspects of the provided network services.

- **Security management**, which deals with the access privileges and data integrity across the networked system.

Throughout the networked system, events occur. These events are monitored by the management system, providing a base of network statistics. These statistics are analyzed within the five functionality areas. Corrective action is then taken by management applications that will issue commands to the managed objects of the networked system. These objects can be clients or servers, in the form of computers, instruments, peripherals or network components such as bridges, routers and the like.

For example, imagine a manufacturing facility with three sequential production lines. Goods are moved from line A, to line B, to line C. At each step, bills of material must be printed, and the goods forwarded just-in-time, to ensure a smooth production flow. Any disturbance will cause a pile goods in progress to build up between the lines, and a shortage of material will occur at following lines.

Now, a networked printer at the first line fails to print a bill of material, due to a failure in the modem connection. Interconnect management detects the fault and sets off an alarm. Information about the fault is passed on to the systems management level. Here the fault is isolated, the output is spooled, i.e. re-routed to a nearby printer. The applications management meanwhile informs the manufacturing application of the delay in data delivery. production is slowed down in anticipation of new data, thus avoiding inventory build-up at an intermediary processing stage.

Each management level in this model builds on the previous, until applications finally interact to achieve the desired fault correction, and restore normality to the network.

Application Programming Interfaces

Application Programming Interfaces are key to the portability of applications. Application programming interfaces surround the infrastructure, transport mechanisms, services and management blocks, providing a common mechanism for accessing the underlying network functionality. By adhering to common standards, such as the portability guides (XPG) published by the X/Open Company Ltd, applications may achieve a high degree of portability. In other words, applications can be developed on small, dedicated development engines, and later ported onto production systems from various vendors.

Currently, a range of APIs address the various levels of a network protocol stack. There is an API for each and every service at the application layer, as well as APIs for each and every level below. An application developer thus needs to learn and use multiple APIs. To minimize the difficulties of applications development, work is in progress to define unified API's. These APIs will sit at the top of a hierarchy of programming interfaces, addressing functionality requests to the appropriate level of the network protocol stack. Ultimately, a single high level should emerge from these efforts.

Network Protocol Choices

Based on the above, which set of network protocols should an applications developer chose? The answer depends on two key factors; the functionality requirements, and the need for interoperability with other computer systems.

Imagine a diagram where the X-axis shows the degree of functionality, on a scale from low to high, and the Y-axis shows the degree of interoperability, again on a scale from low to high. Within this diagram, various networking protocols can be positioned according to their values of the two criteria.

Proprietary protocols, residing on a multitude of legacy systems, typically provide a high degree of functionality. A large number of services are provided, and the transport mechanisms are usually optimized for the particular systems providers' equipment. Unfortunately, this is achieved at the expense of interoperability with other systems providers equipment. Proprietary protocols will thus be positioned in the upper left corner of the diagram, representing high functionality with minimal interoperability.

At the opposite end of the scale, the lower right hand corner, de facto standard protocol suites reside. These protocols are often part of the public domain, freely adopted by any systems manufacturer, and widely implemented throughout the computer industry. De facto standard protocols suites, such as TCP/IP, are today often bundled with computer systems from most major manufacturers, and enjoy a significant market presence. Their widespread acceptance is largely due to their ease of implementation, something that has been achieved at the expense of functionality. While they provide most basic network services, none has been optimized for any given manufacturers equipment. Thanks to the rapid increase in computing power, this is no longer a major concern. More serious is the lack of new developments. For a protocol set to be considered a de facto standard, and to become widely deployed, it needs to be perceived as stable. New developments can thus take considerable time to reach significant market presence levels. Also, any given modification to an existing protocol may seriously impair its usefulness for interoperability with other systems.

Today, only one protocol suite can claim to bring the best of both worlds, high functionality in combination with a high degree of interoperability; The Open Systems Interconnection, OSI suite of protocols. OSI today encompasses more than 80 different standards, providing the five basic classes of network services, a choice of transport mechanisms, and a choice of underlying infrastructure components. The rich and varied functionality of OSI has now been combined into so called profiles. A profile is a vertical slice through the OSI protocol stack, limiting the number of implementation options, while still addressing specific needs of an industry, or an application environment.

Thanks to the use of profiles, OSI is rapidly becoming THE multivendor protocol, surpassing de facto standards such as TCP/IP or NetBIOS in functionality and versatility. Other de facto standards, such as DECnet and SNA offer additional functionality, but are considerably less multivendor than OSI. The multivendor aspect of DECnet, as well as that of SNA, is now being addressed through the gradual incorporation of OSI services into the two proprietary architectures.

OSI is rapidly moving in two directions; towards increased functionality and towards increased multivendor connectivity. More and more companies endorse and adopt OSI standards as the basis for their communications architectures. More and more protocols are added to already existing OSI standards, extending their usefulness to an ever increasing range of application environments.

OSI Deployment

User acceptance of OSI varies around the world. In Europe, one company in three is now implementing OSI. Close to another one in three is committed to implementing OSI in the near future. An additional one in four is seriously interested in OSI. In the US, only one company in ten is currently implementing OSI. Twice that amount is committed to OSI, and one company in three declares an interested in OSI. Adding the implementing, committed and interested segments together, approximately 60% of US respondents in a recent Yankee Group survey would fall into this new segment. The situation is similar in Europe, although the trend towards implementation is more pronounced.

What we are experiencing is a time lag. The US market is lagging the European market in the adoption of OSI standards. Currently this time lag is in the range of one to two years. The time lag can largely be explained by looking at the introduction dates of mandatory Government OSI profiles. In the UK, OSI networking protocols became a mandatory requirement for government information technology purchases as early as in January 1989. Sweden followed suit in April 1989. Other Nordic countries made the same decision later the same year.

In the US, a Government OSI Profile is in force as of August 1990, a full year and a half year behind the UK. Based on current maturity of OSI protocol standards, and the availability of OSI based products, the US can be expected to close this gap in the near future. OSI is now poised for growth, on both sides of the Atlantic.

It is also worth noting that GOSIPs typically include more than just a subset of OSI standards. A typical GOSIP is divided into four sections, a T-subprofile, an A-subprofile, an F-subprofile and a C-subprofile.

The T-subprofile, for Transport, specifies OSI protocols to be used for media access, internet routing and transport. The A-subprofile, for Application, specifies the upper layers of the protocol stack, from the session layer to the application layer. The F-subprofile, for Formats, extends the specification outside of the basic OSI protocols, into the area of data exchange. Here, the Electronic Data Interchange, EDI, format is specified, as is the Office Document Architecture, ODA. The C-subprofile, for Characters, deals with regional character subsets, to correspond to the various languages that have to be supported by implementations of OSI.

The inclusion of format standards have considerably added to the momentum of OSI deployment. OSI based EDI implementations are currently experiencing triple digit growth in most industrialized nations.

OSI will not become THE multivendor standard overnight. A long period of transition from currently used de facto standards, to the de jure OSI Standards, can be expected. There are four basic phases of this transition; pilot implementations, subnetworks based on OSI and a long coexistence phase before the ultimate dominance of OSI comes about.

In the pilot phase, the typical user will require a simple platform onto which he can port a distributed application. The user will install OSI in a controlled environment, isolated from the operational backbone of his organization. The pilot will serve as a knowledge build-up basis for the application developer.

In the second phase, applications based on OSI emerge. The former pilots become integrated with the corporate backbone, but handles specific tasks. Which tasks will be chosen are typically based on which unique OSI functionality has driven the pilot to success. To preserve the existing de facto standards based backbone, dual or parallel protocol stacks need be part of the target applications platform.

During the long coexistence phase, it is expected that OSI will gradually overtake the deployment and use of de facto standards. New applications will use OSI network protocols, older applications are likely to stay with de facto standards. Their peaceful coexistence is assured by the fact that output messages, from the dual stack machines, can be transferred over a common link. When application integration become necessary between heterogeneous networking environments, application layer gateways can be used. At a lower level, multiprotocol routers enable OSI and de facto standards to share a common media.

Standards exist for both the first two approaches, and products have been entering the marketplace since early 1990.

If distributed applications need share a common link and media, but in a single vendor environment so called tunneling may be a temporary solution. Tunneling provides the possibility to encapsulate messages based on one networking standard, into frames sent by another networking standard. As no standards for this approach exist, the multivendor connectivity benefit of OSI will be lost when proprietary tunneling techniques are used. Again, it can be clearly demonstrated that only strict adherence to internationally agreed upon standards can guarantee the desired multivendor functionality that is required of the network.

Conformance and Interoperability

To be truly multivendor, applications using OSI products, must be sure that the services, transport mechanisms and links conform to the existing standards. Conformance to a given profile, or part thereof, can be tested at designated test centers throughout the world. Non-profit organizations such as the Corporation for Open Systems, COS, in the United States, or the Standards Promotion Agency, SPAG, in Europe are recognized test centers for their respective markets. For the Japanese market, INTAP provides a similar service.

In addition, a few vendors of OSI products have become accredited by such test centers, and have in-house testing capabilities to aid in their development efforts. At the time of writing, Bull HN Information Systems Inc. and Hewlett-Packard Company are the only COS accredited test centers in the US.

Conforming to a standard does not guarantee interoperability, however. There are various ways an OSI protocol can be implemented, and numerous options to the basic services may or may not be included in an OSI product. Interoperability is the key to the distributed applications environment, and should be demonstrated by an OSI vendor upon request.

Many vendors have long had interoperability testing programs in place. By linking systems via OSInet, EurOSInet, OSIcom and related organizations' networks, vendors can easily test their respective product offerings for interoperability. In addition, major trade shows often feature OSI LAN and WAN interoperability as a key topic of the demonstrations provided.

OSI Today

What does OSI offer its users today? Consider the following example;

Five organizations interact in a global marketplace; a manufacturer of finished goods, a distribution center, a sales outlet, a supplier of raw materials, and a financial institution. The manufacturing company has recently developed a new product, XYZ, and is now actively seeking customers around the world. One of the manufacturer's sales outlets receives a request for product information from a customer. Some of the desired information is immediately available, but a key component is lacking; the new pricing structure. A request for pricing information is sent off to the manufacturing entity, asking for an update.

The manufacturing company stores product information in a central database; a huge file with new prices, options and quantity discounts. In response to the request from the sales office, a sales support application dispatches the information over the corporate OSI network, using the OSI File Transfer, Access and Management protocol, FTAM. At the manufacturing company, a cooperating manufacturing management application decides to make sure sufficient quantities of product XYZ can be made available to the sales office to meet the expected demand. It sends a query off to the local distribution center, asking how much of product XYZ, option ABC is available. The distribution center database automatically responds.

For this simple exchange, no file transfers need occur. A basic store and forward mechanism for data exchange is usually sufficient. The applicable OSI service would then be based on the CCITT X.400 set of protocols. If the exchange needs to take place in real time, as is frequently the case, OSI can still handle the request. The OSI Remote Database Access (RDA) protocol would then be used in place of X.400.

Assuming the distribution center application does not find sufficient quantities of the product in stock, it reports this finding to a a just-in-time (JIT) manufacturing application back at the factory. If JIT decides to make additional quantities to meet the expected demand, raw materials need to be ordered from a supplier and shipped to a production center. Orders have to be generated, bills of material need filling out, and invoices must be paid. Ordering and invoicing are considerably more complex tasks than file transfer or electronic messaging. Basic OSI services need to be supplemented by additional functionality to handle business applications of this kind.

This additional functionality is provided by a translation service called Electronic Data Interchange, EDI. An EDI translator uses X.400 as its transport mechanism. There is now a UN standard called Electronic Data Interchange For Administration, Commerce and Trade, EDIFACT, which is rapidly being deployed around the world. EDIFACT not only formats data. It also includes facilities for copying messages to multiple locations. This is typically used for providing banks with the transaction data necessary to complete the financial portion of a deal. When a bank receives EDI information, it will extract the applicable data and transfer the correct funds from one account to another. Coming back to the example above; EDI based on X.400 is used to order, document and invoice the necessary raw materials for the new production run. Copies of the transactions are then sent to the banks in questions. Funds transfer applications at two locations are notified of the activity, and use OSI TP to execute the transfer between the relevant accounts in real time.

OSI services are useful not only between remote locations, over wide area networks, or WANs, but are also highly applicable within the manufacturing entity. Expanding the example above to include intracompany communications within the production center, the following scenario evolves;

When the JIT application receives notice that a production run should start, using the material ordered through the EDI application, job data is immediately transferred to a production line cell controller. As this would typically be in the form of a set of instructions, arranged in a file, FTAM would be the applicable OSI protocol. The cell controller downloads instructions to the factory floor equipment. These instructions are real-time commands, such as start, stop, open, close etc.

OSI provides a generic set of semantics for communications between intelligent factory floor devices. More than eighty specific commands are included in the OSI Manufacturing Messaging Specification, MMS, used for shop floor control applications. MMS commands also enable upload of data, from programmable logic controllers, PLCs, or numerical controllers, NCs, on either a continuous basis, or in response to alarm triggers. In certain cases, such as large production cell, a large number of intelligent devices are interconnected. For the cell controller to find the correct device, a directory service may be a necessary addition. OSI provides this functionality through the use of the OSI Directory Services protocol, X.500.

Once the production run is finished, a bill of material is sent to the scheduling application, residing in an area manager computer. Since the amount of data to be transferred is likely to be large, the OSI file transfer protocol, FTAM, will again be the most applicable service to use. If an insufficient quantity has been produced, or additional demand has materialized, the area manager may decide to initiate a new production run. For this purpose, more raw material will be needed, and an EDIFACT message will be sent to the supplier, commencing a new production cycle.

Meanwhile, the area manager will inform the distribution center of the availability of the recently produced goods, and arrange for its transportation to the geographical location desired.

Conclusion

As shown above, the creation of a cooperating applications environment is a possibility already today. Using a sound application architecture, the combination of open systems elements such as standardized user interfaces, open databases, scalable processing power and extensive network communications facilities can result in considerable productivity gains.

If the underlying network architecture is based on the Open Systems Interconnection suite of protocols, an application developer can be assured of investment protection, as well as continued functionality improvement over the long term.

And if the dialogues between these elements is supported by standardized applications programming interfaces, a high degree of applications portability will add significantly lower cost of ownership to the benefits already achieved by the functionality gains.

To sum it up, open distributed computing provides four key benefits, based on the same number of underlying technology enablers;

- **Flexibility** is achieved through the implementation of a **Client/Server** architecture.

- **Portability** of applications is achieved through adherence to common **application programming interface** specifications.

- **Interoperability** between different vendors equipment is achieved through the adherence to internationally agreed upon networking **standards**.

- **Productivity** is enhanced through deployment of fully distributed **network, systems and applications management** applications.

Hewlett-Packard OpenView Vision

Steve Bunten

Colorado Networks Division

Hewlett-Packard Company

Hewlett-Packard's approach to management is driven by structural changes in the information technology structure as our customers' environment evolves from a central computing facility to distributed topologies dominated by many departmental computers and the pervasive presence of PCs and workstations. Today Network Management is mostly distinct from System Management. The installed customer base manages disparate networks and systems with a high degree of granularity. With the rapid deployment of open systems and client-server topology Network and System Management requirements are changing.

HP's Network and System Management strategy is predicated on the belief that ultimately Integrated Resource Management, not just Network and System Managemt, is the valued perspective of our customers. This reflects the need for customers to interact with the distributed network systems environment in the *solutions domain, not in the technology domain.* Our focus is on reaching new heights in ease of use and accessibility of the information widely dispersed in a global computing network.

This paper will briefly discuss the customer problem and some of the products and solutions available from Hewlett-Packard and third parties to meet those needs. In addition, we see a new management paradigm which we will be evolving toward in the next couple of years. Finally, the paper will close with some insights about why OpenView represents a better way for customers to manage their network computing environment today and into the future.

Customer Needs Driving HP OpenView

Fundamentally, customers need access to the information to make informed decisions to enhance their strategic and competitive business positions. This is true whether they be in engineering, production, marketing, sales or administration and whether their business is in manufacturing, service, financial, telecommunications or other. Additionally, they are experiencing organizational changes such as globalization and downsizing. Many industries are under increased competitive pressures in an environment with decreasing regulation. There is a need for improving the information technology in this area for increased service levels, cost effective deployment, managing and operations.

Transparent Network System Management enhances user productivity and his effectiveness in utilizing the critical information to make informed decisions for competitive advantage. The telephone is a good example of an appliance where the technology is transparent: It is easy to use and does not require a wizard or operator to effectively utilize it for global communications. It is also reliable and even during instances of failure, there is a high degree of self healing ability from a user's perspective. It is broadly accessible and transparent when moving from one service provider or technology to another and is affordable (in most parts of the world). We need a similar functional appliance approach to managing and operating the information technologies.

HP OpenView Mission

Hewlett-Packard's mission for OpenView is to deliver superior value to our customers in order to become the world's leading supplier of integrated management solutions for the open distributed client-server environment. HP aims to be the best at managing the customer environment, not just an HP network or systems environment. This means that we must be effective in operating in a multivendor heterogeneous environment populated with PCs, workstations, servers, and mainframes. So, what value does HP provide today? OpenView enables an increase in service levels while reducing the cost of operations for business focused IT solutions. This is achieved by broad adoption of OpenView by the leading network equipment providers, system vendors, and standards groups as the platform upon which to build management services, applications, and solutions.

HP OpenView Winning Plan

HP OpenView began in the late 1980's with a clear focus on establishing OpenView architecture and integrated management technologies as premier in the industry. This has been accomplished through three distinct steps:

1. Open Architecture (1988-89) - OpenView was recognized as the premier architecture by early 1990. Industry consultants, system integrators, and network planners worldwide proclaimed OpenView's distributed modular design viable and superior to all other architectures. In addition, OpenView was the only architecture and strategy targeted directly to Industry Standards and open distributed computing systems environments. OpenView continued to excel in the technology area by introducing the industry's first products which provided dynamic discovery of network elements and automatic map layout of those elements. The architectural strengths designed into OpenView from day one have stood the test of time and have played a significant role in the design wins with the Open Systems Foundation's (OSF) Distributed Management Environment (DME), 50 Network Equipment Providers (NEPs), and thousands of large end user customers.

Network and System Management today is a people intensive task. It reuires a high degree of knowledge of the network and systems environment, a high level of expertise to utilize the Network and System Management tools and extensive experience to interpret the data. Customers need more automation, more intelligence built into these management tools to fully utilize the information technology solutions to enhance the business effectiveness. However, if we manage the individual Network and System elements primarily as separate parts, it is left up to the systems administrator to integrate and manage the environment of networks, computers and applications. In the future, we must manage the system as a whole with more of the analysis/synthesis and expertise embedded in the system. This will lead to improved productivity through increased service levels and reduction in costs.

Issues of Managing Open Systems

Many of the mission critical network computing environments are virtually unmanaged today or dependent upon gurus capable of understanding the complexity and vagaries of planning, installing and operating the network computing environment.

Management tools today are limited, focused on managing the separate and disparate systems in networks. Management applications usually offer a single function and capable of managing only a single type of element and, as a result, management stations span the computer room walls. Solutions to manage the Data Center have not kept up with the distributed multivendor reality of the information technologies. Customers are scrambling to establish a coherent data management center or network operations center. This move begins by designing and deploying network equipment and systems which can be accessed from management systems that know about the capability of the equipment, configuration, and status. The unified management framework enables the interoperability and integration of these managed objects and is fundamental to integrated information management.

Integrated management systems promise to be a significant step to address these issues. They have largely been focused on managing the proprietary environment and lack comprehensive services and high value applications necessary to effectively manage the computing environment. The integrated management system (the technology) is not the only obstacle; the availability of skilled network engineers and system administrators who have the skills and expertise are difficult to recruit as well as expensive and challenging to retain.

Information Technology Appliance

The end user, and to a great extent the systems administrator, should view the network system as an appliance. That is, they should be able to access the information from their desktop or through the application interface without being encumbered by the underlying system technologies or network infrastructure. The most effective system interaction insulates the user from direct contact with the system functions and network technologies.

2. Pervasive Platform (1990-91) - The strategic focus on making OpenView the pervasive integrated management framework in the industry was significantly enhanced by the technology license agreements and delivery of OpenView on multiple hardware platforms including HP, Sun and IBM. With these moves, HP has changed the rules of open systems. In addition to a solid framework, we have provided many innovative features such as Discovery and Automatic Map Layout, the Application Builder, and the Extensible Agent. These capabilities have not only added excitement to the OpenView products, but they have delivered real value to our customers. These products and strategic partnerships provide our customers stepping stones that enable unified management of their environments and provide the business incentive for NEPs and independent software vendors (ISVs) to build their applications on OpenView.

3. Dominant Solutions (1992+) - Design wins such as IBM and OSF have highlighted the strength of the product capability and increased the interest by the press, industry consultants, and customers. Recent surveys in Data Communications Magazine and Computer World gave OpenView "User Choice Award" and "Top Rated," respectively, and are indicative of the strong value our customers have found in the OpenView products. While exciting technology innovations are gratifying, it is especially rewarding to see our customers praise HP for providing solutions to their problems.

   The OpenView management station was made extensible so that the customer could easily customize it to his environment. Features such as the Application Builder enables customers to access and manage networked equipment by building a management application without becoming familiar with the OpenView develoment environment or even writing a single line of code. The Extensible Agent allows customers to access systems on the network and write simple scripts which can gather data about CPU utilization, disc space usage, operating system information, or any other information they need to manage their environment. These are examples of innovative products which increases competitive differentiation and which are highly valued by customers.

   An increased HP focus on solutions will be important but by no means sufficient to deliver the breadth and depth of solution functionality needed by our customers. HP will provide focused solutions in areas where there is an identified problem for which we have an excellent understanding and appropriate product capability. Example include the Network Node Manager for fault, configuration, performance analysis of TCP/SNMP-based networks, OpenSpool for print management, OmniBack for storage management and numerous others, including system performance analysis, software distribution, and networked licensing. In addition, the widespread adoption of OpenView by ISVs is now paying dividends for our customers in terms of solutions they can deploy today.

A Bigger Vision

HP is driven by a bigger vision than the traditional definition of Network and System Management. Our agenda is to place the priority on unified management of the customer's environment. This is accomplished by delivery of high value solutions on multiple hardware platforms for easy integration with the installed base, and a rapidly growing portfolio of comprehensive management services and applications solutions. This capability is delivered with the consulting and services, which includes planning, designing, implementing, operating and supporting the networked computing environment. HP OpenView is a business, not just a technology, in which HP is investing for sustained leadership.

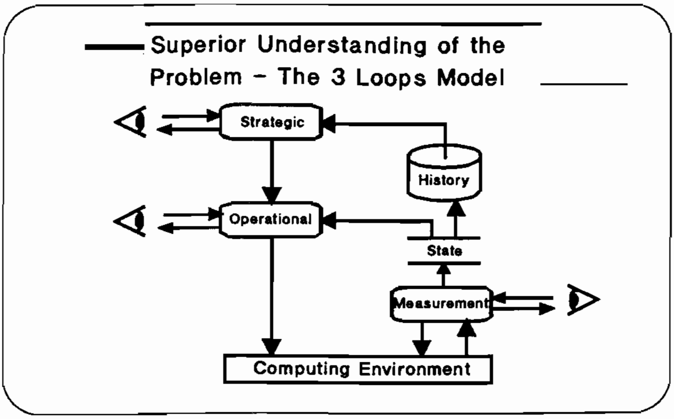
The New Paradigm - Process Management

To deliver on this bigger vision requires continued technology innovation, enhancements, product innovation, and a new paradigm which we call process management. The fundamental enabler of Integrated Resource Management is understanding the tasks and prcesses of managing a customer's environment. HP has had a long standing vision of integrating network management with system management and network measurement technologies from our dispatched, embedded and on-line measurement systems. I will give you an example of how these work together later in the paper.

Another level of process management is building intelligent solutions which deliver more automation for managing the network environment. This capability is especially important for highly complex synthesis/decisions which enable optimization of the information technology resource. This will enable us to move routine tasks to lower skill personnel and utilize the experts for network planning and translating their expertise/exerience into the solutions. This builds on HP's leadership in applying object oriented technologies to the problem of managing the processes associated with resource management. Network and System Management exposes the user to the granularity of the networks and the systems. Ultimately, we want to make individual resources very transparent and appliance-like. We are shifting away from manual management to more automation. The state of the industry today is that users are managing the parts. We need to provide management of the whole - networks, systems, and applications. This is somewhat analogous to an airplane with parts flying in close formation vs. an airplane as a system. Customers want to interact with the information they need to do their job, not take their most talented people and turn them into managing the pieces that make up the network and system environment. We are moving from managing the equipment to managing the tasks and processes.

The Three Loops Model - Superior Understanding of the Problem

The management process evolution or paradigm shift is kėy to making a major improvement in the move toward integrated resource management. Some of the process changes are already apparent. As an example, observe the move from managing networks and systems separately to integrated management or the move away from managing individual work groups distinct from distributed LANs and the Data Center. Other process changes such as the integration of measurement, control, analysis and strategic planning are less well defined. I will use the **3 Loops Management Model** to illustrate the area of Management research currently underway at Hewlett-Packard Laboratories. Additionally, I will show you an example of a prototype which capitalizes on this 3 Loops process view. The key to Integrated Resource Management is the accurate collection of information to establish the configuration, functional status, performance, etc. of the networked environment.



Superior Understanding of the Problem - The 3 Loops Model

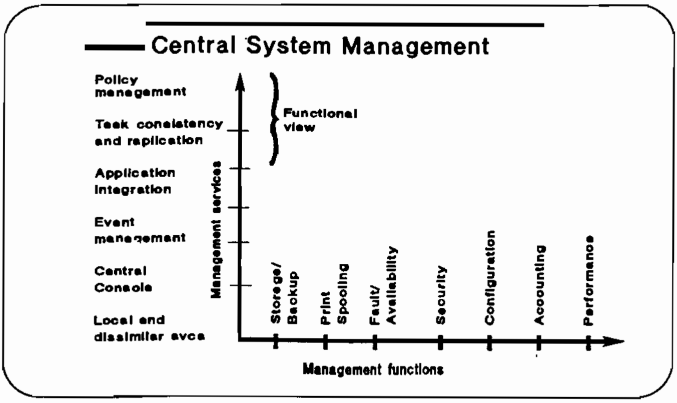
The **Measurement Loop** today is managed with dispatched instrumentation and on-line embedded measurement systems such as the HP LAN Probe as well as measurement technology built into the computing environment itself. HP has products which integrate the information collected by the HP LAN Probe within the OpenView Network Node Manager. As far as automating the management processes goes, this is just a basic utilization of the measurement information.

The second loop is the **Monitoring and Analysis Loop**. This is exemplified by a research prototype which collects network traffic information, topological information, and other network vital statistics, and stores it in a history data base. This information can be queried and analyzed, and compared with current information to better understand changes in the environment and take corrective actions.

The third loop is the **Strategic Planning Loop** which has another level of tool sophistication to enable the user to do *what ifs* on this data to optimize the overall functionality and performance. This understanding, and subsequent development of solutions, has the potential to enable realtime reconfiguration of the network environment.

Process Integration and Task Management

In this example I would like to show you how Process Integration and Task Management can greatly enhance the systems administration task and operational effectiveness. Imagine there are a number of distinct solutions for print management, file management, fault management, configuration management, and so forth. These solutions are not well integrated with the overall Network and System Management or Integrated Resource Management environment. Many of these are tightly coupled to the installed base network and system devices and will remain in place for years to come. The question is how can we capitalize on the integrated environment to turn this into an efficient and effective management process? In the following diagram the vertical axis represents examples of increasing functional integration. This is enable through a central system console to all of the applications running in the networked environment. This provides the ability to check for functionality at a detail level without moving to the network environment where the application is running and to do correlation of information between related events. The next step provides **Remote Event Handling**; that is, integration of events, with more intelligence and automation about how to interpret the events relative to system availability and functional status.



The next step is **Integrating the Data** at a common point with both real time and historical data base tools to further diagnose and implement corrective actions at remote sites. This brings the need for task control so systems can be rebooted, network routing changed, file servers reconfigured, etc. The next step of **Central Management** is policy administration to ensure consistent policies are administered with regard to system access, system configuration, etc. We see a very powerful domain for integrating existing system and network management applications providing for integrating existing system and network management applications providing for integrated administration of remote resources and for centralizing the system administration for more cost effective resource management.

Intelligent Resource Management - Manage the Environment

I will now conclude with a perspective of how **Integra.eu Resource Management and Embedded Intelligence** can deliver a much more effective information technology environment that operates at much lower cost and higher service availability. Today we manage network equipment and systems with very disparate sets of tools which require a high level of skill and training to effectively utilize. We understand the common services such as Discovery, Mapping, Data Management, Event Management, etc., which appear in products such as OpenView to deliver a new level of functionality in an integrated network and system management environment.

The **applications** are critical and we will not be able to maintain the rate of integrated application development consistent with the expanding need and complexity of managing the information technologies. Just to keep up we must find a way to encapsulate and integrate existing applications. We must provide new functionality which gives a common user interface and powerful manipulation tools to effectively utilize the installed base of applications.

Managing the **information** is a critical dimension in the integrated resource environment. Currently relational data bases are the best choice in terms of mature technology and information integration across multiple domains. The **object oriented data management model** will be important in the future so that we can deal with a broad range of objects which could be more effectively managed in an object oriented environment.

The task and process innovation will provide an incredible pay off for more cost effective management of the critical IT environment. One can get a sense of the opportunities through some examples but I would like to utilize one that is even more basic - accounting systems. Twenty years ago when computers were first utilized to automate some of the accounting tasks, we basically automated the tasks of the accounts receivable and payable clerks. This simply automated the old process with very marginal increases in the clerks productivity. As we learned more about how to redesign the financial information flows, technology enabled the process innovation to bring extraordinary advances in timely information access with a dramatic reduction in the people required to deliver that information.

Leading Customers Through Change

HP is leading the customers through this change. It is a complicated and challenging part of our Business Plan. At the risk of over simplifying this process, lets just say that Standards are vital. We have recognized for years the importance of open systems and standards work. We have been freely licensing the *Best of Breed* middleware in the interest of facilitating the integrated information environment. The licensing of OpenView by OSF is a clear example. Providing OpenView on multiple hardware platforms is another clear example of working to integrate the network computing environment and provide more unified management. Integration afforded by the example above is important, but there are many fundamental changes that are facilitating the integration. Integration of networks and systems, integration of HP and Third Party applications are all equally important here today. Innovation is key to leading customers through change, making systems easier to use through innovative products such as the Extensible Agent, the Application Builder, Automatic Discover & Mapping, to deliver value today which enhance customers IT utilization and facilitate evolution of current tasks and processes to take advantage of it. Overall, the consulting and services is fundamental since this is a complex environment today and promises to be increasingly challenging with new technologies, including the extended desktop, multi-media, and advanced telecommunication services.

2040

Creating Telephone Switch to Computer Applications

Bob Alley

Hewlett-Packard
19420 Homestead Road
Cupertino, CA 95014
(408)447-3167

ABSTRACT:

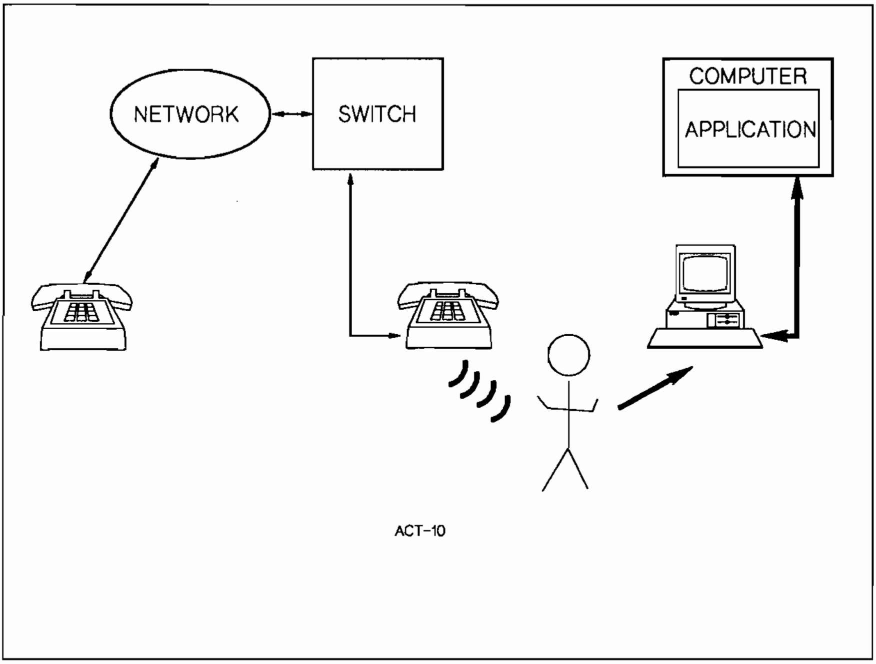
The ability to integrate the telephone system with your computer applications is a reality today. The technology is becoming pervasive. Functional managers (Users) are beginning to realize that this technology can reduce operating costs and at the same time increase customer service levels. Application Programming Interfaces (API's) are available to provide applications developers with the tools needed to implement this technology. This paper will discuss the key application design considerations that should be understood when implementing this new technology.

Introduction

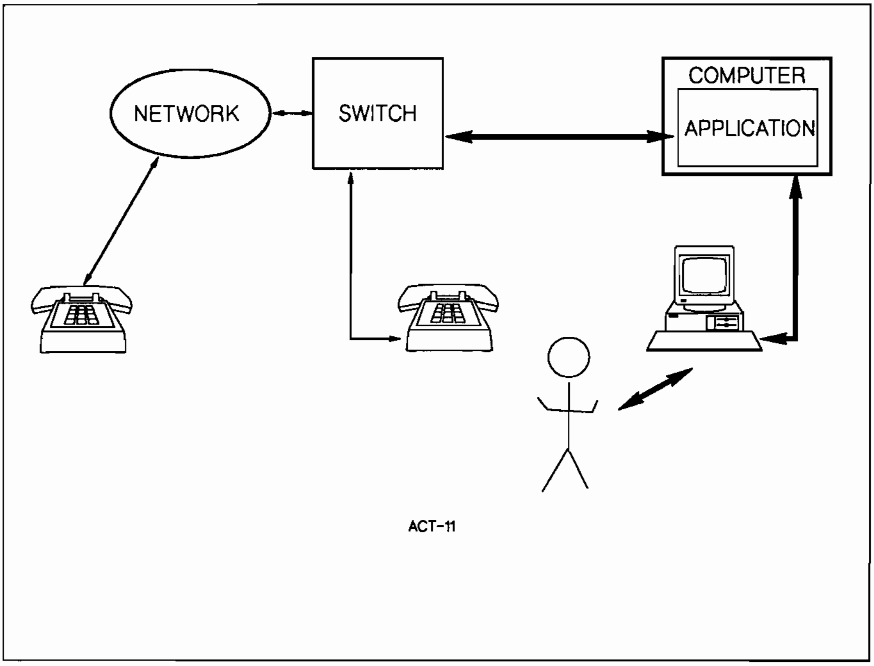
Applied Computerized Telephony (ACT) provides an application programming interface (API) to allow an application developer to integrate telephony functions into an application.

Although ACT provides the API tools there are several application design considerations that should be understood before implementing ACT into an application.

A major difference between a classical application environment and one that deals with telephony is that the telephony environment is much more event driven.

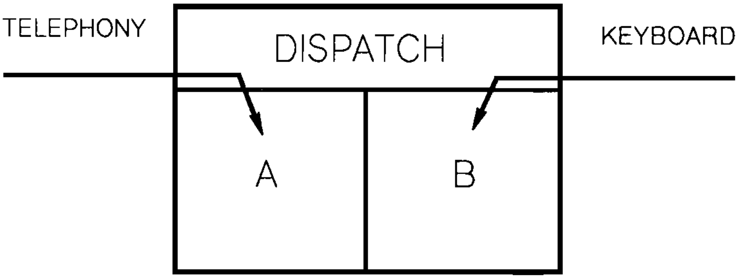


In the classical application environment a telephone would ring and the agent would hear the phone ringing, answer the phone, identify who the customer is and then retrieve information about that customer. The computer application only has one event to deal with, the keyboard input. The person is processing the telephone events manually.



In an event driven application environment the application can receive either a keyboard event or an ACT telephony event and the application must be able to accept and process both at any time.

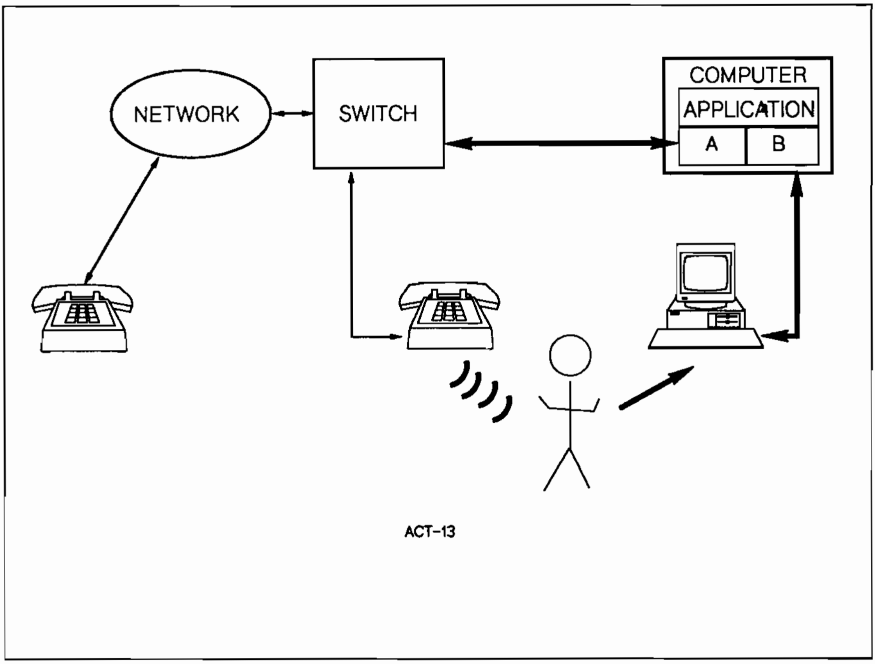
APPLICATION



ACT-12

In these types of applications a dispatch function is typically implemented. After the process has performed it's setup functions it then utilizes a dispatch function to determine what type of event is occurring and then perform the appropriate procedures for that type of event. It then return to the dispatch function to wait for another event to occur.

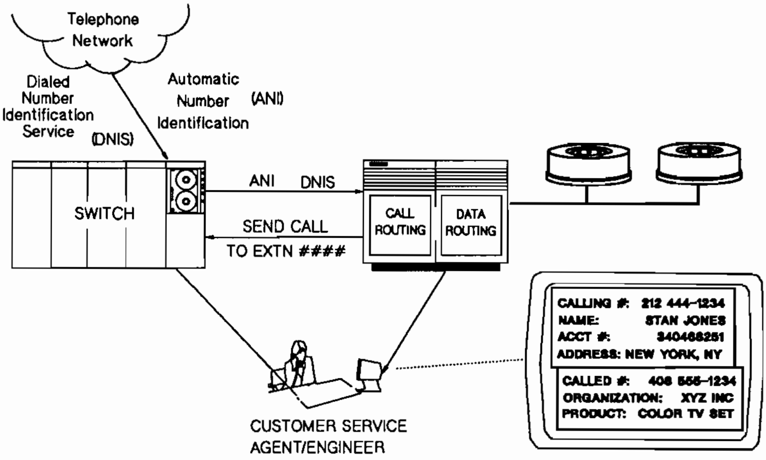
These events are associated with a file input from the terminal or interprocess communication connection. The file number indicates where the event originated. In HPUX the select call is used to indicate an event input completion. In MPE the IOWAIT intrinsic is used.



There is a technic that can be used with the classical environment. In this approach the agent hears the phone ring and through the keyboard tells the application to go process the ACT events. This is a less desirable approach because it does not take into consideration other types of events, however it can be implemented very quickly in an existing application resulting in automatic data retrieval and time savings.

Most installations will have both inbound calls and outbound types of calls, however the two functions are typically thought of separately. The following discussion will address each separately but keep in mind that there is no pure inbound or outbound installation and most will need to address them both together. An example is an outbound call can result in an inbound call back if the person is not available at that time. Like wise an inbound call can result in an outbound call if no agent is available to take the call at that time.

INBOUND APPLICATION


Telephone Network
Dialed Number Identification Service (DNIS)
Automatic Number Identification (ANI)
SWITCH
ANI DNIS
SEND CALL
TO EXTN ####
CALL ROUTING
DATA ROUTING
CUSTOMER SERVICE AGENT/ENGINEER

CALLING #: 212 444-1234
NAME: STAN JONES
ACCT #: 340468251
ADDRESS: NEW YORK, NY

CALLED #: 408 555-1234
ORGANIZATION: XYZ INC
PRODUCT: COLOR TV SET

ACT-14

Inbound Call Applications

Most medium to large call centers will utilize an ACD (automatic call distribution) software package on the switch that queues and manages the distribution of calls to a pool of agents. ACT can work with or without the ACD package. Although there are a multitude of ways an application can be designed to handle incoming calls we will discuss three basic approaches.

Passive Inbound

The passive approach does not get involved in the routing of the call. The application receives an event from the switch indicating an inbound call is being presented to a specific phone. The event will indicate the calling number, the called number and the number of the phone that the call is being presented to. The application then uses these numbers to identify who the caller is from the calling number, why the caller is calling from the called number and which agent received the call from the presented phone number, It is then up to the application to deliver the corresponding screen information to the terminal/PC/workstation.

Computer Assisted Routing

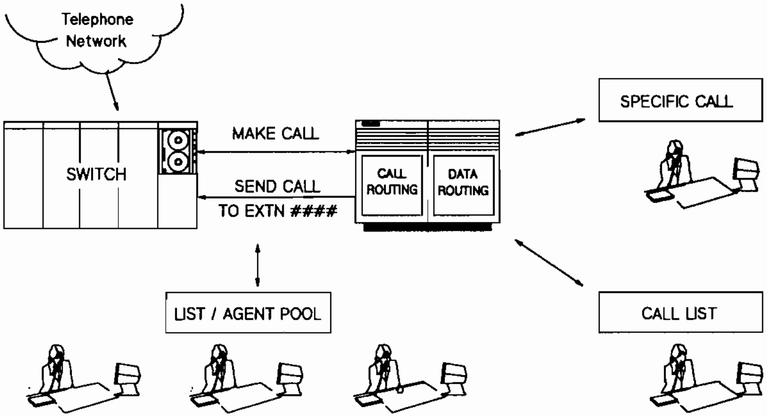
In this approach the application will receive an event from the switch indicating that a call is being received by the switch. If the application indicates a routing preference the switch will route the call based on this preference. If the application does not respond within a predefined time period the switch will use its own routing logic. The application will receive a second event indicating that the call is being presented to a specific phone. At this point the application also presents the screen information.

Computer Based Routing

The third approach is used when the routing of the call is controlled totally by the application. In this approach the switch will route the call to a virtual phone number and send an event message to the application. The application can then transfer the call to the desired location and deliver the appropriate screen information.

Another key design consideration in all three approaches is weather you want the phone to automatically answer the call or wait for the application to answer the call.

OUTBOUND APPLICATION



Telephone Network

SWITCH

MAKE CALL

SEND CALL TO EXTN ####

CALL ROUTING

DATA ROUTING

SPECIFIC CALL

LIST / AGENT POOL

CALL LIST

act-15

Outbound Call Applications

Just like inbound, outbound applications can use three basic approaches.

Specific Call / Single Agent

In this first approach the application uses the makecall procedure to make a call on behalf of a specific phone. This approach is typically used when customer information is already displayed on the screen and the agent now wants to call the customer. The application gets the phone number and places the call is response to a soft key, mouse or keyboard input.

Call List / Single Agent

The second approach uses a call list. The application uses a list to drive the presentation of the screen information and to make the call. This can offer substantial productivity gains as will as improving the overall business process. The application can be written to make the call at the same time the screen information is displayed or it can be written to only place the call after the agent has reviewed the information and is ready to proceed. Once a call is complete the application moves to the next entry in the list. If a busy or no answer is encountered then the application can move to the next item in the list and return later to try again. There are many, many uses for this approach.

Call List / Agent Pool

This last approach is most often referred to as predictive dialing. It is similar to the call list except that the agent doesn't see the screen until the caller has answered the call and the call is transferred to the agent. In this approach the application can be launching more calls then there are currently agents available. The idea is that not all calls will be completed and therefore you seek to maximize agent productivity.

Warning, this approach may be outlawed in the future because of telemarketing abuses in the past.

We have talked of inbound and outbound application, however remember that call centers typically do both. On inbound calls you may need to call the customer back and on outbound call you may leave a message to call back and therefore you want the returning call to be routed back to the originator.

Number Uses

The phone numbers that are delivered to the application can be used in many ways. We will describe a few and you will probability find even more ways to use them.

Calling Number

The calling number is usually though of to identify the caller. In some cases the calling number may represent a main number for a PBX and therefore it may indicate a list of persons that can be displayed to the agent to select from once the customer has identified who they are "Hello this is John Smith from XYZ company."

Another use of this number is to indicate the location of the caller using the area code and prefix. This can be used to rout a call to the person/group supporting that area.

Called Number

The called number is some times referred to as the DNIS (dialed number identification service). It is the number that indicates the number that the caller dialed.

This number can be used to identify why the customer is calling like in the case of an 800 number in a magazine or on television. It can also be used to indicate a group like service, sales, order processing. It can also be used to route calls based on language (i.e. English, Spanish ..).It can also be used to indicate who the caller is if the number is used by only that customer.

Phone / Workstation Mapping

The application is responsible for maintaining the relationship between the agent's phone number and the agent's terminal/PC/workstation session. This can be done using a table file or at agent logon. There are probable an endless number of ways this can be performed. Just remember that is more then one application is using ACT, these applications must not send conflicting messages to the switch.

Working With IVR

In some applications you may want to sent a caller to an IVR (interactive voice response) unit to get a customer ID, an account number or other information. This can be accomplished by routing the call to the phone number of the IVR device and communicating with the IVR to gather more information. Then transferring the call to the desired agent. Communication with todays IVR's can be accomplished using either an async port or LAN connection depending on the interface available From the IVR vender.

Controller Concept

Another design consideration is the use of an ACT Controller process. This is when you have many processes or a process occurrence for each agent and you don't want each process to have ACT control, however you do want these processes to receive event information. A single application is written using the ACT API and it in turn communicated to these other processes using whatever type of interprocess communication you wish. This can prove to be a simple way of implementing a subset of functionality to existing applications quickly.

Centralized and Decentralized Call Centers

Call centers can be totally centralized or totally decentralized. When the call centers are distributed but the data base / application is centralized ACT servers must be at each location where there is a phone switch. The centralized application will then need to establish an API session with each remote ACT server.

Telephone Switch Differences

It is important to note that not all telephone switches are the same and not all provide the same functionality or process calls the same way. ACT provides a common API for all supported telephone switches however there are differences in how they should be used. Over time these differences will be reduced.

Summary:

ACT provides the capability to integrate the phone system with computer applications. You can see from the proceeding discussion that there are many ways to implement this technology. We recommend that you phase this implementation to take advantage of those capabilities with the biggest return to the organization first.

Even though the deployment of this technology is being driven based on business needs it is also a fun technology to learn and implement.

Managing Networked PCs

Michael J. Rafeld
Hewlett-Packard

8010 Foothills Boulevard
Mailstop R5YF
Roseville, California 95678
(916) 785-5615

ABSTRACT

PCs are playing a larger role in offices as more employees are performing their tasks with the aid of PCs. Networks are being added to link PCs to mini-computers, mainframes and to other PCs to improve productivity. This paper focuses on the problems associated with managing networked PCs, and offers some suggestions as to what to look for in evaluating tools to aid in solving these problems.

In many instances, there is a PC administrator or administrators identified to maintain this network. It is this person, or persons, who first realizes the breadth of issues in managing networked PCs. It is hoped that this paper might be used as an evaluation checklist against which networked PC management products can be evaluated. One such product, HP Software Vendor, is used as a case study to illustrate how one tool has approached addressing this set of problems.

An extensive set of references and a bibliography are included to provide directions for further independent research. Many of the references are detailed product evaluations, which will aid a PC administrator in further evaluation.

OVERVIEW

PCs have been ubiquitous in the workplace for at least the past five years. Offices everywhere have realized the uses of PCs in automating common tasks by using spreadsheet and word processing applications. In many cases, these offices are also supported by data processing systems. These systems, perhaps HP, or DEC, or IBM machines, are used for large, number-crunching applications; or for administrative tasks, such as payroll.

Recently, these two worlds have been connected, as local area networks (LANs) have made an appearance in the office, connecting the standalone PCs with the more powerful machines. This move has afforded further possibilities for PC users, including inter-office communication and hugely expanded amounts of disk space, using virtual disks placed on the larger machines.

The term *networked PCs* actually consists of three, interconnected *physical* components:

Servers - The machines connected to the LAN, acting as network managers. The servers are responsible for routing data which is passed through the LAN, and for acting as file servers, enabling their disks as virtual disks, or *shared disks f*or the PCs. Servers also act as print servers, spooling information for both printers and plotters.

LAN - The physical component of the network, consisting of wire (or fiber), gateways, repeaters, and other physical devices which facilitate the transmission of data.

PCs - Personal computers which take advantage of the other two components to expand their capabilities beyond that of merely standalone computers.

The one *logical* component of networked PCs is the *networked operating system* (NOS). This is the software running on both the PCs and the servers which manages the information on the LAN. Examples include Novell Netware, Microsoft LAN Manager, and Banyan VINES.

Although there is some discussion about the advantages of networked PCs over standalone PCs, this paper wili **not** address the issues of initial planning, installation, and setup of a PC network or the selection of a NOS. It also will **not** discuss the ongoing administration of either the network or the servers, although there are a couple of very detailed references in the bibliography, listed under "Network Management". Instead, this paper will focus primarily on the ongoing administration of the **PC component** of a networked PC environment.

ISSUES OF PC ADMINISTRATION

In this section, various aspects of PC administration are described in detail. Each aspect is described as it applies to the administration of standalone PCs (if it *does* apply), and then how the networking of PCs both enhances and complicates this administration. There are also a couple of excellent articles which address the same issues, but from other perspectives [1, 2].

Software Installation and Updates

In a standalone PC environment, the installation and updating of software is straightforward and time consuming: the insertion of floppy disks into the PC, and the invocation of an installation or update program.

Common elements of installation and update software include:

- Querying the installer for drive and directory where application is to reside, or already resides.
- Querying the installer for hardware specifics such as display type, amount of memory, or attached peripherals.
- A dependence upon previously installed software (e.g. Microsoft Excel is dependent upon Microsoft Windows having been previously installed).
- Updates to the CONFIG.SYS or AUTOEXEC.BAT files.
- Asking for a unique *key*, or a password provided with the installation or update software, essentially locking the software to a particular PC.
- And, in the case of update programs, data generated by previous versions must be accommodated.

In both the standalone and networked PC environments, administration of software on more than, say, twenty PCs can be very time consuming. Installations and updates can take many hours to perform, disrupting work schedules. Coordination of large groups of PCs is sometimes difficult, with updates occurring overnight, or during weekends.

The problems of large-scale software installation and updates on standalone PCs has been well documented [3-11]. For at least the past five years, the problem has been understood and a thorough reading of the references reveals a clear need for tools to distribute PC software over networks. The term which has been coined for this is *electronic software distribution*, or ESD. There are currently three approaches to ESD, all discussed below: *network-enabled installations*, *tool-assisted installations*, and *network-aware applications*.

Network-Enabled Installation

The least sophisticated approach refers to products which can be installed onto a PC across a network from a shared disk, but which do not provide any special features to accommodate the network. Many popular applications can be installed in this way, simply by copying the floppies to a shared disk, and then having the installation or update program invoked from each PC, after it has connected to the shared disk. In this way, the network merely *enables* the installation, but the software is not really aware of it in any way.

Care must be taken with this method if installation or update programs expect subsequent floppies to be inserted into a disk drive. Some installation programs are not sophisticated enough to look on a shared disk for this information. Other installation programs require a unique key or password to be installed on the PC, making simple network-enabled installation impossible. However, if neither of these are the case, the installer may interact with the program just as if they were installing from the floppies.

This scheme can greatly reduce installation and update time simply by eliminating the need to "feed" floppies to each PC. However, it will in many cases still require someone to interact with the installation job at each PC while installing or updating the application.

Installing software in this way over a network may also cause some problems. Concurrent installations by a number of PCs may cause the network to slow down considerably. Strict control must be kept of which PCs have access to the application on the shared disk, otherwise there may be problems with proper licensing for the software (see "PC Software License Management," below).

Tool-Assisted Installation

The next level of sophistication is that of *tool-assisted installations*. In this case, a separate *network installation tool* is required to assist in installing the application. Ideally, such a tool would allow any PC software application (except *network-aware applications*, described below, or applications unlocked by a specific key) to be installed onto networked PCs from a central location. Such a tool would eliminate the need for any interaction at each PC during installation, and would allow a central administrator to determine and control each installation and update.

In contrast to the network-enabled installations, this tool would not simply run the installation or update program. Instead, the PC administrator would use the tool to determine *what exactly the installation or update programdoes*, and replicate it. When integrated with PC configuration information (see "PC Hardware and Software Asset Management," below), the installation could be performed unattended, dynamically using configuration information about each networked PC to determine how to install the product. The tool could even be aware of dependencies, so that Excel could not be installed until Windows was installed.

In this scheme, the time savings (as compared to normal, floppy installation and updates), would be enormous. Installations and updates could be automated to the point that a central administrator need do nothing but control where and when the applications are installed, from a single point on the network. Taking this a step further management tools could be developed such that users could even be logically grouped and managed so that certain groups were allocated one type of software (e.g., "Secretaries get word-processors, accountants get spreadsheets"), and installations and updates applied to entire groups of users.

An added benefit is that the networked PCs now become homogeneous, which aids in their ongoing administration (see "PC Hardware and Software Asset Management," below).

Once again, network traffic can become a problem, but since the installations and updates are being controlled centrally, the problem should be easily avoided. Licensing may still be an issue, unless the tool can manage this, as well (see "PC Software License Management," below).

There are a number of products on the market which focus almost exclusively on ESD [12-15]. These references, and the bibliography referring to "Electronic Software Distribution" provide some pointers.

Network-Aware Applications

The most sophisticated approach is *network-aware applications.* This solution bypasses many of the problems of the first two approaches simply by developing an application which is aware it is being run on a network. While only a few applications are at this stage as of now, as networks enter more and more offices, more popular applications will evolve into this form.

The idea is simple: The application resides on a shared disk, allowing multiple users to access it. User-specific data is often kept on each PC. Installation and updates are greatly simplified, because the application is only installed once, on the shared disk. Examples of such applications are given in [16, 17].

Network traffic is again problem. Software is downloaded across the network into PC memory every time a user *runs* the application, as opposed to only when it is *installed.* Licensing, however, is a non-issue, as the application regulates itself. A side affect of this, however, is that, above a certain number, users will not be able to access the application until someone else has finished. Another key concern is availability of the application, if the server or the network goes down. In the two previous methods, this type of interruption would not cause a problem on a day-to-day basis, as the network is only needed during installation, and not during execution.

PC Software License Management

In any environment, PC software license management is a tricky business. Part of this difficulty arises from the fact that there is a confusing array of software license strategies. The following list contains many common software license approaches.

- *Workstation licenses* - one license for each PC with the application installed.
- *User licenses* - one license for each purchaser of the application.
- *Group licenses* - specified number of licenses for a workgroup.
- *Site licenses* - unlimited use of application at a particular site.
- *Concurrent* or *dynamic licenses* - licenses constrained to the number of copies actually in use at any given moment.
- *Corporate licenses* - unlimited use of an application within a corporation.
- *Unlimited licenses* - unlimited use of an application by anyone (e.g., "shareware" or free software).

As you might imagine, this spectrum of license strategies lends itself to difficult enforcement in the workplace, especially for applications which are widely used, such as spreadsheet or word processing programs. There is a great deal of literature specifically dealing with the plethora of licensing strategies [18-31]. One group, the Micro Managers Association (MMA) has even gone to the extent of publishing a "white paper" on network software licensing [23, 24] in an attempt to add some coherence to the structure.

The main problem with software license management is determining who has what, and ensuring that the licenses exist to satisfy the licensing requirements for this software. For instance, in a workplace where floppies are passed about freely, it may be difficult to determine who does *not* have a particular application. Site and corporate licenses can help address this problem, but may not be the most financially viable solution for smaller workgroups.

With this wide variety of licensing schemes, and the difficulty of tracking software installations, it is clear that the lax or improper management of licenses can lead to illegal copies of software being installed. This problem has become so large that an independent group, the Software Publishers Association (SPA), has been formed to "raid" large corporations with the express purpose of enforcing software licenses [32-37].

As with the installations and updates, networks can ease the task of license management somewhat, given the appropriate tools. With the exception of network-aware applications, however, there is still the issue of purchasing copies for every user, or arranging for group, site, or corporate licenses. Networking aids only in the tracking of these licenses, not in the physical management of the software. (For more on management of the physical aspects, see "PC Hardware and Software Asset Management," below.)

At the one extreme, license management for software installed in a network-enabled manner isn't much different than floppy installation, except that access to the application can be somewhat controlled by restricting access to the shared disk. At the other extreme, network-aware applications are tailored for a networked environment, and regulate themselves using concurrent license schemes [16, 38, 39]. This relieves the administrator from license management beyond the initial purchase.

Just as with installations and updates, tools exist to aid in license management on a network. Some tools use the idea of *LAN metering*, which means metering LAN activity, including applications, as they are run [40]. This is effectively concurrent licensing for applications which are not network-aware.

Another approach is to extend the functionality of the network installation tool described in the last section to also keep track of licenses for the installed software. Installation can again be controlled by restricting access to the software, but this approach further enforces restrictions by actually keeping track of how many copies can be legally installed [41]. In the case of group licenses, for example, software is allocated only to the users who need it, and the license count is set to reflect what has been legally purchased. Every time a user installs the software, the count decrements by one. The license count is increased when users "return" licenses by removing the application when they no longer need it, or by purchasing the right to use more copies.

PC Hardware and Software Asset Management

It is unavoidable to discuss PC software installations and updates, and PC software license management without also discussing PC software auditing. Software auditing, or the practice of *inventorying* PC software, is only one aspect of PC hardware and software asset management. The most general description of asset management is: *To know who has what hardware and software, and where it is located.*

PC hardware and software asset management consists of the following tasks:

- PC software auditing and license control (i.e., who has what software, and is it legal?)
- PC hardware inventory (e.g., how much disk space is available, and what type of monitor?)
- PC configuration management (i.e., physical management of software and hardware)

Asset management is difficult in a standalone environment, because it either requires the diligence of PC users to track their own information, or the effort of PC administrators to collect and maintain the information. Many maintain a database with this information, but it is easily outdated, as new software is installed, or hardware changes hands. Some firms have automated this by using bar-code scanners and other technologies to track the hardware assets, but software assets are difficult to track without performing a full software audit on each PC.

PC Software Auditing and License Control

Difficult as it is, the chore of software auditing is a necessary evil, since most larger companies perform at least annual software audits. As mentioned in the previous section, with the advent of the SPA, the need for accurate PC asset management has become an even higher priority [32-37]. Huge fines have been levied against companies found to have illegal copies of software. The SPA provides a free audit tool, *Spaudit*, for performing your own PC software audits [33].

On a network, tools have been developed to inventory the software (and hardware) on each PC in an automated fashion [42-50]. Those mentioned in the references all run on Novell Netware, which provides the basic network tools to enable the network administrator to execute inventory tools remotely.

Taking this a step further, a network installation/license management tool, as described in the previous section, could be further enhanced to run software audits, and match the results against the licensing information in its database. The added value of license control greatly simplifies PC software asset management: The PC administrator simply runs the audit tool on each PC, and a report is generated showing which PCs have legal software, and what software needs to be licensed. (See "Reporting," below, for more on reporting.)

PC Hardware Inventory

Some of the tools mentioned above also perform hardware inventory, querying each PC to determine what type of monitor, size of hard disks, amount of memory they have. Just as software assets can be managed in a central database, so can hardware assets.

Again, in a standalone environment, tools can be run, but the data must be collected laboriously, from each PC. At least in a networked environment, the tool can be invoked from a central location, and a central database maintained.

PC Configuration Management

The tools described in the last two sections can be automated to "gather" the information. The management of the *physical* assets is not quite as simple. Physical management can refer to anything from the name of the PC user and their Internet Protocol (IP) address, to the asset and serial numbers of their PC. As you might imagine, this information can be very dynamic, and change from site to site.

Keeping this information in a centralized database is the natural choice. By enhancing the network installation/license management tool, this information could naturally feed into the installation and update process, as software is installed in different ways, dependent upon the target system (e.g., installing on a PC with an EGA vs. a VGA monitor). Coupled with license and auditing information, reports could be easily generated to immediately tell the PC administrator who has what, and where it is.

Once PC hardware and software asset management is simplified to this level, the entire task of networked PC management eases substantially. Control over installations and updates, coupled with records of who has what, now makes version control as easy as generating a report to see who has installed or updated to which version of the software and taking action.

As mentioned previously, uniform installations from the network also tend to give a homogeneousness to the environment. That is, on any given PC, the spreadsheet application can always be found in the same directory.

Reporting

All of these areas obviously require the further ability to produce reports. This is especially true of the hardware and software asset management. In a standalone environment, where tools exist for performing these tasks, they usually produce reports that all look different, and are not integrated. If the PC administrator has kept a custom database, reports can be generated using a database tool. Now, they will all at least look the same, but the data may be suspect, without a great deal of effort to keep the data current.

Attaching reporting to the network installation/license management/asset management tool allows integrated reports showing all aspects of the networked PC environment. Once again, the addition of the right network PC management tool greatly simplifies one of the tasks of the administrator.

Networked PC Security

With the addition of networks into an environment, there tends to be a heightened awareness of maintaining security. Since all PCs are now linked, data can be vulnerable, especially if it is on shared disks. Another concern is the proliferation of computer viruses, which can be transmitted over the network (although they can be transmitted just as easily with infected floppies).

Some security products are geared towards standalone PCs. They fall into two general categories: products which mimic security features available in larger systems such as access control and encryption, and products that detect and/or eradicate computer viruses [51].

For networked environments, tools exist with not only these features, but the additional features of monitoring LAN activity. Activities such as file movement and print functions can be monitored [40, 52].

Unlike the other issues of PC administration, the addition of a network does not ease the task of maintaining security. In fact, it may even be more difficult. However, the other benefits of adding a LAN should far outweigh the added administrative costs of security.

CASE STUDY: HP SOFTWARE VENDOR

Keeping the previous discussion in mind, let's evaluate a networked PC management tool which addresses many of these issues: namely, HP Software Vendor. This case study looks at each of the areas of networked PC management, and evaluates HP Software Vendor's approach to aiding in that area.

HP Software Vendor is a full-featured networked PC management tool which addresses aspects of software installation and updates, license management, asset management, and reporting. It was originally introduced in May, 1991 [10, 53, 54], as the successor to HP's NewWave Office Install utility, a tool which allowed system administrators to centrally control the distribution of software to PCs [55]. The name, "Software Vendor," was chosen to describe the product because it acts as a "vending machine." That is, users can look at what software is available for them to "vend," and install it on their PC, across the network.

Software Vendor has two components: The Manager, which is a Windows-based application which runs on the PC administrator's PC; and The Vendor, which is a DOS-based program which resides on each of the networked PCs. Both components access a database, as well as software application files on shared disks. Novell and LAN Manager-compatible networks are supported.

Software Installation and Updates

Software Vendor is a network installation tool. It installs applications in a *tool-assisted* fashion, as described previously. Except for *keyed* and network-aware applications, almost any software application which can be installed on a PC from floppies can be installed over a network using the facilities of Software Vendor.

The approach which is used is straightforward: The application is first installed on the PC administrator's PC. After observing how it was installed (e.g., what directories were built, how CONFIG.SYS and AUTOEXEC.BAT were changed, what information was entered), the PC administrator then writes a Software Vendor *installation script*. With The Manager component, Software Vendor provides a rich scripting language which duplicates everything that an application's installation can do, but in an hands-off fashion [56, 57]. By using PC configuration information in the database (see "Hardware and Software Asset Management," below), the installation script can determine how and where to install certain files, based on the type of hardware. The scripting language can also alter the CONFIG.SYS or AUTOEXEC.BAT files.

The PC administrator controls access to the software by assigning it to users. Again, using the PC configuration database, applications are assigned only to the users or groups of users that need them.

Normally, to install software The Vendor must be run from each PC, and the applications which are to be installed are chosen from a menu [58]. However, Software Vendor can be combined with other tools, such as Intel's LanSight [59], to actually centralize the software installation. In other words, the PC administrator may elect to "push" to software onto each of the networked PCs, so that the installation or update is controlled in its release.

Software License Management

In addition to its installation tools, Software Vendor integrates license management. For every application set up with The Manager, a license count may be set. Counts can range from zero (no one can install it) to unlimited (anyone can install it). Every time the application is "vended," this license count is decremented. Later, when it is "returned," the license count is incremented. Software Vendor does not, however, provide LAN metering, or any form of concurrent licensing.

Coupled with the assignment of software described above, this license management facility makes it very simple both to control who has access to the application on the shared disk, and how many may install it.

Hardware and Software Asset Management

As already mentioned, PC configurations are managed in a central database using The Manager. The information which is tracked is customizable by the PC administrator. This information feeds into both the installation scripts and the reports. This allows the management of the physical aspects of hardware and software.

Although Software Vendor does not yet offer an integrated PC auditing tool, it is possible to perform effective software and hardware audits. Using one or more of the standalone auditing tools described in the references, a report can be generated, which can then be compared to a Software Vendor report using tools such as Microsoft Excel. This comparison effectively generates an exception report which flags inconsistencies: applications or hardware which the audit tool does not recognize; applications or hardware which it recognizes, but are not recorded in Software Vendor; and applications or hardware which are not found, although the configuration database says they should have been.

Reporting

Software Vendor provides the ability to generate a variety of reports including PC hardware asset reports and installed software reports. These may be printed directly, or exported to common spreadsheet formats for further manipulation.

Security

Software Vendor does not really address the issue of security. The fact that it runs on many kinds of networks, using shared disks makes for an "open" architecture. Security of the LAN is left to the NOS.

CONCLUSIONS

HP Software Vendor is a good example of a tool which is tailored for the networked PC environment. Its entire design is geared towards the management of large groups of PCs, where management includes not only the installation and updating of software, but also license control, auditing, and configuration management.

In the references, there are many similar tools. However, almost all of them focus on only one of the areas discussed above. Some are very good at electronic software distribution, but make no attempts at tracking licenses; while others are great at concurrent licensing, but do not pay any attention to asset management.

It is clear, however, from the abundance of recent articles, that this is an idea whose time has come. More and more PCs are being connected via LANs, and PC administrators are finding themselves faced with some or all of these problems.

We are beginning to see an awareness of these issues in popular PC applications, as more and more become network-aware. And, with the growth of groups such as the MMA, and SPA, issues of licensing and auditing are coming to the fore.

Novell, with 70% of the PC LAN market, is certainly leading the way in many areas of network management. Many tools, some referenced in this paper, have been written which take advantage of the Netware Operating System to provide excellent software distribution and inventory tools. As yet, however, no integrated tool, built solely for Netware, has emerged.

HP Software Vendor, more than any other product, has blazed the trail for integrating all the varied aspects of networked PC management. However, if current trends continue, the market is sure to see competition in the not-too-far future.

REFERENCES

While not exhaustive, the following references and bibliography represent a fairly good cross-section regarding the issues discussed in this paper. Please use these references to delve more deeply into the problems of networked PC management, and as guides to even more information.

1. Haight, Timothy, "Stalking The Wild Desktop," *Network Computing*, December, 1991, pp. 60-75.
2. Janusaitis, Bob, "LAN Management," *Computerworld*, January 27, 1992, pp. 91, 93, 98.
3. Pallatto, John, "Managers consider electronic distribution of commercial software; developers say they have cut technological limitations," *PC Week*, May 12, 1987, pg. 39.
4. Webber, Julie, "Electronic distribution offers benefits, problems," *InfoWorld*, December 7, 1988, pg. 43.
5. Hindin, Eric M., "A Simpler Way to Send Software," *Data Communications*, March, 1991, pp. 43-46.
6. Korzeniowski, Paul and Desmond, John, "And Now the Software Delivers the Software," *Software Magazine*, May, 1991, pp. 63-76.
7. Gartner Group, Inc., "The Grim Reality of Software Distribution," Software Management Strategies, Key Issues K-824-1047, September 12, 1991.
8. Raths, David, "Software upgrades are more complex than ever," *InfoWorld*, September 30, 1991, pp. 562-564.
9. Busse, Torsten, "LAN managers struggle with updates," *InfoWorld*, December 2, 1991, pg. 8.
10. Dougherty, Elizabeth, "Many Hands Make Light Work," *LAN Magazine*, March, 1992, pp. 73-80.
11. Busse, Torsten, "Say Goodbye to Snakernet," *InfoWorld*, March 2, 1992, pp. 49, 50, 57.
12. Musich, Paula, "Mainframe Software Manages PC Updates," *PC Week*, September 24, 1990, pg. 12.

13. Musich, Paula, "Insurance Firm Takes Out a New Downsizing Policy: Northwestern Mutual Eases Software Distribution," *PC Week*, March 18, 1991, pg. 13.

14. Horwitt, Elisabeth, "Tool automates PC upgrades," *Computerworld*, June 3, 1991, pg. 66.

15. Horwitt, Elisabeth, "Tangram promises control over distributed PC software," *Computerworld*, March 2, 1992, pg. 48.

16. Eva, Elizabeth and Scott, Karyl, "Software firms move to concurrent licenses," *InfoWorld*, October 14, 1991, pp. 46, 52.

17. Fickel, Louise, "WordPerfect shifts to concurrent-use licensing," *InfoWorld*, October 28, 1991, pp. 1,8.

18. Wilkinson, Stephanie, "Is site licensing a dead issue?," *PC Week*, November 24, 1987, pg. 65.

19. Hwang, Diana, "Software Licensing: Brightwork paper discusses the issues," *Computer Reseller News*, June 3, 1991, pg. 57.

20. Currid, Cheryl, "Software needs easier licensing schemes," *PC Week*, June 3, 1991, pg. 70.

21. Morrissey, Jane, "Piracy Wars Obscure Licensing Dilemmas," *PC Week*, June 27, 1991, pp. 1,6.

22. Eva, Elizabeth, "Network licensing stirs controversy," *InfoWorld*, September 16, 1991, pg. 108.

23. Flynn, Laurie, "MMA tries to make software licensing for LAN users easier," *InfoWorld*, September 30, 1991, pg. 564.

24. Scott, Karyl, "Managers' group takes software license initiative," *InfoWorld*, October 7, 1991, pp. 1,111.

25. Herron, D. Keith and Witt, Joanne T., "The MMA's paper on network software licensing," *InfoWorld*, October 14, 1991, pp. 46-52.

26. Scott, Karyl, "Group offers license guidelines," *InfoWorld*, October 21, 1991, pg. 3.

27. Hwang, Diana, "Consortium Seeks Licensing Accord," *Computer Reseller News*, February 11, 1992, pp. 3, 37.

28. Wasch, Ken, "Concurrent licensing: Publishers face obstacles," *Computer Reseller News*, March 2, 1992, pg. 51.

29. Gainforte, Greg, "Pros and Cons of Purchasing LAN Software," *LAN Times*. March 9, 1992, pp. 43, 44.

30. Scheier, Robert L., "Software Tracking Still Stuck in Low Gear," *PC Week*, March 9, 1992, pg. 88.

31. Herron, D. Keith, and Joanne T. Witt, "LAN Software Licensing Poses Problems," *LAN Times*, April 6, 1992, pp. 45, 46.

32. "SPA offers anti-piracy kit, looks for illegal copies," *MacWEEK*, February 20, 1990, pg. 28.

33. Fitzgerald, Michael, "SPA offers free (audit) software," *Computerworld*, December 10, 1990, pg. 41.

34. "Protect yourself from copyright infringement," *LAN Magazine*, February, 1991, pg. 14.

35. Fitzgerald, Michael, "Open up - this is the software police!," *Computerworld*, June 17, 1991, pp. 1, 92.

36. "Understanding Software Licensing," *PC World*, July, 1991, pg. 65.

37. Wasch, Kenneth A., "Before SPA comes knocking...Clean your corporate house by sweeping out software improprieties," *Computerworld*, July 29, 1991, pg. 80.

38. Nash, Jim, "Microsoft eases LAN license policy," *Computerworld*, June 3, 1991, pg. 136.

39. Fickel, Louise, "WordPerfect shifts to concurrent-use licensing," *InfoWorld*, October 28, 1991, pp. 1,8.

40. Maxwell, Kimberly, "LAN Metering Software," *PC Magazine*, September 11, 1990, pp. 295-319.

41. Wilkinson, Stephanie, "Electronic distribution: a possible solution to the site-license problem," *PC Week*, November 24, 1987, pg. 78.

42. Dryden, Patrick, "Audit tools aid network managers," *LAN Times*, April 15, 1991, pg. 19.

43. Hwang, Diana, "Audit software upgrade: Brightworks to ship LAN Automatic Inventory v.2.0," *Computer Reseller News*, August 26, 1991, pg. 50.

44. Lauriston, Robert, "LAN Hardware and Software Inventory," *PC World*, September, 1991, pg. 142.

45. "Take Your Inventory over the LAN," *BYTE*, February, 1992, pg. 66PC-8.

46. Crowley, Aileen, Inventory Tools Help Track Network Expansion," *PC Week*, March 2, 1992, pp. 83, 86.

47. Crowley, Aileen, "Auditing Tools Keep Watch over Growing LANs," *PC Week*, March 2, 1993, pg. 91.

48. Homer, Blaine, "No-Nonsense LAN Auditor," *LAN Times*, March 23, 1992, pp. 56, 59.

49. Homer, Blaine, "Frye Offers New Program for Inventorying," *LAN Times*, March 23, 1992, pp. 60, 61.

50. Homer, Blaine, "Check Out Check It LAN," *LAN Times*, March 23, 1992, pp. 61, 62.

51. Jackson, Keith, "Security Software," *Which Computer?*, July, 1990, pp. 72-75.

52. "Menuing Programs Provide LAN Security," *PC Week*, July 30, 1990, pp. 91-92.

53. Fickel, Louise, "NewWave Office Upgrade Lets Users Share Data," *InfoWorld*, April 15, 1991, pg. 8.

54. "Electronic Software Distribution Comes of Age," *Data Communications*, August, 1991, pg. 24.

55. "Electronic Software Distribution," *PC Week*, January 12, 1988, pg. C7.

56. *HP Software Vendor Administrator's Guide*, D2506-90003 E0591, Hewlett-Packard Company, 1991.

57. *HP Software Vendor Application Vending Tips*, D2506-90009 E0591, Hewlett-Packard Company, 1991.

58. *Using the Vendor Quick Reference Card*, D2506-90005, Hewlett-Packard Company, 1991.

59. *Intel LanSight Support Reference Guide*, SLAN1225, Intel Corpoation, 1991.

BIBLIOGRAPHY

Network Management

"Audit Trail Software Helps Maximize LANs," *PC Week*, January 22, 1990, pp. 67-74. A thorough review of four products which aid in tracking network activity on a variety of networks.

"Audit Trail Programs Keep LANs in Line," *PC Week*, January 21, 1991, pp. 103-107. An update of the article above, reviewing three network management products.

"Successfully Supporting All Network Users," *LAN Times*, February 10, 1992, pg. 73. A brief (but thorough) overview of the issues of LAN management.

"The Many Strands of Systems Management," *LAN Technology*, March, 1992, pp. 33-42. Another overview of LAN management issues, with a product comparison of 27 products.

Networked PC Administration

"Stalking The Wild Desktop," *Network Computing*, December, 1991, pp. 60-75. Compares networked PC management at six different organizations.

"LAN Management," *Computerworld*, January 27, 1992, pp. 91, 93, 98. Discusses issues of networked PC administration and compares thirteen products.

Electronic Software Distribution

"A Simpler Way to Send Software," *Data Communications*, March, 1991, pp. 43-46. A good overview of the problems of standalone installations and updates, and a look at the vendors in the market at the time.

"And Now the Software Delivers the Software," *Software Magazine*, May, 1991, pp. 63-76. An overview of both ESD and distributed change management systems. Includes an exhaustive product list of products in these two arenas.

"Many Hands Make Light Work," *LAN Magazine*, March 1992, pp. 73-80. Excellent article with actual success stories. Mentions and compares a total of twelve ESD applications.

"Say Goodbye to Sneakernet," *InfoWorld*, March 2, 1992, pp. 49, 50, 57. Another good overview of the problem, with some specific successes cited.

PC Software License Management

"The MMA's white paper on network software licensing," *InfoWorld*, October 14, 1991, pp. 46-52. A complete reproduction of MMA's white paper, describing their proposals for various licensing methodologies.

"SPA offers free (audit) software," *Computerworld*, December 10, 1990, pg. 41. Describes what SPA does, and how to go about ordering a kit to perform a software audit yourself.

"LAN Metering Software," *PC Magazine*, September 11, 1990, pp. 295-319. An extensive and very thorough article evaluating eleven LAN metering products on the merits of dynamic licensing, individual audit, application audit, and LAN security.

"Pros and Cons of Purchasing LAN Software," *LAN Times*, March 9, 1992, pp. 43,44. Compares five different licensing schemes; examines pros and cons of each.

"LAN Software Licensing Poses Problems," *LAN Times*, April 6, 1992, pp. 45, 46. Spells out the problems of various licensing schemes and expands on MMA licensing white paper.

PC Hardware and Software Asset Management

"Audit tools aid network managers," *LAN Times*, April 15, 1991, pg. 19. Evaluates three audit tools which inventory hardware and software in a central database.

"Inventory Tools Help Track Network Expansion," *PC Week*, March 2, 1992, pp. 83, 86 Compares sixteen LAN-based inventory tools.

Networked PC Security

"Menuing Programs Provide LAN Security," *PC Week*, July 30, 1990, pp. 91-92. An evaluation of five network security products.

"Security Software," *Which Computer?*, July, 1990, pp. 72-75. A good evaluation of seventeen PC security products. Has a European slant, as this is a British magazine.

HP Software Vendor

HP Software Vendor Administrator's Guide, D2506-900003 E0591, Hewlett-Packard Company, Palo Alto, California, 1991. The reference volume for the HP Software Vendor product.

Paper #2042

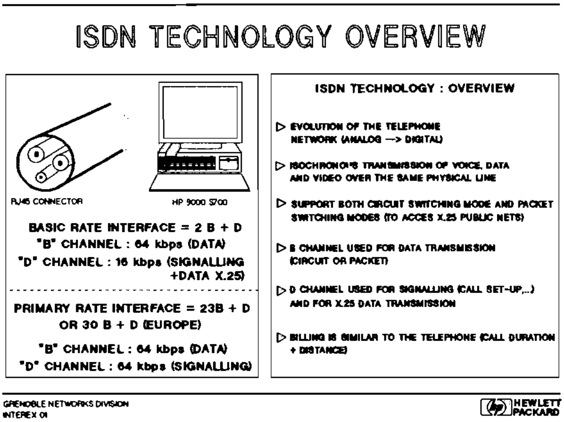
ISDN Applications

by

Eric Grall
Hewlett-Packard
Grenoble Networks Division
5, avenue R. Chanas
38053 Grenoble Cedex 09 - France

1. WHAT IS ISDN ?

ISDN is an evolution of the telephone network. It provides digital transmission of voice, data, and image on the same physical line. The ISDN Basic Rate Interface (BRI) is a line with 2 x 64 kb/s digital channels , named B (Bearer) channels, which can simultaneously transmit voice, user data or images in a circuit-switching mode.



ISDN TECHNOLOGY OVERVIEW

RJ45 CONNECTOR
HP 9000 S700

BASIC RATE INTERFACE = 2 B + D
"B" CHANNEL : 64 kbps (DATA)
"D" CHANNEL : 16 kbps (SIGNALLING +DATA X.25)

PRIMARY RATE INTERFACE = 23B + D OR 30 B + D (EUROPE)
"B" CHANNEL : 64 kbps (DATA)
"D" CHANNEL : 64 kbps (SIGNALLING)

ISDN TECHNOLOGY : OVERVIEW

▷ EVOLUTION OF THE TELEPHONE NETWORK (ANALOG —> DIGITAL)

▷ ISOCHRONOUS TRANSMISSION OF VOICE, DATA AND VIDEO OVER THE SAME PHYSICAL LINE

▷ SUPPORT BOTH CIRCUIT SWITCHING MODE AND PACKET SWITCHING MODES (TO ACCES X.25 PUBLIC NETS)

▷ B CHANNEL USED FOR DATA TRANSMISSION (CIRCUIT OR PACKET)

▷ D CHANNEL USED FOR SIGNALLING (CALL SET-UP,...) AND FOR X.25 DATA TRANSMISSION

▷ BILLING IS SIMILAR TO THE TELEPHONE (CALL DURATION + DISTANCE)

GRENOBLE NETWORKS DIVISION
INTEREX 91

HEWLETT PACKARD

ISDN circuit switching provides users with several advantages:

o Cost-savings for datacommunication (pay for call duration, not for volume), in particular for file transfer over wide area networks.

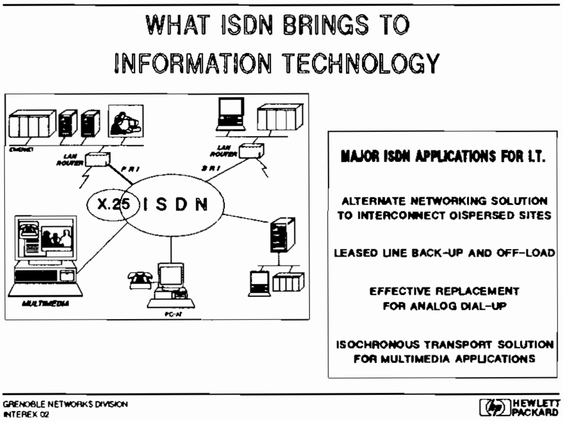
o 64 kb/s reliable data transmission over each ISDN B channel,

o Fast dial-up call establishment,

o Flexible bandwidth allocation by opening one or several 64 kb/s circuits,

o Access security, by identifying callers.

2. WHAT ISDN BRINGS TO INFORMATION TECHNOLOGY



WHAT ISDN BRINGS TO INFORMATION TECHNOLOGY

MAJOR ISDN APPLICATIONS FOR I.T.

ALTERNATE NETWORKING SOLUTION TO INTERCONNECT DISPERSED SITES

LEASED LINE BACK-UP AND OFF-LOAD

EFFECTIVE REPLACEMENT FOR ANALOG DIAL-UP

ISOCHRONOUS TRANSPORT SOLUTION FOR MULTIMEDIA APPLICATIONS

GRENOBLE NETWORKS DIVISION
INTEREX 92

HEWLETT PACKARD

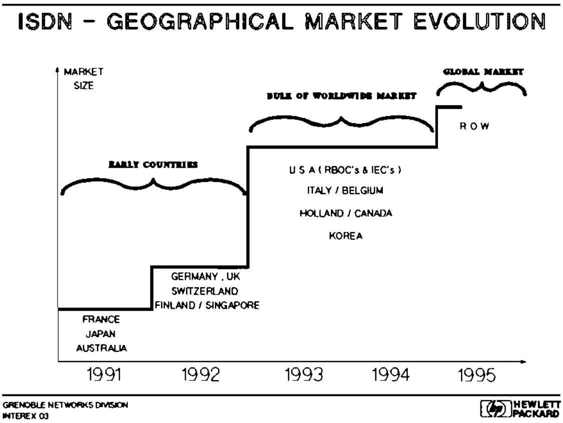
During the late 80's, the Integrated Services Digital Network (ISDN) has been too often incorrectly presented by the news/media and the Telecom Operators as the new, hot, network technology that would soon replace all kinds of established telecommunications. It therefore raised unreasonable expectations.

However, even if ISDN is not the "networking panacea", it is a clearly emerging and growing technology that will fit the needs of many businesses by providing a cost-effective solution to transfer information over wide areas. For example, it is a cost-effective way of connecting multiple, geographically distributed branch offices to central sites when non-permanent exchange of information is necessary.

In the near future, with its embedded X.25 support on both the B and D channels, narrowband ISDN will clearly be a networking alternative available to users in the same way as X.25 services today, or Frame Relay services soon. It will not compete, but be complementary, to future high-speed wide area network services (commonly named SMDS, ATM or Broadband-ISDN) for small to medium communication needs. Also, it will be a support of choice for many of the expected multimedia applications.

3. ISDN : GEOGRAPHICAL MARKET TRENDS

As it appears on the figure below, the first characteristic of the ISDN market is that it is a geographically growing market that should truly become a worldwide mass-market by 1994/1995. The main reason behind this is that ISDN comes together with the digitalization of the telephone switches progressively deployed by the various PTT or Operators on a country per country basis.



ISDN - GEOGRAPHICAL MARKET EVOLUTION

GRENOBLE NETWORKS DIVISION
INTEREX 03

HEWLETT PACKARD

3.1 ISDN in Europe

By January 1991, commercial ISDN had been launched in five European countries : Belgium, Finland, France, Germany and UK. France was the first one to launch the service, followed by Germany. Because of the heavy investments made by the operators in these two countries and because of their "business" size, France and Germany represent the primary geographical market for ISDN product manufacturers in Europe. UK should also come back quickly among the main countries. As an example, France Telecom is announcing about

120,000 installed BRI lines until November 1991 (excluding PRI lines) among which, probably 30% to 40% are used for datacommunication purposes.

Within three years - by January 1994 - commercial ISDN services should have been lauched in almost all the European countries.

So far, the implementation of ISDN has differed between countries. This has led to incompatibilities between services in different countries. However, in an effort to overcome the problems, progress is being made towards the concept of a common European ISDN implementation, so-called "Euro-ISDN", and based on the ETSI standard. It should be implemented within all EC countries and between them by the end of 1993, more probably by 1994.

3.2 ISDN in Asia-Pacific

Asia-Pacific is probably the second most important region of the world, as far as commercial ISDN services are concerned.

Furhermore, Japan is viewed as a leader in providing ISDN services. It is actually a result of the dismantling of Japan's domestic and international telecommunication monopolies in 1985. NTT is the dominant domestic carrier. NTT launched its basic rate ISDN services, "INS-Net 64", in 1988 and primary rate ISDN, "INS-Net 1500", in 1989. National coverage is targeted for 1995. The number of INS-Net 64 customers increased from 6,000 to 19,000 in 1990. NTT forecasts that it will have 20 million ISDN customers by 2005. Due to the attractive ISDN tariffs from NTT, ISDN is becoming more and more a lock-out specification for computer manufacturers targeting the Japanese market.

Also emerging in the Pacific Rim is Australia with commercial ISDN services launched in 1990. It is a cost-effective alternative to the expensive leased lines in this vast country. Then come countries like Singapore and Korea which are expected to "join the club" by 1993/1994.

3.3 ISDN in USA

Today, in the USA, the large installed base of ISDN basic access is made up mostly of ISDN Centrex. (Centrex is a service whereby local switching is provided by the public exchange. It provides digital PBX functionality via a public exchange).

One problem is that the use of ISDN is confined to "islands" around individual exchanges. Until Signalling System Number 7 is implemented between Inter Exchange Carriers and Regional Bell Operating Companies, it is likely that commercial ISDN will not ramp-up significantly in the US. The Bellcore's "National ISDN-1" programme is however gathering support from telcos and equipment vendors under the pressure of users and should help in accelerating the process of unification with actual results expected in 1993/1994.

Another element to integrate for the US ISDN market is the need for 56 kbps support since it is unlikely that many of the existing circuits between public exchanges are to be upgraded by RBOC's for a few years due to the heavy related costs.

Nevertheless, some niche markets exist today for ISDN in the USA and directly relate to early adoption of ISDN by some corporations or entities like McDonalds, Lawrence Livermore Labs, Mc Donnel Douglas, ... and the necessary investments by RBOCs like PACBELL, AMERITECH, BELLSOUTH, ...

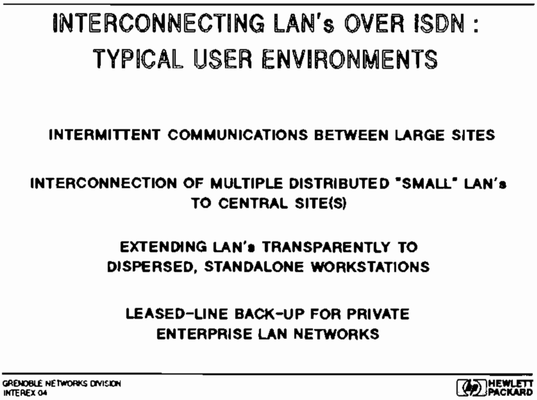
4. TYPICAL APPLICATIONS OF ISDN FOR DATACOMMUNICATION

The figures below outline some real-life examples of networking applications in various environments. As mentioned in all examples, the key driving factors to implement ISDN for users are :

1. Cost savings : in all these environment ISDN allows for cost-savings compared to other alternate technologies such as X.25, leased lines, ...

2. 64 kb/s reliable digital datatransmission

3. Bandwith on demand through use of N x 64 kb/s B channels



INTERCONNECTING LAN's OVER ISDN :
TYPICAL USER ENVIRONMENTS

INTERMITTENT COMMUNICATIONS BETWEEN LARGE SITES

INTERCONNECTION OF MULTIPLE DISTRIBUTED "SMALL" LAN's
TO CENTRAL SITE(S)

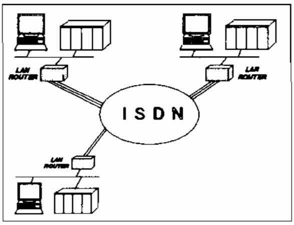
EXTENDING LAN's TRANSPARENTLY TO
DISPERSED, STANDALONE WORKSTATIONS

LEASED-LINE BACK-UP FOR PRIVATE
ENTERPRISE LAN NETWORKS

GRENOBLE NETWORKS DIVISION
INTEREX 04

HEWLETT PACKARD

INTERMITTENT COMMUNICATIONS BETWEEN LARGE SITES



INDUSTRY :	NUMBER OF SITES :
RESEARCH - ELECTRICITY	3 (FRANCE)

ISDN USER NEED

TRANSFERING LARGE, SCIENTIFIC FILES BETWEEN ENGINEERING CENTERS

MAJOR ISDN BENEFITS

- ▶ ENHANCED PERFORMANCE AT A LOWER COST THAN 64 KB/S LEASED LINE
- ▶ NO CHANGE TO EXISTING APPLICATION ENVIRONMENT (TCP/IP)

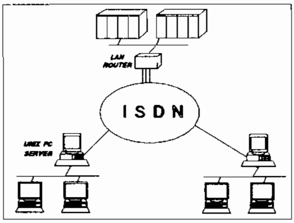
TECHNICAL CHALLENGES

- ▶ ACCESS LINE MANAGEMENT ==> PREEMPTION TIMERS
- ▶ ACCESS TO ISDN THROUGH PBX's
- ▶ N X 64 KBPS TRANSMISSION STABILITY ON PUBLIC NETWORK (N>=6)

GRENOBLE NETWORKS DIVISION
INTEREX 05

HEWLETT PACKARD

INTERCONNECTION OF MULTIPLE, DISPERSED SMALL LAN's TO A CENTRAL SITE



INDUSTRY :	NUMBER OF SITES :
PUBLISHING	150 (FRANCE)

ISDN USER NEED

FILE/IMAGES DISTRIBUTION FROM CENTRAL SITE TO DISPERSED AGENCIES

MAJOR ISDN BENEFITS

- ▶ LOWER COST THAN LEASED LINES OR X.25
- ▶ FASTER THAN ANALOG DIAL-UP

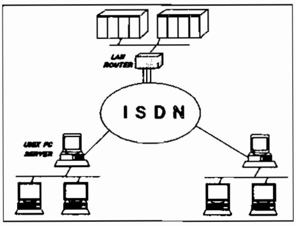
TECHNICAL CHALLENGES

- ▶ ACCESS LINE MANAGEMENT ==> INACTIVITY TIMERS
- ▶ ISDN ROUTING IN LAN PC-SERVER
- ▶ NETWORK MANAGEMENT OVER DIAL-UP NETWORK ISDN

GRENOBLE NETWORKS DIVISION
INTEREX 06

HEWLETT PACKARD

INTERCONNECTION OF MULTIPLE, DISPERSED SMALL LAN's TO A CENTRAL SITE



INDUSTRY :	NUMBER OF SITES :
PUBLISHING	150 (FRANCE)

ISDN USER NEED

FILE/IMAGES DISTRIBUTION FROM CENTRAL SITE TO DISPERSED AGENCIES

MAJOR ISDN BENEFITS

- ▶ LOWER COST THAN LEASED LINES OR X.25
- ▶ FASTER THAN ANALOG DIAL-UP

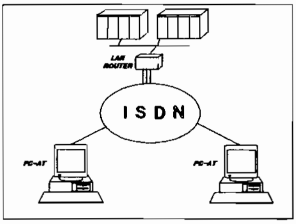
TECHNICAL CHALLENGES

- ▶ ACCESS LINE MANAGEMENT
  ==> INACTIVITY TIMERS
- ▶ ISDN ROUTING IN LAN PC-SERVER
- ▶ NETWORK MANAGEMENT
  OVER DIAL-UP NETWORK ISDN

GRENOBLE NETWORKS DIVISION
INTEREX 06

HEWLETT PACKARD

EXTENDING LAN's TO DISPERSED STANDALONE WORKSTATIONS



INDUSTRY :	NUMBER OF SITES :
FOOD / RESTAURANTS	100 (USA)

ISDN USER NEED

WORK-AT-HOME FOR EMPLOYEES

MAJOR ISDN BENEFITS

- ▶ TRANSPARENT ACCESS TO CENTRAL LAN's
- ▶ FASTER THAN ANALOG DIAL-UP

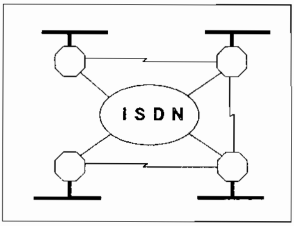
TECHNICAL CHALLENGES

- ▶ ACCESS SECURITY
- ▶ 56 KB/S SUPPORT FOR SWITCHED LINES
- ▶ MAXIMUM NUMBER OF CONCURRENT ACCESS

GRENOBLE NETWORKS DIVISION
INTEREX 07

HEWLETT PACKARD

LEASED LINE BACK-UP FOR PRIVATE NETWORKS



INDUSTRY :	NUMBER OF SITES :
MANUFACTURING	10 (FRANCE)

ISDN USER NEED

BACK-UP OF 64 KB/S LEASED LINES

MAJOR ISDN BENEFITS

- ▶ LOW-COST, PERFORMANT SOLUTION VS DIAL-UP/PUBLIC X.25
- ▶ FASTER THAN ANALOG DIAL-UP

TECHNICAL CHALLENGES

- ▶ BACK-UP MANAGEMENT (ALARMS, ...)
- ▶ ISDN NATIVE SUPPORT IN X.25 SWITCHES & ROUTERS
- ▶ INTERNATIONALIZATION OF SOLUTION

GRENOBLE NETWORKS DIVISION
INTEREX 08

HEWLETT PACKARD

5. POSITIONING ISDN WITH ALTERNATE NETWORKING SOLUTIONS

As mentioned before, ISDN is not the panacea and unique networking solution for wide area communications.

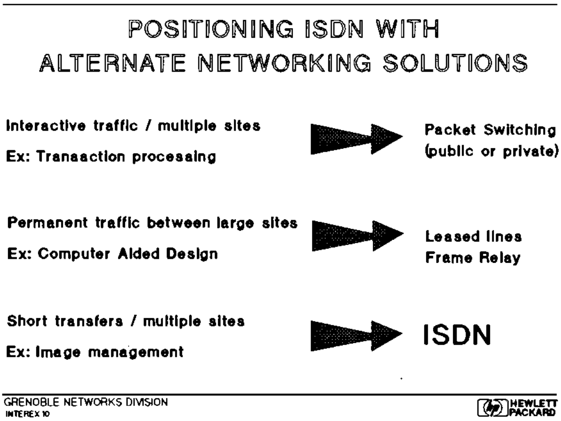
Below is a summary of the networking alternatives that may compete with ISDN-based solutions :

o X.25 : X.25 public and private networks have proliferated in the 80's, in Europe and Far-East mainly, and, to a lesser extent in the US. In fact, X.25 was the first wide area networking standard to be defined and actually implemented by PTTs, carriers and computer manufacturers. It has been largely used and deployed by small to large companies and will continue to prevail because of its reliability and proven technology image. It is however recognized now as limited in terms of performance (64 kbps throughput) and is not considered as an efficient mean of interconnecting bandwith-demanding LANs. Despite the fact that access to public ISDN networks will include access to X.25 services and connectivity between X.25 and ISDN devices, it is highly probable that X.25 will remain in the market for the next 3 to 5 years.

o Frame Relay : Frame Relay can be considered as a natural evolution of X.25, with simplified protocol handling and increased performance. It is based on the assumption that, due to the proliferation of digital lines in the major countries of the world, the need for transmission control and error-checking in the networks was no-longer necessary and could be left to the end-CPE (ie computers or access routers). X.25 uses the first 3 levels of the OSI/ISO model while Frame Relay is limited to the first two levels. Even if Frame Relay is not, yet, a CCITT standard, most of its procedures and implementation rules have been commonly defined by the ANSI committee and the Frame Relay Forum (which includes most of the network equipment vendors that provide, or will support, Frame Relay services). Because of its increased performance compared to X.25 (Frame Relay runs today from 64 kbps up to 2 Mbps), it is a good alternative to interconnect LANs and computers over wide areas. Carriers like Sprint, MCI, WilTel, BT, Cable and Wireless, and CompuServe have committed themselves to providing public frame relay services. But the slow supply of public services by these carriers today is currently limiting frame relay deployment. In Europe, only BT, France Telecom and the Finnish PTT's have committed themselves to offer public services.

o Routers and Leased Lines : with LAN proliferation came the need to interconnect LANs through bridges first and then, increasingly, through multiprotocol routers. Routers interconnect various types of LANs (using various technologies and protocols) over wide areas through leased lines or X.25 networks today. Soon they will support frame relay and BRI & PRI ISDN interfaces for wide area links (CISCO has announced ISDN support, 3COM already has a native BRI interface). As such, they will provide an alternative for interconnecting computers and LANs together.

The following figure gives an overall positioning of all these networking technologies based on the type of datacommunication traffic involed. Of course, it only gives an overview. In every real-life environment, one should analyse in details its datacommunication needs in order to select the technology that will fit its needs from both a cost and performance standpoint.



POSITIONING ISDN WITH
ALTERNATE NETWORKING SOLUTIONS

Interactive traffic / multiple sites
Ex: Transaction processing
→ Packet Switching (public or private)

Permanent traffic between large sites
Ex: Computer Aided Design
→ Leased lines
Frame Relay

Short transfers / multiple sites
Ex: Image management
→ ISDN

GRENOBLE NETWORKS DIVISION
INTEREX 10

HEWLETT PACKARD

SCSI -- A view from both ends of the cable

Michael Rusnack
Hewlett-Packard Company
Disk Memory Division
11413 Chinden Blvd.
Boise, ID 83704
(208) 323-2054

Introduction

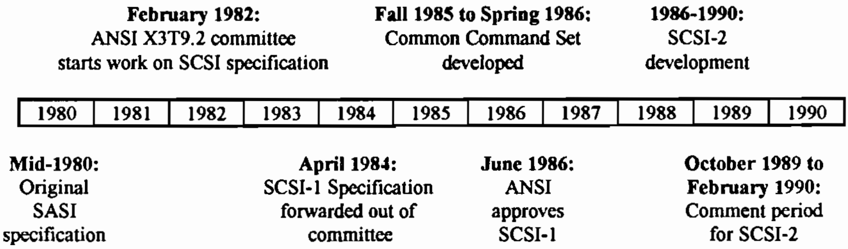
The *Small Computer Systems Interface* (SCSI) is rapidly gaining in popularity as the interface of choice for computer system developers. As a disk drive interface, the intent was for application on small computers. Over the years this interface has grown in popularity among system developers. Companies have pooled resources in the development of today's SCSI specification. Emerging from these standards, many workstation manufacturers adopted SCSI as the peripheral interface of choice. Most recently, it was introduced for use on the Multi-User (mini-computer) systems.

The use of this interface has many implications to the end user. This presentation discusses those implications by first presenting a short history from its inception to the present day. Next, this presentation will discuss terminology and features that define SCSI devices. Included in this discussion are how these features can impact the system implementation. Third, a review of device/system features, and their possible incompatibilities. Fourth, the SCSI interface is compared with other peripheral interfaces in an effort to expand the understanding of the reader. Finally, the presentation briefly covers future implementations of the SCSI interface.

A Short History

The interface standard finds its roots in the 1960s where it began as an IBM mainframe interface. IBM 360s had a byte-wide, parallel I/O bus that could do fast block transfers to and from peripheral devices. Then, this bus was called the Selector Channel, subsequently known as the IBM OEM-channel. After some political wrangling, this parallel I/O bus evolved in the ANSI committees as the intelligent peripheral interface (IPI). Workstation users may be familiar with this bus. At about the same time, Shugart Associates envisioned the need for a flexible I/O bus. The result of this work was known as the Shugart Associates system interface (SASI). When the proponents of SASI approached ANSI with the suggestion that this interface specification become adopted as a standard, they learned that they were competing with IPI, another high-level interface. Furthermore, they discovered that there was a third contender called the intelligent system interface (ISI). Rather than being a third contender for the high-end specification, SASI proponents opted to work with the standards committee that dealt with low-level interfaces, and called the new standard SCSI to set it apart from the others.

All of this work was taking place in the early 1980's. The interface was finally adopted as an Interface Standard by the ANSI committee in 1984. SCSI and its successor SCSI-2 have most of the capabilities of the IPI and a few it lacked. Whether it was politics, fate or just luck, even before the SCSI specification was finalized in 1986, it was more widely accepted than the IPI.



February 1982: ANSI X3T9.2 committee starts work on SCSI specification

Fall 1985 to Spring 1986: Common Command Set developed

1986-1990: SCSI-2 development

1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990

Mid-1980: Original SASI specification

April 1984: SCSI-1 Specification forwarded out of committee

June 1986: ANSI approves SCSI-1

October 1989 to February 1990: Comment period for SCSI-2

Despite its existence for over 8 years, relatively few people are aware of this interface. In an informal poll, MIS' with an average 15-20 years of computing experience, had not heard of SCSI until as recently as two years ago. This is a clear indication of how rapidly this interface is taking the industry by storm. The SCSI interface was initially intended for the Small Computer or PC user. As it has evolved, it eventually found use in the Workstation Market, and today is found on some mini-computer/servers. This is seemingly a big step for what was initially developed as a Small Computer Systems Interface.

An Overview

The SCSI bus comes in two different forms: *single-ended* SCSI, in which the signal's logic level is determined by a voltage relative to a common ground reference, whereas in *differential* SCSI, the signal is the potential difference between two wires. The signal transmission on the *differential* bus is much more robust and less susceptible to electrical noise.

The SCSI specification asserts that the *single-ended* cable can be no more than six meters long. This cable length not only includes the cable connection from the host to the drives, it includes any cables internal to the cabinets. A *differential* cable can be up to twenty-five meters long. *Differential* and *single-ended* devices may not be connected onto the same bus. Besides signal immunity and cable length, today's *differential* SCSI devices are faster in transfer rate.

Delving deeper into the SCSI standards, one will learn that there are variations on these variations. Both the *single-ended* and *differential* buses are available in 8 and 16 bit data lines. These buses are often referred to as *narrow* and *wide*. Expanding further on the jargon of the interface, the *differential* interface is sometimes referred to as *fast*. Thus when discussing a 16 bit -- *differential* bus, it is often called *Fast-Wide.*

The variations of the buses and their characteristics are:

	Narrow /Wide	Cable Length	Total Devices	Transfer Rate
Single-ended	N	6 m	7	5 MB/sec
	W	6 m	15	10 MB/sec
Differential	N	25m	7	10 MB/sec
	W	25	15	20 MB/sec

The evolution of the SCSI-1 standard continued with the advent SCSI-2. SCSI-2 defines features that are very common in high performance environments. Some of these devices include:

- Complete compatibility with SCSI-1
- Transfer rates of 10MB/second on 8-bit SCSI bus
- 16- and 32-bit bus specifications that will ultimately offer 40 MB/second
- Device level error recovery to free the CPU for other tasks
- Command queuing will let intelligent devices improve their own performance

Hard disk drives are not the only SCSI devices available for use on systems with this interface. SCSI devices available today includes:

- Floppy disk drives
- Hard disk drives
- Tape drives
  - ¼ inch cartridge
  - 4-mm Digital Audio Tape (DAT)
  - 8-mm
- Optical drives
  - CD-ROM
  - Read/write
  - WORM
- Printers
- Scanners

An advantage of SCSI devices is the fact that multiple types can be connected to the same bus. Though there are restrictions, for the most part, the standard will allow the collection of disk drives, tapes and optical devices -- not only on the same bus, but in the same enclosure.

Though not completely, I have described the SCSI bus, along with some of its characteristics. To date, numerous papers and articles have been dedicated to the technical execution of SCSI as well as its impact on system development. If a more in-depth study is desired, I would suggest one or more of these articles.

SCSI -- Compatible or Maybe Not

SCSI devices are becoming more and more pervasive as a standard for interfacing different vendors' to a wide variety of peripheral devices. Chances are that if you own an HP or Sun workstation, a Macintosh, a NeXT cube, an Atari ST or an Amiga with a hard disk drive you may be already using SCSI. With these computer manufacturers and more adopting SCSI as the interface of choice for hard disk drives (and other peripherals), you might think that just about any disk drive would work on any computer system -- NOT SO. This lack of compatibility is despite the standards and carefully documented by ANSI in the SCSI-2 Standard and common command set (CCS). The implementation of the SCSI command set and its features are at the discretion of the developer.

This interpretation of the SCSI specification allows each developer of a system -- system interface to peripheral, has creative license. The interpretation allows the developer to design a system that is just slightly different from the next. This differentiation may offer an edge over a competitor's system. An example is *read-ahead cache*. This disk drive will read a "block A" as commanded by the host, then independent of host request will fill its buffer with the next several blocks. If the next read command from the host is for "block B" -- the data is immediately available.

Another example is a tuned system approach. The developer will oversee the integration and development of all pieces of the puzzle. In general, there are three major pieces to the system: host driver, system interface and peripheral. By using the products specified by the developer, the system is optimized. That is the best overall performance is achieved. Another reason to use the *recommended and supported* peripherals you as an end-user is assured that the configurations are tested.

Remember, not all disk drives are the same. As a purchaser and user of these systems, you are often forced to make trade-off; balancing price with performance. Again caution is advised. HP and other makers of computing systems offer several flavors of their drives. In essence, all drives appear to work--so what is the difference?

- Performance -- it appears to work. Is that good enough?
- Bootable device -- does the system recognize this as a *boot device*?
- Feature set -- the system expects to negotiate for a feature and commands it.
- Connector -- does the drive have the correct I/O connector?
- Power fail support -- the drive completes sector writes during power fail.

These are just a few questions that could be asked. It is my personal experience that even with in Hewlett Packard, the same 1.3 GByte disk drives achieve their personality from the firmware. The firmware in this or any other vendor's drive determines the characteristics and feature set. My best advise is to consult the systems sales and configuration guide -- otherwise **caveat emptier**.

SCSI versus Other Interfaces

Thus far, one might begin to wonder how the world ever survived without SCSI? Well, it has, and in many cases -- it continues. In the world of workstations, SCSI has been a standard interface for several years. Today, migration of this evolving interface is seen on mini-computers. Computing systems that in the past that have only used proprietary peripheral interfaces. In HP's case, HP-IB (parallel-copper wire) and more recently HP-FL (fiber channel) are two examples. Other companies like DEC and IBM offer multiple interfaces, including SCSI.

The reason SCSI is emerging as the interface standard on many computer systems is cost. The cost saving is realized by both the developer and end-user. The developer now has a standard to develop against. Of course, this is less expensive, with the savings being passed on to the buyer. If SCSI were so good then, why do companies continue to offer a second choice -- their proprietary interface?

As an R&D project manager, I had the unusual experience to partake in a performance comparison. SCSI and FL specifications were compared from the data sheet. The specification compared was -- *performance.* From the data sheet, 4- and 5 MBytes/sec transfer rates were noted for FL and SCSI interfaces respectively. With this data, it appeared a sure bet that SCSI drives would out perform the FL hardware. When the results were published, the opposite was observed. The FL drives bettered the SCSI consistently by 10+%. The benchmark was performed on identical systems, except for the disk drive interface. The SCSI interface required more direct control by the CPU, whereas the *proprietary* interface operated much more independent of the CPU. This experiment showed that SCSI works; it may be simple and low cost. However where speed and raw performance are necessary, it may not fit the entire bill.

Looking Ahead

SCSI-2 came about because several manufacturers wanted to increase the mandatory requirements of SCSI and define features for direct access devices. While maintain compatibility with SCSI-1, SCSI-2 far exceeds the original standard. This coupled with the common command set (CCS), the SCSI standard was extended to all device types, adding caching commands, performance enhancement features and other worthwhile functions.

Two significant options offered in SCSI-2 was *wide* SCSI (up to 32 bits wide) and *fast* SCSI (synchronous transfers up to 10 MBytes/second). The combined effect of these two features alone could result burst transfers of 40 MBytes/second. *Fast-Wide* disk drives are expected to be available this Fall. These devices (16 bit wide) are capable of transferring at 20 MBytes/second. These devices will exceed the performance capabilities of the proprietary interface noted above.

Eventually, the SCSI standard will evolve to a fiber channel--capable of up to 100 MBytes/second. SCSI *fiber* is part of the SCSI-3 specification. It is not known when the first products will be available, however there is sufficient room for growth in performance and feature set with the SCSI-2 definition until then.

In Summary

Has SCSI met its promise and lived up to its early expectations? The initial goal was to offer multiple plug-and-play compatible types of peripherals on the same physical bus. Today it is the host adapter (SCSI interface for the computer) that is a considerable part of the equation. If the operating system drivers for that particular interface are developed the goal may be met. Another facet of the equation is the disk drive. Not all SCSI devices are created equal. Each has its own personality. The plug-and-play intention of the SCSI standard is most successful when using supported devices. Rather than gambling on loss of data, functionality or system down time, use the recommended peripherals.

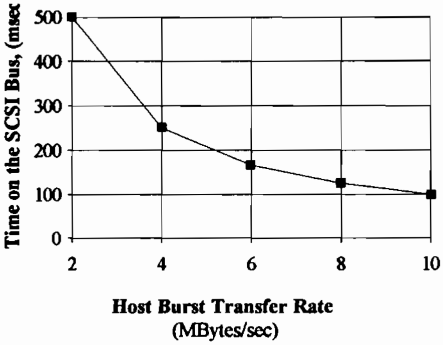
Earlier, I discussed the various attributes of the peripheral interface -- *single-ended* and *differential.* In conclusion, I wanted to follow up on this discussion. The *single-ended* interface, by far the simplest and most versatile of the two has advantages and disadvantages. A real advantage is the connectivity. In the ideal plug-and-play world of SCSI, multiple device types may be housed on a single enclosure. A typical configuration might be:

TAPE BACK-UP
CD-ROM
MULTIPLE HARD DISK DRIVES

The flexibility of these devices affords the user to put up to seven devices in a single enclosure (and/or on a single bus). This reduces cost and increases reliability with the elimination of extra enclosures and power supplies. The price however is in the cable length constraint. The maximum length of cable from the host driver electronics to the final termination is six meters (or approximately 18 feet). When combining several devices/enclosures together, six meters is not very long.

Another short coming of *single-ended* SCSI it transfer speed. This is the time it takes to transfer the data from the peripheral to the host or vice versa. This is important when systems are moving vast amounts of data back and forth. A *single-ended* device can burst transfer data at up to 5 MBytes/second.

From the following chart, it can be observed how transfer rate can affect performance. The faster the transfer rate, the less time spent on the data bus for a fixed amount of data moved. This performance bottleneck is analogous to traffic on a congested freeway. The faster the traffic moves, the more traffic that is allowed to pass, likewise with the SCSI bus. The faster interface -- *differential* SCSI is faster, thus allowing more data to be exchanged.



As you may well have observed, the selection of a SCSI peripheral is no simple task. The application, hardware and software all must be brought together, with an understanding of how the system is to play together. This can only be accomplished through careful planning and development. Even though the *plug'n play* dreams of the first SCSI specification may not yet be realized, you the end user continues to benefit. Development is leveraged from device to device -- platform to platform. In general, you are offered storage devices at a lower cost and performance point that exceeds the rate projected just five years ago.

Conclusion

Now that you have experienced a non-technical overview of SCSI, what does it mean? By no way are you an expert. Engineers with technical degrees spend hours in seminars to cover the technical nuances of this evolving specification. The intent however is to provide you with a background; a greater understanding of an interface that will most likely be offered on your next new computer purchase or even offered as an upgrade in the future.

Paper # 2044
Performance Management Coexistence
Living with MPE/iX and HP-UX
What do I need to know ?

Rex Backman
Hewlett-Packard Company
8050 Foothills Blvd. MS: R10IF
Roseville, Calif. 95678
USA
(916) 785-5638

The increase in the use of the Unix operating system over the past few years has presented us with new challenges in the performance management discipline. Experienced with MPE/iX, we have had to learn the nuances of the HP-UX environment to properly manage the resources of our new Unix hardware investment. Both systems provide the capability to service our customers in a transparent fashion. However, from a performance management perspective, the differences are not as clear. The objective of this article is to address the challenges and issues with managing performance in a "mixed-mode" environment such as this. We will review performance concepts associated with each operating system (MPE/iX and HP-UX), provide suggestions, and discuss ideas on how to properly manage performance in this type of environment.

As System Managers or System Administrators we have seen an incredible amount of change over the past few years. On the MPE/iX side, we have seen regular increases in available hardware speed with equitable decreases in physical size. Operating system functionality has also improved from MIT to MIT. The same holds true of the HP-UX environment. New hardware boxes yield more power. Newer HP-UX operating system releases provide a richer set of commands. All of this change in the operating system environments challenges the task of performance management. A mixed mode environment further compounds the difficulty due the inherent differences in the two operating systems and the tasks associated with each. So, the problem that exists in this mixed-mode environment as it relates to performance management is one of inconsistency. The solution, or the goal of System Managers/Administrators is consistency in their performance management program. The challenge is the work required to define and implement a performance management program that overcomes the differences of the disparate operating systems by providing consistent methods of analysis and reporting.

Lets first review the basic performance concepts associated with each of these operating systems. In these reviews we will discuss the traditional performance bottlenecks, metrics used to help identify them, and provide suggestions on how to resolve them,

MPE/iX Performance Review

Performance management on MPE/iX systems is broken down into the four main potential bottleneck areas: **CPU**, **Memory**, **Software Locks and Latches**, and **Disk**. Our review will touch on the performance metrics that a System Manager should monitor when trying to identify performance bottlenecks.

Each of these potential bottleneck areas has symptoms that can be identified if we key in on the proper metrics. Additionally, the past few years of it's development and use in the user community has yielded a distinct personality in terms of performance characteristics. Generally speaking, the potential main bottleneck with MPE/iX is the CPU, followed by Memory, Software Locks, and then Disk. Due to the design of the operating system, Disk bottlenecks are **<u>very rare</u>** on MPE/iX.

A CPU bottleneck will manifest itself in many ways. From a end-user perspective response time will slow down. Other indicators are decreases in throughput which can be identified as batch jobs running longer than normal.

Technically, the System Manager can identify a CPU bottleneck condition by keying in on the fields defined in the following table:

CPU Bottleneck Identifiers for MPE/iX

User CPU %	Greater than 70%
Overall Switch Rate	Exceeds System Limits
Compatibility Mode %	Exceeds 20%
Current Ready Queue	Averages more than 15

The first metric defines the workload associated with user processing and its' impact on the CPU. Note that operating system impact is not included in this measurement. The 70% threshold is a guideline and should be considered important only if the value is one that is consistently seen over extended periods of time.

Switch rates refer to the process of changing modes of executing processes back and forth between "Compatibility Mode" and "Native Mode". A high switch rate will cause the CPU to expend resources on this overhead activity instead of concentrating on user CPU demands. The threshold for switch rates is machine size dependent as the following graph illustrates.

MPE/iX Switch Rate Guidelines

Series 922, Series 925	150 per second
Series 932	240 per second
Series 935, Series 950	300 per second
Series 949	550 per second
Series 955, Series 960	450 to 600 per second
Series 980	1000 > per second

Compatibility Mode % denotes the time spent in programs or processes that are executing in compatibility mode. Application performance is best when done in "Native Mode", but due to various reasons, some applications utilize compatibility mode. This holds true even for certain portions of the operating system even today (though, this area has decreased dramatically). The guideline for compatibility mode utilization on MPE/iX is **20%**. A sustained rate that surpasses this threshold may be an indicator of a possible CPU bottleneck.

The final primary indicator for the CPU is the Ready Queue, sometimes referred to as the Dispatcher Queue. The Ready Queue, which is pretty much self-defining, is a operating system structure that contains a "list" of processes that are ready to run. These processes are waiting for the CPU. A Ready Queue value that is consistently high may indicate that the CPU can't keep up with the demands of the applications. The guideline for the Ready Queue is 15. Machine size also plays a determining factor here. On the larger machines such as the S/980s, a Ready Queue of 15 may not be cause for concern. Again, the Ready Queue value under investigation needs to be a running average over time.

Memory is the second most frequent performance bottleneck on MPE/iX. With the large sizes of physical memory available, people moving over from the MPE/V (or "classic") architecture find this fact hard to believe. But due to the RISC architecture and operating system features such as the Transaction Manager, MPE/iX will experience memory pressure if physical memory is under configured. The main indicators for memory pressure for MPE/iX are listed in the following table.

MPE/iX Memory Bottleneck Indicators

Memory Manager CPU %	Consistently > 8%
Memory Manager Clock Cycle	Greater than 25 per hour
Overall Page Fault Rate	Greater than 30/second

Memory CPU utilization is the primary indicator here. A value that is consistently greater than 8% may be indicative of a machine that is under configured for memory.

Clock cycle rate defines how often the memory manager "cycles" through memory while it searches for free pages. If it cycles through memory more than 25 times **in an hour** then memory pressure may be a possibility .

Overall page fault rate indicates the number of page faults per second. A page fault occurs when a process wishes to access data in a page and that page is not in physical main memory. This metric should be considered as a secondary metric when evaluating memory pressure. The previous metrics, memory manager CPU % and the clock cycle rate are the primary indicators.

Software Locks and Latches is a bottleneck condition that is usually application oriented. Locks and latches are implemented in applications to insure data integrity. An example is TurboImage locking. Here various methods are available to insure data integrity such as data base or data item locking and conditional versus unconditional locking. Depending on design, application performance could be impacted. Careful design considerations must be made to optimally use software locking. If done incorrectly applications will be found to be impeded quite often.

Historically the Disk component of computer systems has been a possible bottleneck area. Due to design choices HP has almost all but removed Disk as a bottleneck on MPE/iX machines. Instead of seeing Disk bottlenecks we see CPU and/or memory bottlenecks. There are several factors for this change such as the large memory sizes, the MPE/iX Transaction Manager, Memory Manager pre-fetching algorithms.

This does not mean that Disk I/O should not be looked at. Any time Disk I/O can be eliminated, applications have the opportunity to execute faster due in part to the removal of the mechanical linkage to Disk I/O. For MPE/iX, a couple of indicators to keep in mind are the physical I/O rate and the amount of time that the system is paused for I/O. The threshold for Disk I/O rate starts at 30 per second. Due to the range of sizes in the MPE/iX family, this should be used as a starting point. Overall system paused for disk should be lower than 25%. Paused for disk reports how much time the system has had to wait (or "pause") while disk I/O completed and when compared to the speed of other subsystems such as CPU and memory, it is quite slow. For this reason, it is wise to eliminate disk I/O even though it is not commonly found as a bottleneck on MPE/iX machines.

HP-UX Performance Review

The basics such as **CPU**, **Disk**, and **Memory** apply to HP-UX. But due to the differences in the way Unix machines are used we need to concern ourselves with some additional potential bottleneck areas. These areas are: **Kernel Resources**, **Graphics**, and **Networking**. These three new items plus the traditional CPU, Disk, and Memory are the bottleneck areas HP-UX System Administrators need to be concerned with.

HP-UX, due to its' different types of hardware platforms provides additional challenges in performance management. HP-UX machines can be used as stand alone machines, multi-user platforms, or configured in a client/server environments. Configurations such as these make for a radically different environment when compared with the MPE/iX environment. System Administrators need to be aware of how their hardware is configured as these configuration types can create unique demands on system resources.

CPU pressure on HP-UX machines can be identified by monitoring certain metrics which have been defined in the following table.

CPU Bottleneck Indicators for HP-UX

Idle Time	No available idle time
User Mode	High level of user activity
Run Queue	Large Run Queue value
Priority Blocks	Processes frequently are blocked on "Priority".

Utilizing these four metrics, it should be pretty straight forward to isolate a CPU bottleneck. Lack of idle time may be an indication of CPU problems. By itself, a machine that has no idle time (100% busy) does not mean a CPU bottleneck condition is present. Use this metric in conjunction with the other metrics available. User mode is similar in this fashion. As a stand alone metric it could be misrepresented. However, if combined with other data points it can help identify if the CPU is the problem.

Run Queue is probably the most helpful metric for CPU bottleneck identification. Simply stated the Run Queue is the current number of processes that are waiting for access to the CPU at a certain point in time. In Unix, this Run Queue value is then used to compute the Load Average, which is a computed value (an average) of the Run Queue measurements.

What constitutes a bad Run Queue value or Load Average really depends on the size of your machine and your applications. Larger machines can handle larger values. This is similar to the aforementioned MPE/iX Ready Queue. What is important for the System Administrator is to understand the characteristics of their machine and understand the changes seen in the Run Queue or Load Average from normal conditions to a period of peak demand.

Memory bottlenecks on HP-UX can be identified by looking for the following symptoms:

HP-UX Memory Bottleneck Indicators

High overall swapping activity
High paging activity
Very little free memory available
High disk activity on swap devices
High CPU usage in system mode

The above metrics that we associate with a memory bottleneck condition individually will not pinpoint a problem. But much like the CPU metrics if several of these symptoms occur it indicates that physical memory has been under configured on the machine.

Swapping and paging are metrics that indicate that as processes gain and relinquish control of the CPU that memory is under enough pressure that these processes data structures and control structures can't remain in memory and must be swapped out to disk. Related to these activities is the overhead associated with the swapping and the paging that will show up in increased system CPU usage by the system daemons responsible for this overhead activity. If we are doing excessive swapping and paging then it also can be identified by the activity of disk activity occurring on the configured swap devices.

The question still remains on how much is to much swapping ? This is where again the local expertise of the System Administrator becomes important. By understanding the normal activity on a machine, changes in the metrics we have discussed will allow for a easier and quicker identification of the memory bottleneck.

HP-UX Disk Bottleneck Indicators

High overall disk activity

CPU idle waiting for I/O requests to complete
High overall rate of physical reads/writes
Long disk queue lengths

The disk metrics can be used to identify if the machine is bottlenecked on disk I/O. A point to remember is that on machines with many disks, an individual disk that is highly utilized may be your problem. The solution in this case would be to balance out the files on the popular disk to other disks.

Other areas to look into for removing a disk problem should include: application optimization, use multiple swap disks, and tune file system parameters. Application optimization can be achieved by perhaps doing larger and fewer I/Os or using memory mapped file techniques. If a swap disk is being heavily accessed then consider adding more swap area on other disks. Balance is what the System Administrator should be looking for here. File systems can be tuned using the HP-UX **tunefs** command which can result in better I/O efficiency. HP field personnel can provide guidance on this.

Kernel resources, Networking, and Graphics, the other bottleneck areas that we have identified can each be managed but will be secondary areas for concern if attention is focused on the primary resources of the CPU, Disk, and Memory. Kernel resources should be tuned if memory is tight. Removing un-needed drivers from the kernel will relinquish memory space and make more user memory available. Networking tuning can take place by the proper use of **netmask** features as well as using hardware devices such as bridges to help minimize and filter network traffic.

Management Techniques

Now that we have reviewed the performance basics of the MPE/iX and HP-UX operating systems we have an understanding of what we as System Managers need to understand from the technical side of performance management. The challenge will be to bring together these two disparate platforms into a cohesive program of performance management that provides consistency in terms of our techniques. The techniques in question include our method of identifying performance problems, monitoring performance and reporting performance.

The return on investment by focusing on consistency is that we mask the differences of MPE/iX and HP-UX at the lower technical levels and provide a uniform approach to performance management at higher levels . This allows for the System Managers to turn their attention to applying performance management to where it should really be and that is to be focused on maximizing system performance to the benefit of the business. The wrong approach would be to focus on the technical issues of each platform. This will only cause more confusion and in the end increase your costs of performance management.

Consistency can be achieved by implementing a strong performance management program that focuses on the following areas:

- Proper selection and purchase of performance tools and services.
- Adequate training and education on these tools.
- Instituting a regular reporting process for system performance
- Working closely with end-users to understand performance expectations.

Tool selection is important as you will need to fit your needs for a performance tool into the demands of your budget. Tools can be broken down into some distinct categories such as **diagnostic**, **resource management**, **capacity planning** and **application optimization**. In the Hewlett-Packard environment, products are available in each of these areas and can be purchased from Hewlett-Packard or a third party firm.

The type of tool you purchase depends on your budget and needs. It is safe to state that everybody should have a diagnostic tool to show you what is happening on the machine. Tools of this nature are real-time in design and provide you functionality to answer the questions of the user on "why the machine is slow now ?". They are also the least expensive.

Resource Management tools provide more trending information via the logging of data to files over extended periods of time. A tool such as this will allow the System Manager to implement the reporting process that a good performance management program demands.

Capacity planning tools and services available to end-users allow for the planning of future business demands on the computer system. If you are in a static environment then you may not need such a product or service. However, if you are in a high growth company, a capacity planning product or service (via Hewlett-Packard or third party consultants) is essential.

Application optimization tools tackle the issue of providing data for programmers such that application systems can be tuned for efficiency. Sometimes it may be cheaper to spend the time improving the efficiency of an application then purchasing more hardware.

In deciding upon performance tools consider the mixed mode environment that you will be supporting. Purchase a tool that provides to you the information you require but also works on both platforms. A common look and feel sounds unimportant, but after switching back and forth between MPE/iX and HP-UX any degree of consistency will be a savings. Do not make the mistake of treating each platform separately and purchasing tools for each platform independently. A mistake such as this will increase costs in terms of training, and supportability.

Once the proper tool or set of tools is selected do not forget training and implementation. If you are new to performance management it may be well worth the money to invest in a few hours of consulting on your purchase. Diagnostic tools are pretty straight forward and if a user spends the time to review the manual that accompanies the product, they should receive full benefit of the product. The capacity planning tools and resource management tools require some thought on their configuration. Each site is different and to receive maximum benefit they need to be properly utilized which again emphasizes the need for education and training.

Most important in a mixed mode performance management program is a regular reporting process that depicts system performance utilization. To be successful, this reporting program must be consistent and similar. The common look and feel found in a good, strong performance management program affords the System Manager the ability to report on the utilization of the computer systems in a manner which ties the information to the goals of the end-user. Technical reports are fine for the technical staff, but the measure of success in a performance program is the acceptance and understanding of the performance data reported and how it relates (either positively or negatively) to the business needs of the company. In a mixed mode MPE/iX and HP-UX environment it is imperative to keep this fact in mind. Use professional graphing packages (some performance tools already provide these) to report the data in a way that abstracts the lower level technical information and highlights the business components.

Examples would be to graph the impact of the Accounting department on the overall use of the machine. Try and break down the business components and correlate their usage of machine resources. Do not report ICS overhead and Swap utilization. No end user cares. Instead report how specific functions impact the machine.

Recently over the past three years the Hewlett-Packard community has started to hear about **Service Level Agreements, or SLAs.** SLAs are simply contracts that stipulate the performance expectations of the user community with those people supplying the computing resources. SLAs are common sense vehicles to insure communication is taking place such that business demands can be meet with the proper computing resources.

Our mixed mode environment demands that we manage at a high level our machines and the SLA is a process that fits in to the success of a well designed and managed performance program.

Conclusion

It is possible to manage performance in a mixed mode MPE/iX and HP-UX environment. It requires an understanding of the performance basics associated with each operating system. But by learning these performance basics and applying a performance management program that abstracts the low level technical information, confusion is eliminated and a proper performance management atmosphere is attained. This atmosphere is maintained by the proper evaluation and purchase of the required performance products and relevant implementation assistance. Professional reporting techniques should be used to inform the end user community of their use of system resources while also understanding their current and future expectations of system resources with Service Level Agreements.

Following steps outlined here and understanding the basics of the operating systems running in your environment will insure a proper return on investment.

Using Standard Tools to Build an Open, Client/Server Prototype

Paper # 2046

Bernard S. Hirsch
bernie@apollo.hp.com
Hewlett-Packard Company
930 East Campbell Road
Richardson, Texas 75081
(214) 699-4197

INTRODUCTION

The seemingly elusive goal, whereby the wealth of information and services in an enterprise is transparently accessible to end users on demand, is one about which much is written and discussed. Furthermore, there is an ever accelerating requirement to accomplish this in an open, distributed computing environment.

The first requirement that the computing environment be distributed is due to several ongoing trends in which an organization's business units, functions, data, users, and computing equipment are all now more and more distributed. The second requirement that the computing environment be open is resulting due to the need for these enterprises to control their own destiny. That is, they do not want to be reliant on a single hardware/software vendor when using information technology to help meet business goals. With the frantic pace at which new technologies, products, and methodologies are introduced nowadays, an enterprise would like to be in control to incorporate these to increase its competitive advantage, while still leveraging its many existing computing investments.

This paper describes how a prototype of an open, distributed application environment for stock brokers was created in which financial information and services are transparently accessible across a range of heterogeneous computing hardware and software, including multivendor operating systems, networks, architectures, and databases. Further, it is shown how this environment, the "Financial Desktop" (or FDT), was created using standard, off-the-shelf tools and technologies including the Open Software Foundation (OSF) Motif graphical user interface and components of the OSF Distributed Computing Environment (DCE). Using this open approach it is shown how installed assets are leveraged, new technologies can be integrated, and how the focus of control for the environment has shifted away from a single hardware or software vendor.

First, though, some key terminology is formally defined so that a common understanding of the goals of this prototype is achieved. Next, the FDT user interfaces and services are described in detail, and the tools and technologies that are available for developing this prototype are surveyed. An analysis of the final selected tools and technologies is then presented followed by a summary of the prototype development. Finally, the prototyped environment is analyzed and futures for it are discussed.

TERMINOLOGY

Open Systems

Some of the characteristics of an Open Systems-based solution include:

1. *Standards compliance* and conformance to international, national, and consortia recognized standards.
2. *Portability* to a variety of other environments by using standards compliant application programming interfaces (API's).
3. *Interoperability* with other components and environments using standards compliant protocols, API's, and common data formats.
4. *Scalability* of the solution to a potentially unexpected future scenario that has greater processing (e.g., faster CPU, graphics), networking (e.g., more clients per server, more protocols, more network traffic), and other demands on the environment.
5. *Availability* of the solution from a variety of vendors that have a commitment to Open Systems.

The concept of using Open Systems is important today because it holds the promise to help meet both the business pressures and technical challenges for the coming years. Some of the common benefits of architecting an Open Systems-based solution to a business problems are:

1. Prior investments can be protected and extended because of the ability to build upon core technology.
2. More control over technology choices can be exercised due to the ability to choose from a variety of hardware, software, network, and service/support vendors.
3. The standards focus of open systems facilitates easier incorporation of new technologies.
4. Productivity and efficiency can be achieved through interoperability, common user interfaces, and reusable code.
5. Cost reductions can be realized through reduced development time, although the investment needed to re-train and re-tool skills for long term cost gains should not be underestimated.

Distributed Computing

There are two notions of distributed computing/processing in use today. One notion is that of having replicated sites that are distributed with respect to their geography, and thus their various processing are also distributed. A typical scenario here is where an central mainframe polls the distributed sites and collects the various remote data nightly via modem.

The definition, though, that is used in this paper is that of a much more tightly integrated distributed computing environment, where a single computation (i.e., a database transaction) occurs over two or more computing platforms (or "nodes"). The major differences are the transparency to the user(s) and software developer(s), and the much shorter time duration to complete the computation.

Client/Server

The terms client and server do not imply any particular piece of hardware, such as an Intel 80x86 personal computer (PC). Rather, in this paper a client is the requestor, user, and consumer of some resource, service, or piece of information. And a server is the supplier or producer of that resource, service, or piece of information. A server typically provides access to the service that it supplies by providing a set of server operations.

Client/Server is one model of distributed computing that is in popular use today. In this "many to one" model, multiple nodes acting as clients access one or more other nodes acting as servers, for some required information, some service or resource (see Figure 1.)

It is important to note the subtle difference between client/server based tools and technologies, and client/server computing environments. That is, client/server tools do not necessarily yield a client/server environment, and in fact can be used to implement some other distributed computing model. Further, non-client/server tools and technologies can be used to implement a client/server environment.

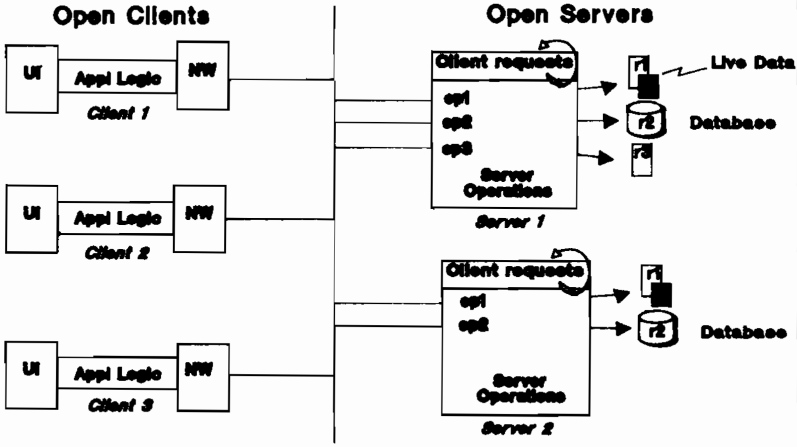
GUI

Graphical user interfaces (or GUI's), are a popular means to allow users to more intuitively (than with just text) interact with applications by presenting simple graphical representations of real life interface metaphors, such as windows, menus, and pushbuttons.

Since the GUI is what the user interacts with directly, this is sometimes also referred to as the client.

Interoperability

Client/Server Architecture



Open Clients

UI

Appl Logic

NW

Client 1

UI

Appl Logic

NW

Client 2

UI

Appl Logic

NW

Client 3

Open Servers

Client requests

op1

op2

op3

Server Operations

Server 1

r1

r2

r3

Live Data

Database

Client requests

op1

op2

Server Operations

Server 2

r1

r2

Database

Using Standard Tools to Build an Open, Client/Server Prototype 2046-Figure 1

Interoperability, in general terms, allows transparent communications between the nodes of a distributed computing environment. Factors that affect interoperability include common communication protocols, common API's, and common data formats. This paper addresses the problem of how applications and data interoperate using standard, client/server tools to build an open client/server computing environment.

DESCRIPTION OF THE PROTOTYPE ENVIRONMENT

User Interfaces and Functionality

The setting for the prototype is an application environment for a stock broker, that consists of a series of hardware and software components working together in unison, so that the stock broker can very efficiently and transparently get his or her job done, by having presented at their desktop all of the needed information and services. The software components that come into play in this environment are a collection of services (or servers) and user interfaces (or clients).

From the stock broker's perspective, four (4) user interfaces "drive" the entire application (see Figures 2-3). That is, all interaction with the system is done by simply entering some information -- such as the customer account number -- and then pressing a button in the GUI. All of the information that the stock broker needs to know about that customer, for example, and the status of his or her investments are then automatically and transparently presented to the stock broker in the same unified interface. These four user interfaces/clients are:

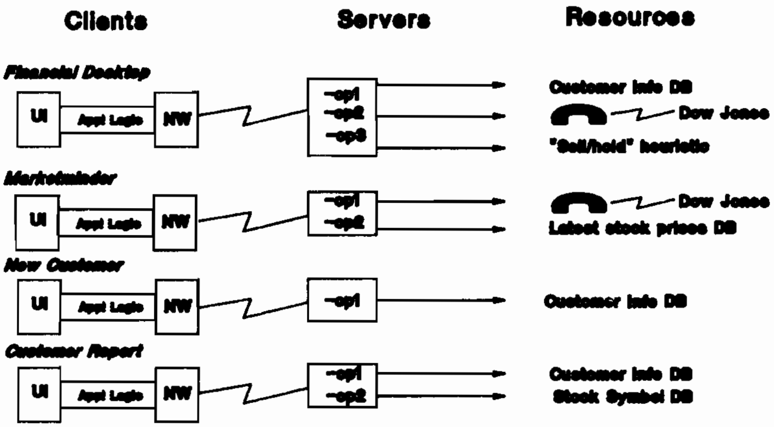
1. *Marketminder*, using a "real-time" feed to the Dow Jones, simultaneously presents the NYSE and NASDAQ ticker tape data (i.e., latest stock prices) and allows the stock broker to query the latest price for a particular stock. Specific information on each stock presented to the stock broker includes:

   a) Stock Name.
   b) Current, High, and Low stock prices for the current trading period.
   c) Stock price change between the current and the previous trading periods.
   d) Volume.

2. *Financial Desktop (or FDT)*, allows the stock broker to query a customer information database for individual customer and portfolio data. A heuristic function analyzes various data and suggests whether the time is right to sell particular customer stocks. Specific information presented to the stock broker includes:

   a) Customer name, social security number, and address.

FDT Application Architecture



Clients

Servers

Resources

Financial Desktop

UI

Appl Logic

NW

-op1

-op2

-op3

Customer Info DB

☎ Dow Jones

"Sell/hold" heuristic

Marketminder

UI

Appl Logic

NW

-op1

-op2

☎ Dow Jones

Latest stock prices DB

New Customer

UI

Appl Logic

NW

-op1

Customer Info DB

Customer Report

UI

Appl Logic

NW

-op1

-op2

Customer Info DB

Stock Symbol DB

Using Standard Tools to Build an Open, Client/Server Prototype *2046-Figure 2*

b) Number of shares and purchase price for each customer stock owned.
c) Current trading price for each customer stock owned.
d) Sell/Hold recommendation for each stock owned.

3. *New Customer,* allows new customer data and customer stock portfolio data to be added to the customer information database.

4. *Customer Report,* generates letters to all of the customers of the stock broker that own a particular stock. The letters advise them of pending recommendations and actions to sell particular stocks based on some activity. Additional functionality included in the "Customer Report" client allow the stock broker to either (a) look up a full stock company name based on an approximate stock symbol, or (b) look up an exact Dow Jones stock symbol based on the company name.

Services

From a functionality perspective, a significant amount of processing is transparently occurring to be able to present all of this information to the stock broker within each of these four GUI's (see Figures 2-3). A description of the various server data and operations in this environment will help explain some of this processing. Five (5) servers provide access to all of the information and services within this prototype environment. These five servers are:

1. *Customer Information Database service:*

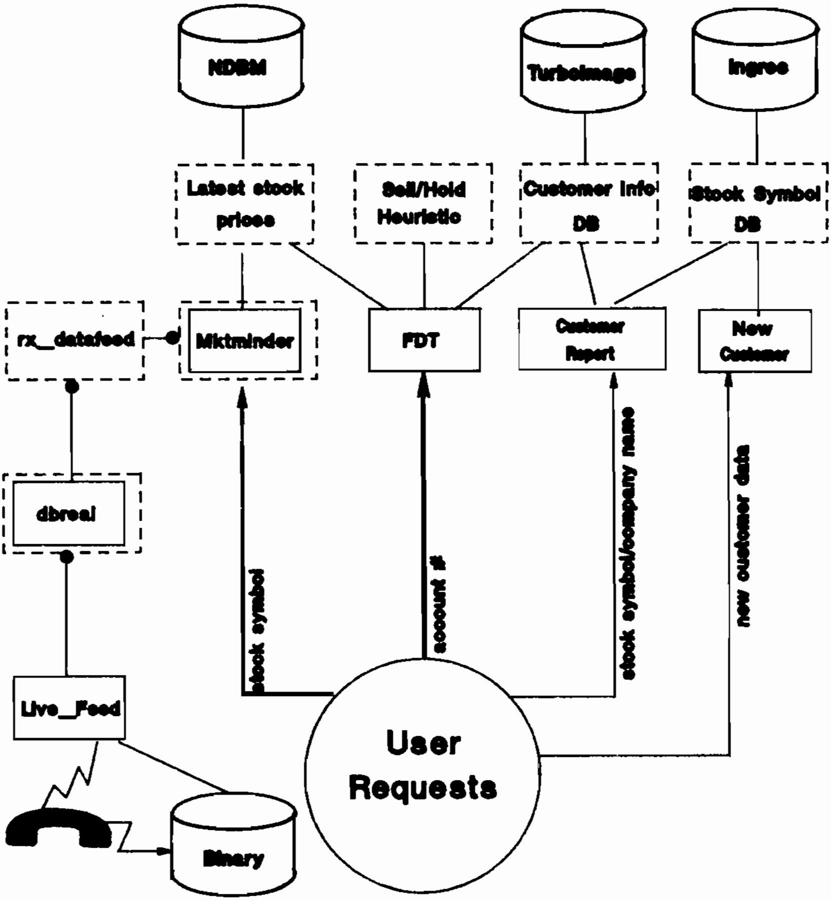
The customer information database service is implemented as an MPE/iX TurboIMAGE network database of customer information that includes the following data:

a) Customer account number, customer name, social security number, address, and number of stocks owned.
b) Stock symbol, number of shares, and purchase price for each customer stock owned.

Its server operations include:

i) "Get customer data" retrieves all of the data in (a) above, given a customer account number.
ii) "Get customer portfolio data" iteratively retrieves all of the stock information in (b) above for each customer stock held.
iii) "Add new customer" adds the new customer and customer portfolio data [specified in (a) and (b) above] to the database.

FDT Architecture



iv) "Get customer owning stock" iteratively retrieves all of the customers that own a particular stock, given a specific stock symbol.

2. *Dow Jones ticker tape service:*

The Dow Jones service provides a real-time feed to the latest NYSE and NASDAQ stock prices. Either the Telerate or Prodigy dialup services can be used as underlying services.

3. *Latest Stock Price Database service:*

The latest stock price database service is implemented as a UNIX NDBM database that maintains the latest stock prices for all of the stocks coming across the ticker tape.

4. *Stock "sell/hold" heuristic analysis service:*

The "sell/hold" analysis service provides a recommendation to the stock broker as to whether to sell the customer stock or hold onto it, based on historic and current market conditions.

5. *Stock Symbol Database service:*

The stock symbol database service provides a mapping from stock name to stock symbol, or from stock symbol to stock name, for each of the stocks in the NYSE and NASDAQ. This database service is implemented as a relational database using Ingres.

The following server operations are supported:

i) "Get stock symbol" retrieves the exact Dow Jones stock symbol, given the company name.
ii) "Get stock name" retrieves the company name, given the Dow Jones stock symbol.

Hardware, Operating Systems, and underlying Network

The customer information database service is implemented using an existing TurboIMAGE network database on a HP 3000 Series 900 business computer running the MPE/iX operating system.

The Dow Jones ticker tape service, latest stock price database service, and the "sell/hold" analysis service are all implemented on the following hardware and operating systems:

a) HP 9000 Series 700 workstations and Series 800 servers running the *HP/UX* operating system.
b) IBM RS/6000 Model 320 workstation running the *AIX* operating system.
c) DEC DecStation 3100 workstation running the *OSF/1* operating system.
d) DEC VaxStation 3100 running the *VMS* operating system.
e) DEC DecStation 5000/200 workstation running the *Ultrix* operating system.
f) Siemens-Nixdorf workstation running the *SINIX* operating system.
g) Groupe Bull workstation running the *BOS* operating system.
h) Stratus fault-tolerant minicomputer running its variant of the *UNIX System V Release 4* operating system.

The stock symbol database service is implemented using the Ingres relational database on the HP 9000/730 workstation running the HP/UX operating system.

The clients are implemented on the following platforms:

a) HP 9000 Series 700 workstations running the *HP/UX* operating system.
b) Intel 80386 PC's running Microsoft *DOS 5.0* and *Windows 3.0.*

The clients and servers are networked together using both Ethernet and IEEE 802.5 token ring.

SURVEY OF AVAILABLE TOOLS AND TECHNOLOGIES

User Interfaces

Several GUI technologies and tools are surveyed with respect to their applicability to this application environment. These provide the appearance and behavior with which the user will interact.

Based on the computing platforms on which the clients will reside, the GUI candidate technologies break down into two (2) primary areas: PC-based technologies and X-Windows based technologies. The available technologies and tools that were surveyed are discussed below.

Microsoft Windows GUI technologies and tools:

Microsoft (MS) Windows 3.0 Software Development Kit (SDK)

The Microsoft Windows SDK includes the Windows API and Windows libraries that allow software developers to build Windows-based GUI's. GUI's developed with MS Windows can run on 80x86 PC's.

Visual Basic

Visual Basic is an interface prototyper that facilitates the very easy creation of MS Windows GUI's without requiring one to know anything about the Windows API. GUI elements, such as push buttons, menus, and text fields, can be chosen from a menu and simply placed where needed with a few button presses. GUI prototypes generated with Visual Basic can run on 80x86 PC's in interpretive mode or executable mode.

Toolbook

Toolbook is another interface prototyper that facilitates the very easy creation of MS Windows GUI's without requiring one to know anything about the Windows API. GUI elements, such as push buttons, menus, and text fields, can be chosen from a menu and simply placed where needed with a few button presses. GUI prototypes generated with Toolbook can run on 80x86 PC's.

X-Windows GUI technologies and tools:

Motif 1.1 development environment

Motif is the GUI technology adopted by the OSF as a standard for creating user interfaces across multi-vendor computing platforms. Motif has a similar appearance and behavior as MS Windows and Presentation Manager (PM) thus obviating the learning curve for using this GUI. Primary differentiators for Motif are its intuitive, three-dimensional appearance/behavior and its support over 145 hardware platforms and 47 operating systems including UNIX, DOS, MPE/iX, VMS, and MVS. Motif has been submitted to a POSIX subcommittee as the recommended standard user interface technology.

Interface Architect

Interface Architect -- also known as UIMX by Visual Edge -- is an interface builder that facilitates the creation of Motif GUI's without requiring one to be an expert in the use of the Motif widget set (i.e., API). GUI elements, such as push buttons, menus, text fields, and scrollbars, can be chosen from a menu and simply placed where needed with a few button presses. Pure Motif 1.1 C code can be generated by Interface Architect, thus allowing portability of the constructed interface to any other platform that also supports Motif.

Open Dialogue

Open Dialogue is a user interface management system (UIMS) that facilitates the creation of Motif-compatible GUI's without requiring one to be an expert in the use of the Motif widget set. Further, Open Dialogue assists in the separation of user interface code from application logic, thus increasing the portability of the application to some other user interface environment. The Open Dialogue user interface description, however, is only portable to other platforms that also support Open Dialogue, which is currently very limited.

OLIT

OLIT, the Open Look Interface Toolkit, is the GUI technology developed by Sun Microsystems, Inc. and proposed as a GUI standard. OLIT is available from Sun and UNIX Systems Laboratories (USL). Open Look does not have similar appearance or behavior as MS Windows or PM.

Client/Server Application Interoperability

Several application interoperability technologies and tools are surveyed with respect to their applicability to this application environment. These are the enabling technologies that provide the "glue" between the clients and the servers. The available tools and technologies that were surveyed are:

Berkeley Sockets

Berkeley sockets is a networking API that allows for the creation of distributed applications by passing data between programs without requiring an understanding of the many layers of networking protocols. Various communications domains and protocols are supported by Berkeley sockets, including the Internet address family domain and its TCP and UDP protocols. BSD sockets, as they are also called, are typically available on any computer platform that supports TCP/IP.

X/Open Transport Interface

The X/Open Transport Interface (or XTI) is an API offering open access to the transport layer of the OSI model. XTI is implemented as a set of library calls that enable two or more processes on the same or different computers to communicate. Since XTI is an internationally recognized standard, it is available on many different computing platforms.

Network Computing System (NCS) 1.5.1

NCS is an implementation of the Network Computing Architecture (NCA), that defines a network communication protocol for implementing a remote procedure call (RPC) interface between separate processes in a distributed application. A remote procedure call is a generalization of the procedure call facility supported by many high-level languages. The NCS Network Interface Definition Language (NIDL) compiler plays a key role in this process by automatically generating the client and server network stub (or "glue) code, given an interface definition that includes the interface operation declarations.

OSF DCE

The OSF DCE provides many robust functional capabilities. At the heart of DCE is the API through which this functionality is accessed, the DCE RPC (a direct descendant of NCS 1.5.1). Similar in some respects to other RPC technologies, the DCE RPC allows for the execution of functions that reside on remote computer systems (see Figure 4). The DCE Interface Definition Language (IDL) compiler plays a key role in this process by automatically generating the client and server network stub (or "glue) code, given an interface definition that includes the interface operation declarations. Unlike other RPC technologies, however, the DCE RPC provides the following additional robust capabilities:

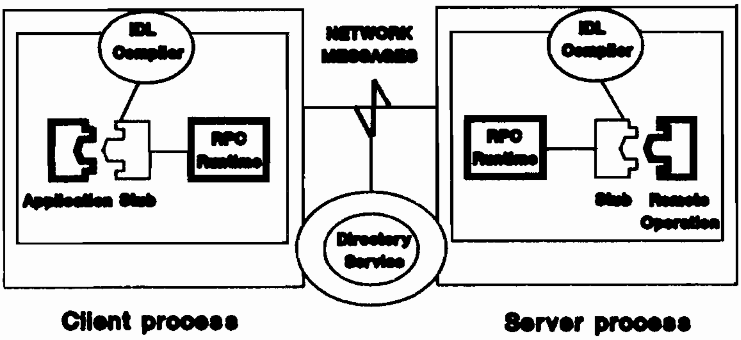
1. "At most once" (or local) procedure call semantics, by default.
2. Threaded RPC, for more convenient concurrent programming model.
3. "Receiver makes it right" network data representation, as opposed to a more costly canonical format scheme.
4. Integrated security service using authenticated RPC with options for requesting data integrity or data privacy.
5. Integrated cell-wide and global directory service capability, for transparent location of network services.

DECISION CRITERIA FOR SELECTED TOOLS AND TECHNOLOGIES

When deciding on key technology components, whether software- or hardware-based, many categories of issues need to be considered. Examples of these are:

Functionality
Scalability
Supportability
Cost
Timeliness
Ease of Use
Openness

Client/Server Implementation



There is no inherent priority to these categories. The priorities must be set based on the current and future needs for the target environment. In the case of the prototype discussed here, the prioritized decision criteria for the technologies are:

A1 Standards/Openness.
A2 Identified as strategic technology or widely used technology.
A3 Industry recognized support on range of computing platforms.
A4 Capability to deploy in production environment.
B1 Ease of Use.
B2 Functionality.
B3 Timeliness.
C1 Performance.
C2 Cost.

where "A" is a vital priority, "B" is important, and "C" has some value. Having prioritized these key decision criteria, they can then be mapped to the candidate development technologies and tools thereby allowing for an optimal decision of technologies and tools for the prototype environment.

The technologies and tools that were selected to be used to implement the Financial Desktop application environment are discussed below.

User Interfaces

1. Motif 1.1 using Interface Architect 2.0:

It was decided that OSF Motif 1.1 was the primary GUI technology to be used to implement the client user interfaces. The familiar and intuitive appearance and behavior of an interface that complies with the Motif style guide empowers end users to be productive immediately. Further, Interface Architect 2.0 was selected as the interface builder for the Motif GUI, due to its ease of use, quick prototyping capability, and ability to generate pure Motif C code, so that portability to future client platforms can occur very easily. In addition, since very efficient C source code is generated, the prototype interface code can also be deployed quite appropriately in a production environment.

2. Xvision X-Windows product for MS Windows 3.0:

Due to the requirement to protect the investment of existing '386 PC's as low end client platforms, Xvision X-Windows product for PC's was selected to allow the Motif generated interfaces to run on the PC's. This product turns the PC into a software-based X-terminal, while providing either Motif or MS Windows window management. It takes advantage of the fact that Motif is layered on top of the X-Windows distributed model, where X-based interfaces can be distributed to a computer that is different from where the actual application is running.

3. Microsoft Windows 3.0 SDK:

Finally, the Microsoft Windows 3.0 SDK was used as a secondary, supplemental GUI technology to develop MS Windows interfaces for two of the FDT clients that are to run natively on the PC platforms. Although MS Windows is not standard or open, it is the most popular GUI in use today on PC's, and thus it is strategic to many users. Also, it is extremely easy to use a Windows based application and it is very low cost.

Application Interoperability

1. DCE Remote Procedure Call (RPC):

It was decided that the OSF DCE RPC was the enabling technology to be used to achieve application interoperability between the four FDT clients and the five FDT servers. The primary decision criteria here was the openness of DCE, as an enormously popular consortia-sponsored interoperability standard, and as such the expectation that it will be increasingly available on almost every computing platform. For the prototype, early versions of DCE were used for '386 Windows-based PC's, Series 700 HP-UX workstations, and MPE/iX minicomputers, to achieve interoperability across the various FDT clients and servers.

Other decision criteria also affected the selection of the DCE RPC. First, the transparency, and scalability afforded the application developer and end users is much greater for applications architected with DCE RPC than with some of the less robust, pure messaging technologies. The familiar and intuitive local procedure call semantics empower software developers to be productive developing distributed applications with minimal training. Second, DCE RPC greatly simplifies the development of the clients and servers by automatically generating client and server stub (or "glue") C code. Most distributed computing complexities are taken care of automatically by the DCE RPC tools, generated stub code, and the DCE RPC library. These complexities include:

1. Making data formats compatible across different architectures.
2. Ensuring "at most once" (or local procedure call) semantics by default for server operations that require this guarantee (for example, a database update operation).
3. Providing transport independence in a multi-protocol world.
4. Having a client obtain a "binding handle" to a desired server, without knowing *a priori* the location of that server.

5. Taking care of communication failures and other error conditions that can occur between clients and servers. If possible, for example, the DCE stubs will solve the problem of a server aborting before completing a computation, by catching the failure and automatically "rebinding" to another equivalent server -- with all of this being completely transparent to the client application.

Finally, DCE RPC can just as easily be used to implement a production worthy implementation as it can to achieve the prototype described in this paper, since this third generation technology was designed with these goals in mind.

SUMMARY OF PROTOTYPE DEVELOPMENT

Before the summary of the steps involved in developing the FDT prototype application environment is presented, it should be noted that the selected tools and technologies promote parallel development of the application. For example, in the development of FDT, there were three types of software developers concurrently developing three different components of the clients and servers software. The first type is the GUI developer whose task it is to produce the appearance and behavior of the user interface, using Interface Architect. The second developer type is the client software developer whose job it is to write the client application logic to obtain the requested information and services from the various servers, and present the results to the user. Conceptually, this piece sits in between the GUI and the server. The third developer type is the server software developer who implements the specific server operations (or services) that were previously agreed upon. The implementation of these operations usually involve interfacing with some database, live feed, or other specific underlying service.

The benefit of this approach is not only that the prototype will be produced more rapidly since it is produced in parallel, but also that the learning curve is minimized since current skillsets can be partitioned into these three pieces. That is, for example, user interface experts can concentrate exclusively on building GUI's, and specific technology experts can deal solely with providing access to that technology/service by developing the server and server operations. An expert on the TurboImage database can provide access to TurboImage data by implementing the previously agreed upon server operations, and a client developer can then access TurboImage data without having to know anything about the TurboImage database, by simply invoking the appropriate server operation.

The primary steps, then, that were followed to create the FDT prototype are outlined below:

1. ***Client/Server Interface Definition:*** For each service, the client and server software developers need to agree upon the client/server interface definition and its supported server operations. These server operations include the operation name, input and output parameters required by the operation, and possibly some additional, optional attributes. This information is specified in a DCE IDL file. These server interfaces and server operations were discussed above in *"Description of the Prototype Environment"*. One of the server operations for the customer information database service is specified as:

```
[idempotent] void get_customer_data([in]  handle_t    h,
                                    [in]  char        acctNum[4],
                                    [out] customer_t  *custData,
                                    [out] long        *numStocks,
                                    [out] long        *status);
```

2. ***Interface Compilation:*** For each interface, the DCE IDL compiler is executed taking the DCE IDL file as input and producing as output the respective client and server side stub codes (see Figure 4).

3. At this point, the client and server software developers can start their parallel developments, since their interface has now been formally defined.

   3a. ***Server Implementation:*** The server software developer will develop the implementation for the server operations agreed to in Step 1, above. In the implementation of each server operation, the server developer needs to manipulate the underlying resource as specified by the operation. This entails using TurboIMAGE system calls, for example, in the customer information database server, to query or update this database. In the stock symbol service, the server developer needs to issue imbedded SQL statements to retrieve the requested stock symbol or company name.

   3b. ***Client Implementation:*** Development of the client application can also be done in parallel here, by first specifying the formal interfaces between the GUI and the application logic. What needs to be agreed to and specified up front are (1) the callback function names and parameters, so that the appropriate function can be called when the user presses a button, and (2) the names and types of the GUI objects/widgets so that the client software developer can read from and write back to the appropriate user interface elements.

      3b-i. ***GUI Development:*** The user interface developer will create the appearance and behavior of the GUI and will link the user requests for action (e.g., button presses) with client application "callback" functions. The information presented in the user interfaces was discussed above in *"Description of the Prototype Environment"*. Interface Architect was used here to create the Motif GUI's, and the Microsoft SDK was used to create the MS Windows GUI's.

3b-ii. ***Client Application Logic:*** The client software developer will develop the necessary "callback" functions and client application logic, and will call the server interface operations as needed. Also, the client software developer needs to read from and write to the appropriate graphical user interface elements. Some pseudo-code for the FDT client shows a sample of the flow of processing that occurs when the stock broker enters a customer account number and presses the "***OK***" button:

```
void ok_callback();
{
...
/*
 * First read the customer account number from the GUI.
 */
acctNum = XmTextGetString(textWidget);
...
/*
 * Next invoke the customer data "query" operation.
 * Note: This is an RPC call.
 */
get_customer_data(bindingHandle, acctNum, &custData, &numStocks,
&status);
...
/*
 * Then, put the customer data that was just obtained to the screen.
 */
XtSetValues(firstNameLabel, data1, count1);
XtSetValues(lastNameLabel, data2, count2)
XtSetValues(ssnLabel, data3, count3);
XtSetValues(addrLabel, data4, count4);
...
}
```

CONCLUSIONS

A prototype client/server financial application for stock brokers was developed using standard, off-the-shelf technologies and tools -- OSF Motif and OSF DCE. The technologies and tools that were used to develop the application were selected primarily due to their openness, standards compliance, and their capability to also be deployed in a production environment. By prioriting openness as the highest decision criteria, it is shown how installed assets can be leveraged, new technologies can be integrated, and how the focus of control for the environment can be shifted away from the vendor and back to the customer, where it belongs. By using standard compliant API's that are portable to many different systems (PC's, MPE/iX, and Unix), by using standard compliant protocols that are interoperable with other vendors' tools, technologies, and implementations, by using tools and technologies that are widely available by tens and hundreds of vendors, and by using tools and technologies that scale well from a prototype application to a fully deployed application, an enterprise can start to regain control of its computing destiny.

Specifically, it is seen that by using these open technologies, the investment in installed assets, such as the TurboImage database, can be protected, and in fact enhanced, by opening up access to the data to the entire enterprise. In this fashion, where the enterprise's business units can use this technology to methodically open up the access to their business data and services to anyone who needs to use them, business processes can start to be better optimized and reshaped.

It is also seen how new technologies, such as the Ingres relational database (and in the future, object-oriented databases and audio and video), can then also be easily added to the environment for the additional benefits of those latest technologies.

Finally, the client/server development process has demonstrated to lend itself very nicely to rapid and effective systems development, in part by allowing the capability for parallel software development, in part because the development tools (Interface Architect and the DCE IDL compiler) automatically generate much of the source code, and in part because if the services are developed in a generalized manner, they can be reused many times over by many new clients.

REFERENCES:

"Hands-On with Open Client/Server Technologies", HP Computer/Instrument Systems Training Course, 1992.

Lyons, Tom, *"Network Computing System Tutorial"*, Prentice-Hall, 1991.

Paper 2047

Using 4GLs for Application Development

Graham Masters

Software Technology Center
Hewlett-Packard Company
8010 Foothills Blvd.,
Roseville, CA 95678
(916) 785-5490

Introduction

This paper describes the features of typical Fourth Generation Language (4GL) environments. It discusses using 4GL's to both prototype and create production quality applications. Issues to be considered when choosing a 4GL are discussed, including how to spot potential problems areas such as performance and maintenance.

What is a 4GL?

We will start by noting that the term "fourth generation language" or 4GL is really a misnomer. This is because 4GLs are not just languages; they are environments for the *rapid development of database applications.*

A typical 4GL environment has the following components:

- A data dictionary.

- A screen creation tool, and a programmatic interface to the screens.

- A report generator.

- Interface to the target database systems.

- Control logic.

- Debugging tools.

Other features found in some environments are:

- Tools to enable the importing and exporting of data definitions to and from other environments.

- Tools to create databases directly from the data definitions held in the 4GL's data dictionary.

- Context sensitive help subsystems.

- Tools to assist with application creation.

  "Module Builder" in HP's ALLBASE/4GL and "Express Windows" in SQLWindows from Gupta Technologies are examples of this technology.

- Graphical User Interface (GUI) Support.

- Client/Server architecture.

- Multimedia features.

  For example: the storage and retrieval of voice and video.

The Data Dictionary

The scope of the dictionary will vary depending on a vendor's implementation. It may just keep the definitions of persistent data (i.e.. field and table definitions to match the target data base) or it could store every definition used by a 4GL application including data items, tables, temporary variables, 4GL source, and documentation. The important feature of a data dictionary in a 4GL environment is that it should be *active*. By active we mean that any change to a definition is propagated by the environment to *all* uses of the definition. The active propagation of changes ensures that all parts of the application: screen fields, data base items, temporary variables and so forth, remain consistent. This is in contrast to a dictionary which is used *passively* . Using COBOL with HP's product DICTIONARY/XL is an example of a passive relationship. Definitions can be extracted from the dictionary and translated into COBOL data divisions through the DICTCDE program. However if a data definition changes in the dictionary these changes will not be reflected in the COBOL code unless the developer goes through the process of running DICTCDE and manually incorporating its output into the COBOL code.

The Screen Interface

The screens are the means by which the end user interacts with the system. They include:

1) Forms for formatted data entry.

2) Report screens for the formatted display of data.

3) The method by which the end user navigates through the application. Namely: Function keys, menus, drop down menus, radio buttons...

The form that (3) takes will depend upon the interface medium - whether the interface is running on a character based terminal, or a Graphical User Interface (GUI) such as Microsoft Windows or Motif.

The 4GL must provide a scrolling window mechanism to allow browsing of database tables. Also, there is usually a way of combining a fixed data display window with a scrolling window. This makes it a simple task to create a data display for two database tables where there is a one-to-many relationship between the two tables. (In a TurboIMAGE database this relationship is normally represented with a master and a detail dataset.) An example of this capability is where a user enters a customer number into a form, and the application responds by displaying the customer information, name and address in one window, while at the same time displays a scrolling list of orders for that customer.

Report Generation

The environment must provide a method for creating formatted reports suitable for both display on the screen and for printing. As well as classic number and text reports, a package may include various charting features which are useful for the pictorial representation of summary results.

Database Interface

4GL environments typically support one or more relational database systems, simple sequential flat files, and an indexed sequential access method (KSAM files on MPE/iX operating systems, or ISAM on HP-UX.) Also, 4GLs running on MPE/iX often support TurboIMAGE databases.

The interface to relational databases, is almost invariably through SQL. For the other data sources, the 4GL will provide a record orientated interface

Control Logic

The logic of a 4GL application is responsible for:

- The application flow from screen to screen. This depends upon the menu selected, or data input, by the end user.

- Data validation. This ensures that data values which are input by an end user meet defined criteria. For example: "Product number must have 6 digits and be in the range 100000..999999"

- Enforcing the business rules applicable to the data. For example: "A customer cannot be deleted from the customer table, if the customer's number appears in the order table"

- The action to be taken in case of an unexpected error.

Debugging Tools

To be consistent with the 4GL goal of providing "rapid development of database applications" a developer writing an application will need symbolic debugging support. Remember that 4GLs are proprietary languages: thus this debugging assistance must be provided by the 4GL environment. Indeed, for those 4GL environments that are interpreted, the developer will have no other source of debugging support other than the facilities provided by the 4GL.

Typical symbolic debugging facilities are: logic statement tracing, displaying variable values, breakpoints and so forth.

Tools to Assist Application Creation

Some 4GL environments go a step further to ease the developers' task. They provide a package to create applications with no specification of logic at all. The screen fields are based upon definitions extracted from the database schema, using a default layout. The result is usually a straight forward application that can display and maintain data in a database. Although the applications generated using these tools may be limited, they provide a good start and they can then be customized using the 4GL environment.

Definition import tools and Database creation

The situation often arises where database definitions must exist within both the DBMS and the 4GL dictionary. If a particular 4GL architecture requires this duplication it will save developers time if the 4GL provides the tools to:

- Import those definitions from an existing database schema.

- Create a new database schema from the dictionary definitions.

With HP's product ALLBASE/4GL we get an interesting insight into the designers' intentions here. ALLBASE/4GL can import schema definitions from HP's network database TurboIMAGE, however one can not create a TurboIMAGE database from a 4GL dictionary. Conversely one can create an ALLBASE/SQL database from a 4GL dictionary definition, but not import ALLBASE/SQL schema (catalog) into a 4GL dictionary! Thus there is a migration path from TurboIMAGE to ALLBASE/SQL, but not vice versa.

Help Subsystem

Some 4GL's provide assistance with creating context sensitive help screens for the end user. Context sensitive help systems can be used to provide:

- Help on individual screen fields. (For example such a system can be used to build a help screen to answer questions like: "What value can I put in this field?")

- Help linked to individual error messages; thus cause and corrective action text for a given error condition can be embedded into the application.

Graphical User Interface Support

Some 4GL systems run under a Graphical User Interface (GUI). Two examples are Ingres/Windows which runs under both Motif and Microsoft Windows, and SQLWindows from Gupta which runs only under Microsoft Windows.

The multi-window support provided by a typical GUI is a very useful assistant for the developer as it allows multiple contexts at once. Thus while examining a screen definition, it would also be possible to look at the field definitions at the same time.

For the end user the main advantages of GUI in a 4GL application are:

- The application can support the display of picture images extracted from the database. For example a personnel database could contain the scanned picture of each employee.

- The application can create pictorial reports through the use of bar charts, pie charts and so forth.

- The application can export or import data from other tools. This is especially true in the PC environment where the application could be used to extract data from the DBMS and have the data copied to a spread sheet for analysis.

A GUI requires the use of a PC or a workstation. However, unless the DBMS is a single user system, the DBMS must reside on a shared computer node. To support this, either the 4GL must support a Client/Server (C/S) architecture or the DBMS must be distributed. However a distributed DBMS requires a full DBMS to be running on the PC or workstation. In the case of the PC or a disc-less workstation, this will probably cause performance problems or require an investment in a more powerful workstation. For this reason the C/S architecture is preferred.

Client/Server architecture

In a Client/Server (C/S) architecture part or all of the database application executes on a user's PC or workstation while the DBMS executes on another computer node. The PC or workstation is referred to as the client and the DBMS node is referred to as the server. The client and server are typically connected through a local area network (LAN) running Netware protocol or TPC/IP. Currently the message formats used between the clients and the server are proprietary. However for relational DBMS the conversations between the client and the server are more often than not based upon SQL.

For efficient support of the C/S architecture it is important that the DBMS supports stored procedures, and that the 4GL can take advantage of them. Without stored procedure support network traffic may be excessive.

When to use a 4GL

4GL environments are suited to the rapid creation or prototyping of applications that query, report and update database systems through a forms or screen based interface.

The underlying motive for using a 4GL is to reduce the costs of creating and maintaining an application. A 4GL enables a reduction in cost by reducing the number of lines of executable logic required to perform the application's task: fewer number of lines of logic means faster construction with fewer defects.

The fact that a 4GL can quickly create applications has made the *prototyping process* feasible. With a prototype approach, incomplete or vague system specifications can be used to create a prototype version of the application. This can then be tested with end users to clarify their requirements in a step wise refinement process. However the prototype application is often created without regard to issues such as performance and long term maintenance.

The question remains: should I use a 4GL for a production application? To answer this requires a candid assessment of the following issues for each 4GL considered:

- Are the features of the 4GL suitable for my intended application?

- The amount of system resource that will be available to run the application.

- Will a "large" application in the chosen 4GL be maintainable in the long run?

- Will the skill sets be available in the organization and marketplace for the long term maintenance of the application(s).

- Cost of training and consulting from the vendor.

- Long term vendor support.

- Costs associated with deploying the resulting application.

We will now consider each of the above issues in more detail.

Required 4GL features

The first question that must be answered is: "Will it be possible to create the application I want using the chosen 4GL?"

There are many variables that determine the answer to this question. Indeed the answer may not be realized until the unfortunate developer attempts to implement some application feature! However, there are some facets of a 4GL that can be investigated beforehand. Here, we will limit our discussion to the following few topics:

Data Types

All 4GL's can handle numbers and text as these are the "bread-and-butter" of data base management systems (DBMS). However one must consider if the application will need to use other specialized data types, such as date, time or Binary Large Objects (BLOBs). BLOBs allow DBMS's to store large volumes of data in a database field, and still have updates to that data under the control of the transaction management system. This data is not interpreted by the DBMS, only by the application. The BLOB can be used to store large text fields, images, or voice.

If the application needs to be able to handle graphic images, then the 4GL chosen will most likely be either running under Microsoft Windows on the PC or Motif on a Unix workstation. Unless the database is a single user database residing on the PC or the workstation, the 4GL will need to support a Client/Server architecture.

Interface Style

While all 4GLs in essence provide a forms-driven interface, the underlying capabilities of the presentation medium will restrict how the 4GL interacts with the user. What interface style you will need depends on your business requirements, future plans, and of course, budget. Here we will consider four styles and how they fit some stereotypical users and business situations.

There are four significant classes of presentation medium that 4GLs use:

1) Character based, one form at a time, interface.

2) Character based, windowing interface (overlapping forms.)

3) PC based Graphics User Interface (GUI). This is dominated by Microsoft Windows.

4) Unix workstation running the X Windows System. For example the Motif user interface.

Of course styles (1) and (2) only require low cost terminals, whereas (3) requires a PC, and (4) requires a workstation, or an X Windows graphics terminal.

Now let us take look at a number of stereotypical business situations and the end users. We have formed the groups according to the nature of their work, and the way they would tend to interact with the application:

a) Clerical staff. Clerical staff are data driven. They normally have simple, set work flows and handle large quantities of data. Often the applications are mission-critical or mission-important (i.e. the enterprise's business will be (severely) affected by application failure, or data loss.) They need flawless operation and good response times to do their job effectively. They probably would have difficulty dealing with more than two contexts at once (i.e. windows). They are not computer literate and need simple interfaces. Also, due to the high rate of data entry in such applications, the interface design should take into account such ergonomic considerations as minimizing hand movement over the keyboard during data entry.

Data entry environments are typically sensitive to per "seat" computing costs although users may already be provided with either terminals or low cost PC's. Even if provided with PC's these will probably not be powerful enough to run Microsoft Windows, networking software, and a 4GL environment in a satisfactory manner. An example of this group is a enquiry desk at an insurance company that answers customers' questions concerning their policies.

Interface styles (1) and (2) are the best fit.

b) Technician. Uses the application to retrieve technical data. Lower volume of data than (a). May need graphics display for displaying parts drawing, etc. They probably would have difficulty dealing with more than two or three contexts at once. They may not be computer literate and need simple interfaces. Depending on individual situation, application response will be important and it may be mission-important.

If graphics are needed then interface styles (3) or even (4) may be needed, otherwise (1) or (2) are acceptable.

c) Corporate Professional or Manager. Uses application to retrieve and maintain data about enterprise's business. Most likely PC computer literate, and already have a '386 or '486 PC at their desk. May use multiple contexts (i.e. windows). Application response time less important than (a) [If the application saves a week of work, does it matter that the chart takes five minutes to plot?] Probably will need charting and spreadsheet capabilities or links to tools to provide those capabilities. Moderately sensitive to per "seat" computing costs.

Interface style (3) will probably be the best fit.

d) Engineer. Uses application to retrieve data about project, or assist in its design. Computer literate, likes to use multiple contexts. An engineer typically already has a large investment in computer equipment and software, and so will not be sensitive to per "seat" computing costs.

Interface style (4) will be the best fit.

e) John Doe. The user has no foreknowledge of the system, nor the application interface. The application interface must be very simple and self explanatory. Color and graphics are important. Interaction may be via the screen rather than keyboard or mouse. Per "seat" computing costs are not applicable. Application will probably not be mission-critical nor mission important. An example is the walk up automobile reservation system used directly by customers.

Requires using the graphics features of interface style (3) with the restricted options provided by style (1). Typically this would be implemented either with a custom interface, or by using style (3) with access restrictions to prevent the user leaving the application.

The Operating Environment

The chosen 4GL must execute on the proposed mix of hardware, O/S, and target DBMS. However if there are plans to re-sell the resulting application, then thought must also be given to the advantage offered by those DBMS and 4GL combinations that execute on multiple hardware and O/S platforms.

Report Writer

4GL environments are normally packaged with a report writing facility to allow the creation of printed reports. In keeping with the flavor of the 4GL screen environment, the report specification will be done with a layout tool or painter. Here the various report entities such as text, data fields, sub totals, totals and so on are positioned on the screen in a "point and click" style rather than the specification of row and column positions. However, such packaged report writers may not be as powerful as a standalone full function report writer. For this reason, it may be wise to determine at the outset if an alternative report writer is available for your environment in case your reporting needs grow in sophistication over time.

Performance

A well written 4GL program will consume more computer resources than a well written 3GL program. Thus on a given computer configuration, the 4GL environment will support fewer users, or the end user's response time will be slower. Some informal bench marking done at HP has suggested that a good performing 4GL can achieve about 70% of the performance of a comparable C program. Some 4GL's may perform considerably slower.

Thus a choice has to be made between the extra development cost of an application written in a 3GL and the extra cost of more computing power. Experience with assemblers verses 3GLs in the 70's tells us that it is worth the extra hardware costs to enable the use of the higher level language and if necessary, re-code performance sensitive areas in a lower level language.

However there are some indicators that can be used to gauge how much computing resources an application created with a given 4GL will consume. The following sections discuss several performance related issues.

Dynamic verses Static SQL

- 4GLs that use *dynamic* SQL as the interface to the database system will be slower than 4GL that use *static* SQL.

4GLs usually interface to a relational database through the SQL language. Like many languages, SQL can either be compiled and optimized before hand, or interpreted at run time. If the 4GL uses static SQL sections, then the SQL is compiled before hand. If the 4GL uses dynamic SQL then the ASCII SQL statements are interpreted (actually they are compiled and optimized on the fly) during the application execution. Dynamic SQL allows greater flexibility because it provides the capability of end users entering ad hoc SQL statements. However it is debatable whether the typical end user of a 4GL application should be allowed access to SQL statements. Apart from a lack of knowledge of SQL, an inappropriate query such as SELECT upon a non indexed column in a large table could cripple a system.

Stored Procedures

- 4GLs operating in a Client/Server (C/S) environment will perform better if they have access to stored procedures.

The stored procedure facility was originally developed by DBMS vendors to improve the performance of their C/S systems. In C/S, stored procedures significantly reduce the network traffic between the server and clients, as one stored procedure can do the work of many SQL statements, and yet only requires one request/reply message pair.

A stored procedure is simply a sequence of SQL statements compiled, optimized, and stored in the DBMS. The sequence can be invoked by procedure name through a SQL "execute procedure" statement. The variables to individual SQL statements in the procedure are then passed via procedure parameters. Also the SQL is often extended to allow the specification of flow control within the procedure.

The use of stored procedures can ameliorate the performance problems associated with the use of dynamic SQL in a 4GL, as long as the 4GL allows the developer to use the "execute procedure" SQL statement. Effectively the 4GL causes only the execute procedure to the compiled on the fly, the bulk of the work is done by the precompiled statements in the procedure itself.

It is worth noting that a good stored procedure facility can reduce the amount of logic required in a database application - rather than encoding the enforcement of integrity constraints and business rules into the 4GL or 3GL application, these rules can be encoded into stored procedures. Viewed from an integrity perspective, stored procedures are valuable, because they enforce the integrity constraints upon *all* accessors of the database system, including users of miscellaneous ad hoc query tools.

Use of the SQL interface

Some 4GL's that interface to many DBMS's that are both relational and*non-relational*, may not take full advantage of the query optimization capabilities of a relational DBMS. This situation can arise because the 4GL may try to present a relational interface to the developer for both relational and non-relational DBMS. To do so, the 4GL has to implement relational operations such as JOIN for the non-relational DBMS. However it is easier for the 4GL if it can treat both kinds of DBMS the same way at a lower level - thus the 4GL environment may use its relational capabilities rather than the underlying capabilities of an relational DBMS. For certain operations such as JOIN on larger tables, this interface technique has a very serious performance implication.

With regard to vendors' performance claims it should be noted here that the DBMS benchmarks based upon the "banking database", namely TP1, TPC-A, and TPC-B only exercise the relational operations: UPDATE, INSERT and SELECT. Most business applications also need to use other relational operations such as JOIN and ORDER BY.

Interpreted verses Compiled Language

4GL environments can take one of the following approaches:

1) Interpret the ASCII 4GL logic statements directly.

2) Compile the ASCII 4GL logic statements into an internal representation, which is then interpreted at run time.

3) Translate the ASCII 4GL logic into a 3GL, and then compile the 3GL. (These systems are often called generators instead of 4GLs.)

4) Compile the ASCII 4GL logic directly into executable programs.

Generally, technique (1), has a greater performance overhead than (2), which in turn has a greater overhead than (4). The performance differences realized between (2) and (4) will depend on the the efficiency of the internal representation and the interpreter, compared to the effectiveness of the compiler's optimizer (if indeed any is used!). However, a program which was compiled with a compiler using an good optimizer will always outperform an interpreted program.

In a Client/Server architecture where the 4GL application is executing on a client, the above discussion should not be a concern. The resource overhead of an interpreted 4GL is not significant compared to the other client processing loads, caused by subsystems such as GUIs and LAN software. With Client/Server systems, the developer should be more concerned with the issues discussed above, in the "Stored Procedures" section.

4GL Program Maintenance

During the life of a production application, more time and effort is put into its maintenance than into its creation. A large application, even in a 4GL, is still a complicated beast. It will probably contain over 100 screen images, and perhaps 10,000 or more lines of logic. Thus it is very important that the tools and techniques used to create the application do not hinder its maintenance.

Open verses Closed Development Environment

In some 4GL development environments the application is created through a dialog with the 4GL development system. In this type of environment the 4GL provides everything: screen painter, logic editor, version control system, debugger, and so forth. We'll refer to such an environment as "closed". In other 4GL environments, specification of at least some of the 4GL application (usually the logic) occurs via 4GL specifications kept in ASCII files. We'll refer to such an environment as "open".

In a closed 4GL all tools, normally associated with the successful maintenance of large programs must be provided by the 4GL environment as access to outside tools is either not possible or limited. The tools that immediately come mind are: editors, x-reference facilities, revision control, and debuggers.

In an open 4GL, because some specification is done via ASCII files, the developer has access to familiar tools: vi, emacs, xref, grep on HP-UX or tdp, hpedit, hpsearch on MPE/XL.

If you choose a closed environment it would be wise to find out if the tools are adequate for the maintenance of large programs. Or as an alternative, are utilities provided to unload and load all specifications to ASCII files? This can provide an escape route, allowing the developer to analyze, and edit the specifications outside of the 4GL environment.

The language

The 4GL environment will have a language to express the logic of the application. The language may have been influenced by one of the well known 3GL's, and thus may show some similarities to COBOL, BASIC, PASCAL or C. However all said and done, the 4GL will be proprietary and often the only text available about the language will be the vendor's reference manual.

4GL languages often claim to be *declarative* rather than *procedural.* By a declarative language we mean that the developer has to only specify *what* is to be done, not *how* it is to be done. Consider SQL for a moment. SQL is a declarative language. When one specifies:

```
"SELECT  number, f-name, l-name FROM customer
        WHERE number = 75984;"
```

The select statement only specifies which row is wanted, not how the row is to be found. Of course in a large table we hope the system chooses to use an index!

However, rarely are 4GL specifications purely declarative. Ultimately the logic must specify how to handle error situations, or specify that certain calculations have to be performed. In this case the logic will be procedural and for larger applications should allow good structured programming practices just like a 3GL. The constructs to look for are:

1) Functions to allow the decomposition of the logic into manageable pieces
2) Function parameters.
3) Function local variables.
4) Structured flow control: if...then.., if...then... else
5) Structured loop control: while, repeat and for loops.

A 4GL may have a limited power of expression, so another worthwhile feature to look for is a way to access routines written in a 3GL and to be able to pass parameters back and forth. For certain computing tasks 3GLs are better: they can express the algorithm in less code, and the resulting code will execute faster.

Available skill sets and Consulting

There is no standard for 4GL environments. Thus one vendor's language and environment will be very different from the next vendor. For this reason the pool of knowledge available within your organization and indeed within the marketplace for a given 4GL will be limited. Thus vendor provided consulting will be an important asset when developing a large and complex 4GL application. A consultant can provide important knowledge that may be difficult to otherwise find.

Having chosen a 4GL, management should make allowances in the project budget for training and consulting.

Long Term Vendor Support

As we have mentioned above, the language will vary from vendor to vendor. The consequence of this is that it is potentially expensive to port a large application from one 4GL to another: you just can't change compilers and recompile. Typically each data definition, and logic statement has to be inspected and hand translated. To be forced into this situation is indeed unpleasant and to be avoided if at all possible. For this reason you need to be certain that if you have a problem with the 4GL in your production application, that the vendor can deliver support, both now and five years from now. Indeed, many business applications have a life span of a decade or more.

Cost of Deployment

A 4GL application in an interpreted environment may have extra costs associated with its deployment. It may be necessary to purchase extra run time licenses for each computer or even each "seat" depending on whether the vendor uses user based pricing or platform based pricing.

Conclusion

4GL's allow a prototyping approach to be a viable methodology of refining an application's specification. 4GLs can also be used to create final production versions of an application. However careful thought, planning and assessment are required to prevent problems arising later on in the life cycle. We would recommend that a potential developer use the prototyping phase to not only create the application prototype, but also to evaluate in depth the features, performance, and maintainability of the chosen 4GL.

Some vendors provide a trial copy of their 4GL scftware at little or no charge. We would encourage you to take advantage such offers to complete initial assessments of alternative 4GL environments.

Measuring and Modeling UNIX Systems

Dan Sternadel
Performance Technology Center
Hewlett Packard Company
Roseville, California

Abstract

Queuing Network models intrinsically do not have knowledge of operating system platforms. The biggest problem in building UNIX models is not the models themselves, but deriving inputs from measurement data. This paper discuss a new technique for gathering measurement data from a UNIX system that can easily be translated to model inputs. This technique supports continuous collection, pre-summarization of data logged, and pre-defined workloads. The measurements will be presented in terms of building a queuing network model.

As UNIX* systems become more prevalent in the business environment, UNIX capacity planning techniques will determine the long term success of UNIX in the commercial computing arena. Although the methodologies and techniques for capacity management in the proprietary commercial operating system environment (such as IBM MVS) are mature, it will take some effort to port that technology to the UNIX arena. UNIX measurement tool developers have the opportunity to reference a significant knowledge base from which they can build new measurement technologies. Since capacity planning methodologies are already established, performance data can be fine tuned to accommodate capacity planning.

Measurement Technology

Historically, computer system performance measurements were considered a necessary evil. Diagnostic problems such as unacceptable response times or poor throughput created a demand for diagnostic tools to troubleshoot low level system problems. System capacity management engineers seized the opportunity to utilize the detailed diagnostic data to build analytic models. Since the data was far too detailed in its raw form for analytic model inputs, a great deal of data summarization was required.

*** UNIX is a trademark of AT&T Bell Laboratories**

Another problem with using diagnostic data for analytic models is validating the derived model. Since most computer system diagnostic tools attempt to measure response time based on some sort of trace, several difficulties arise in validating response times. Trace based tools typically measure the time between a terminal read complete and the next read attempt. The problem is that severe "end effects" may occur since we do not record the response time until the event completes **[Ferrari83]**.

Defining a transaction can also cause a significant amount of grief. For environments where a program does not issue an external trigger event, such as a terminal read, one may consider counting a complete program execution as a transaction. On a UNIX system where several thousand programs may be executed, it is difficult to correlate program execution rates to any meaningful business transaction counts.

Since analytic models do not require trace measurement data, we can consider an alternate method of data collection. One can think of an analytic model in its simplest terms as a central system that provides service to a customer. A customer enters the central system and moves through the servers (CPU and disk etc.) then exits the central system to wait on a terminal read (terminal class) or immediately re-enter (batch class) **[Lazowska84]**. With this in mind we can think of a customer as either being IN the central system or being OUT of the central system. Customers IN the system are consuming resources and are therefore accumulating system response time. A customer that is OUT of the system is not consuming resources and therefore is not accumulating response time.

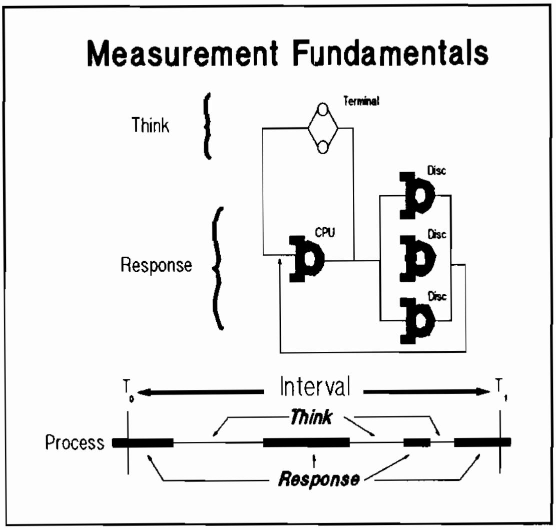


Figure 1

Figure 1 shows that during the life of a process it is either IN or OUT of the central system. The thick line within the interval represents the amount of time the process spent either acquiring service or queuing for service at the system servers. Our research efforts have resulted in a UNIX measurement technology that can identify the states that a process is in for the entire life of the process. By classifying the wait states from these measurements we can gather performance measurement data that is more appropriate for analytic models.

IN the system	*OUT of the system*
CPU seconds	*TTY wait*
Disk wait	*Console wait*
Dispatcher wait	*Son wait*
Memory wait	*Father wait*

Table 1

Table 1 shows how we may bucket different wait times as being either IN the system or OUT of the system. With this technique we can calculate a total system response time for any process by summing up the IN system time. The think time for terminal classes can be calculated from TTY wait time.

Summary Data Collection

Finding the right balance between data collection detail and ease of data interpretation has continually been the nemesis of capacity planners. As the fundamental characteristics of a particular computer systems become better known, collection tools can pre-summarize detail measurement data thereby reducing the complexity of data analysis. Since UNIX systems provide a similar operating system architecture across mult-vendor platforms, the potential for a industry standard set of performance metrics could be realized.

Characteristic	Detailed Collection	Summary Collection
30 Megabytes of Data	1 Hour	1 Month
Diagnostic Capability	Excellent	Good
Long Term Trending	None	Excellent
Modeling Capability	good	Excellent
System Overhead	>10 %	< 10 %
Consumer of Data	Engineer	Engineer/ Manager

Table 2

Table 2 compares the general attributes of the two different performance data collection techniques. Except for low level diagnostic capabilities, long term data collection techniques excel over detailed data collections. The higher level data is easily consumed by less technical personnel, allowing the business planners to have a better understanding of their interactions with data processing. Since the data is pre-summarized it is available for immediate analysis, eliminating the need for post collection data reduction.

It may seem counter-intuitive that reducing performance measurement data at collection time results in less overhead than post reduction techniques.

Our research has consistently shown however that logging the right data in a summary form produces less system overhead. This is based on the observation that most of the overhead occurs in the logging routines as compared to the actual data collection. By pre-summarizing the data we significantly reduce the amount of overhead incurred for logging. In fact the resources saved from the reduced data logging activities more than make up for the additional computations required for summarization at collection time **[Wade88]**.

Predefined Workloads

If at data collection time we summarize the process wait states into predefined workloads, we can significantly reduce the amount of overhead required to log detailed process data. For example, if we have a method of defining a workload called "compilers" which could summarize all of the compiler processes wait states into one record at a five minute interval, we could very easily build a modeling workload definition from this workload record. Not only do we have the data needed to parameterize the workload, but we also have the IN system response time to validate against. This technique produces minimal overhead on the system being measured, thereby allowing continuous data collection.

Pre-summarization of workload data poses a significant problem for trace based collection techniques. Aligning the work performed by the workload with the response time logged to the workload can be difficult, if not impossible. For example, consider a shell menu process that initiates a terminal read, then launches a child process that actually executes the work the user requested. If the child process does not perform any terminal activity, then there is no way in a trace based scheme to assign a response to the child process. This results in the menu driver process logging response time although it did not do any work. The child process on the other hand will be charged with CPU and disk consumption while not logging any response time. We cannot expect a model to validate in this situation.

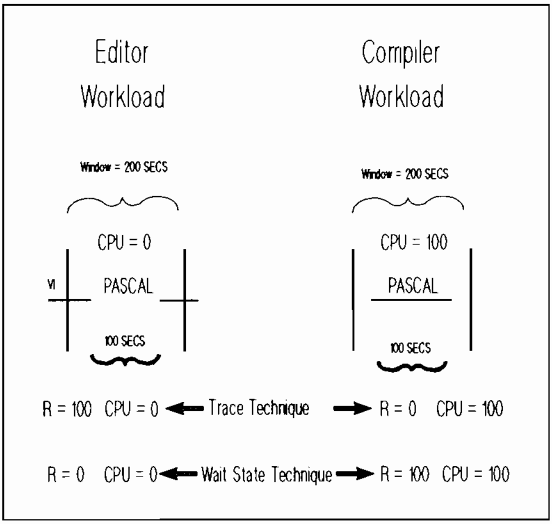


Figure 2

Figure 2 demonstrates the problem of aligning work with response time in a trace based collection. The **Editors** workload will log all editor processes such as *vi* (a UNIX screen editor), while the **Compilers** workload will log all compiler activities such as a Pascal compile. If *vi* launches a Pascal compile, trace based collection techniques will have no way of knowing when to shift response time metrics to the **Compilers** workload (assuming the Pascal compiler does not issue a terminal read). One may speculate that we could transfer response time metrics when a terminal write occurs from the Pascal program. The problem is that if there were no compiler errors, response metrics would be transferred at the completion of the compile (when the end of compile message is displayed), after the work had been completed. This would only slightly improve the situation.

The wait state method of system response time collection overcomes the trace based problems by essentially keeping response time metrics (IN system) aligned with the executing process. In Figure 2 the wait state technique shows that the response time will be aligned with the work. Since the CPU consumption occurs in the Pascal compile, we will log CPU seconds (and queuing time) into the IN system time for this execution. If more that one compiler was running concurrently, then their IN system time will be summed into the **Compilers** workload IN system time. The IN/OUT of system technique offers an intuitive approach for building models from measurement data.

The disadvantage of this technique is that we must know what the workload definitions are before we collect the data. But since we can run the collection tool continuously, we can make adjustments at our leisure, allowing a continual refinement of workload definitions.

Model Inputs

For a terminal class **[Lazowska84]** the population of users needs to be determined. **Effective users** is a term used to describe the number of users that were effectively thinking or responding during an interval. If in a 60 minute interval a process was in a terminal read wait state for 45 minutes and spent 15 minutes in system response time, then we would have 1 effective user:

(45 Think + 15 Response) / 60 minutes = 1 Effective User

Now consider a 60 minute interval that has several processes running concurrently. For example, if we summarized at collection time all of the terminal read wait time and system response time, say 950 minutes and 250 minutes respectively, we would have 20 effective users:

(950 Think + 250 Response) / 60 minutes = 20 Effective Users

Notice that effective users is derived directly from the wait states. This technique removes ambiguities typically found in parameterizing terminal classes. For example consider the situation where a single terminal is shared by 3 employees. Typically an employee will logon to the system for several minutes then logoff. There may be several minutes of logoff time before another employee uses the terminal. From a modeling perspective it is not appropriate to say that there were 3 users since there was only one physical device. Nor is it appropriate to say that there was 1 user since the device was not being used during the entire interval.

The effective user technique will calculate an equivalent user load from a modeling perspective. The logoff time is automatically removed from the calculation since the device is inactive (there is no wait state). The sum of terminal read wait time and system response time will result in an effective user population that is appropriate for analytic modeling. If each of the three employees described above were running separate programs, we could couple the pre-defined application technique to calculate an effective user for each of the "actual users".

The basic calculations for parameterizing a terminal class are:

Think Time = TTY wait / transactions

CPU/Tran = CPU seconds / transactions

Disk /tran = physical disk I/O counts / transactions

Effective Users = (IN system time+TTY wait) / interval

Disk service times can be calculated from system global disk utilization metrics, which gives us all the information we need to build a terminal class for an analytic model.

Batch classes are a little more complex. From a modeling perspective, a batch workload is a terminal workload without think time. We assume that when a batch "job" is completed it is immediately replaced by another exhibiting exactly the same workload characteristics.

Effective jobs is a term used to describe the productive time a job spent during a selected interval. A job that is consuming resources (such as CPU or disk) is considered to be "responding" to the jobs demands, while a job waiting on a system external action (tape mount, operator request, etc....) is considered to be non-productive, and therefore "not responding" to the jobs demands.

What we need is an independent indication of effective jobs that exist during the modeling window. The workload pre-summarization techniques allows us to measure the total IN system time (responding to a jobs demands) that an application had during an interval. For example, consider JOB A that was launched at 8 AM and finished at 11 AM. If our modeling window was from 9 AM to 10 AM then we would see that one hour of time was spent responding in that window. In this case we would have one effective job in the 9 AM to 10 AM window:

1 Hour Response / 1 Hour Interval = 1 Effective Job

Now suppose that JOB B was submitted at 9:30 AM and did not complete till 11:30 AM. Since our measurement window is from 9 AM to 10 AM we would see we had 1.5 hour response in the window (JOB A + JOB B). We would now have 1.5 effective jobs:

1.5 Hour Response / 1 Hour Interval = 1.5 Effective Jobs

Consider one more scenario where there was a constant stream of JOB C's that were submitted. Each Job lasted 10 minutes and was immediately replaced by another Job C that lasted 10 minutes. This Job environment exists from 7 AM to 5 PM. We would now see that we had 2.5 hours of response in the 9 AM to 10 AM window (JOB A + JOB B + (6 * JOB C)). This of course results in:

2.5 Hour Response / 1 Hour Interval = 2.5 Effective Jobs

The most important point to note is that we are not concerned at all with how many "actual" jobs existed in the system but more with how many **"effective jobs"** existed in the system. In addition, effective jobs only have meaning in the context of the measurement. For Example, 2.5 effective jobs only has meaning when talking about the system we measured. This effective job value is a result of some amount of demands placed on the system and its devices during the interval, resulting in some "response" that was collected for us.

Deriving batch class model inputs from pre-summarized application records is very similar to that of terminal classes. The main differences are that transactions are replaced by effective jobs, and workload populations are defined as effective jobs rather than effective users. So we now have the following definitions:

Effective Jobs = IN system time / Interval

CPU / transaction = CPU seconds / Effective Jobs

Disk / transaction = physical disk I/O / Effective Jobs

Business Units

Successful capacity planning exercises relate business planning to data processing resources. Business units[1] can quantify system demands in terms of business activity, such as number of orders received, widgets manufactured, widgets shipped etc. When we correlate business activity with a system resource such as CPU **[Browning90]**, we have a powerful means of parameterizing analytic models.

The IN/OUT of system technique allows us to very easily implement a methodology for incorporating business units into our modeling exercises. By taking the IN system time and dividing it by whatever business units are appropriate, we can represent a system in terms of its business function.

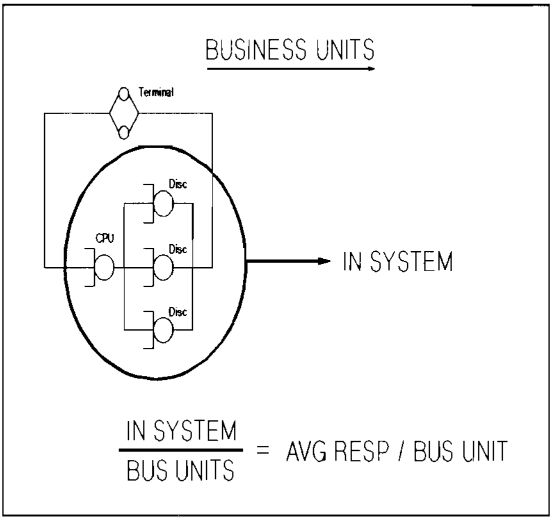


Figure 3

Figure 3 shows conceptually how the business unit scheme works. For example, if we were a wine distributor, our business unit may be the number of cases of wine shipped. By dividing the IN system time by the number of cases shipped we would be defining response in terms of our business rather than in terms of our computer. Since computers support business (rather than the other way around) this is a preferable technique. Of course our

[1] also known as natural units, business indicators, application workload units etc.

demands would be defined in terms of business units also. For example a terminal class parameterization in business unit terms would be as follows:

Think Time = TTY wait / Business Unit

CPU/tran = CPU seconds / Business Unit

Disk/ tran = physical disk I/O counts / Business Unit

For batch class demands we would merely replace effective jobs with business units.

Of course it is up to the application to define what the business unit is, and provide some method of logging system business unit completions. Defining, quantifying, and collecting business unit data is often the most difficult task **[McGalliard89]**, but the return on investment is well worth it. Once a successful business unit strategy is in place, computer system capacity planning can be logically tied to the business planning process. As business demands increase (or decrease), business unit calculations can quickly evaluate system capacity requirements for the data processing department.

Examples

The following data was collected from a commercial UNIX system using the pre-summarization techniques previously discussed. Collector overhead was less than 2 percent, which allowed the collector to be enabled 24 hours a day.

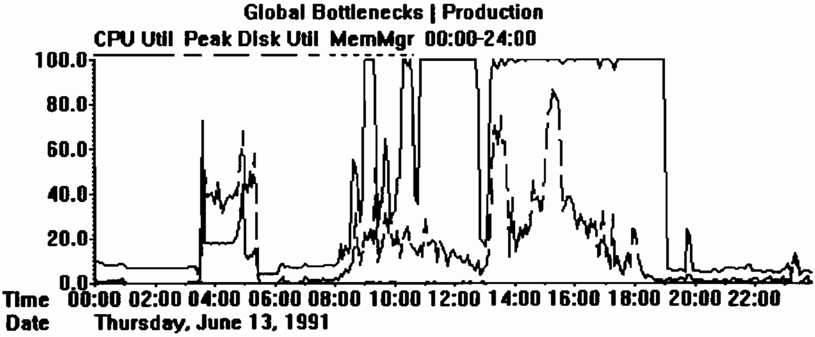


Figure 4

Figure 4 shows overall system activity. This graph can be used to gain an understanding of general system characteristics. For example, we can differentiate the overall disk to CPU relationships at different times of day.

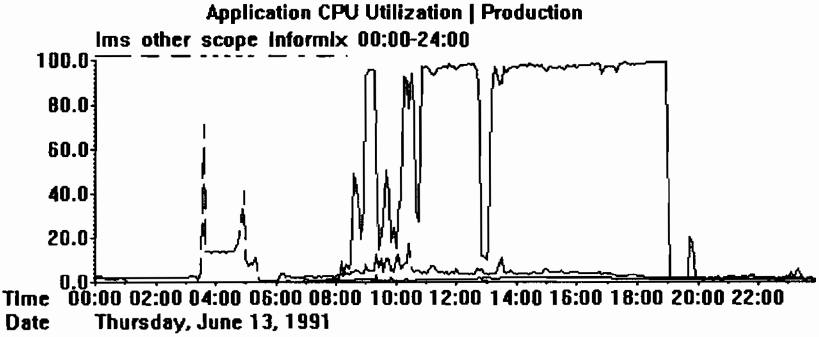


Figure 5

Figure 5 shows the same time interval but from an application perspective. The data for this graph was derived from the pre-summarized application records collected during measurement.

Reviewing application graphs in conjunction with the global graphs allows the capacity planner to select the most appropriate collection window from which to base analytic models.

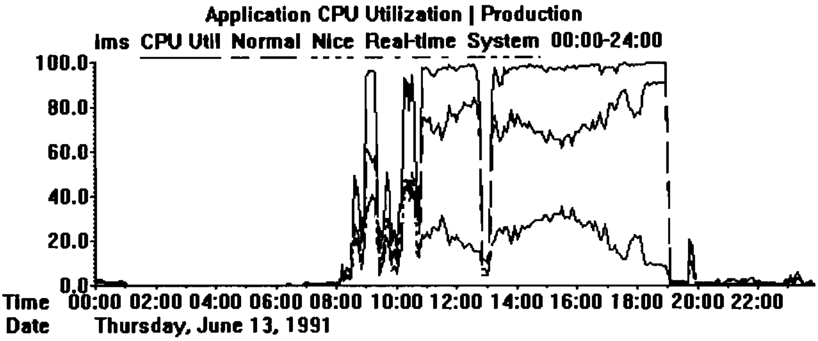
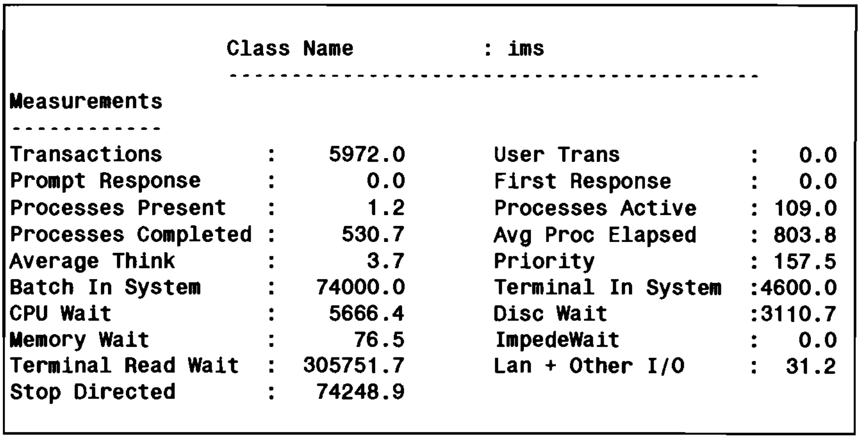


Figure 6

Figure 6 shows more detail of what the distribution of CPU resources were for the application of interest. Since the data was pre-reduced at collection time alternate views of the application data can be generated quickly. This methodology encourages interactive system analysis and modeling.

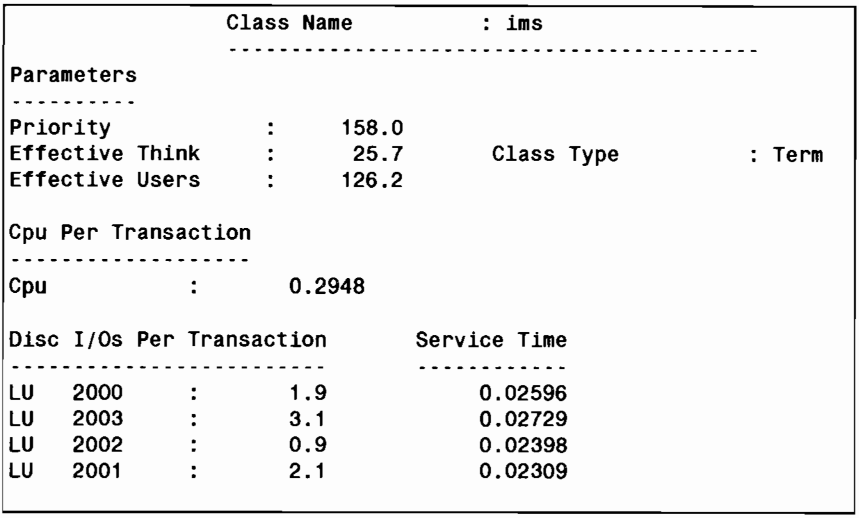


```
                  Class Name          : ims
                  ------------------------------------------
Measurements
------------
Transactions          :    5972.0       User Trans           :   0.0
Prompt Response       :       0.0       First Response       :   0.0
Processes Present     :       1.2       Processes Active     : 109.0
Processes Completed :     530.7       Avg Proc Elapsed     : 803.8
Average Think         :       3.7       Priority             : 157.5
Batch In System       :   74000.0       Terminal In System   :4600.0
CPU Wait              :    5666.4       Disc Wait            :3110.7
Memory Wait           :      76.5       ImpedeWait           :   0.0
Terminal Read Wait  :  305751.7       Lan + Other I/O      :  31.2
Stop Directed         :   74248.9
```

Figure 7

Figure 7 is subset of the data available from the pre-reduced application records. The IN system time measurements are used to calculated response times for modeling purposes.

The wait states allow the modeler to reference actual queuing delays for comparing model queuing delays. This can be helpful in the validation process.



```
                 Class Name          : ims
                 ------------------------------------------
Parameters
----------
Priority              :     158.0
Effective Think       :      25.7       Class Type           : Term
Effective Users       :     126.2

Cpu Per Transaction
-------------------
Cpu             :       0.2948

Disc I/Os Per Transaction      Service Time
-------------------------      ------------
LU   2000       :       1.9         0.02596
LU   2003       :       3.1         0.02729
LU   2002       :       0.9         0.02398
LU   2001       :       2.1         0.02309
```

Figure 8

Figure 8 shows an example of model input parameters that were generated from the pre-summarized application records.

Since the pre-summarization algorithms are designed with modeling in mind, models can be built very quickly, again encouraging interactive modeling exercises.

```
                          |   Response        |   Throughput      |   CPU
Application          pri  |  Meas  Model   %  |  Meas   Model  %  | Mea  Mod
------------------ -------+------ ------ ---+------ ------ ---+---- ----
other                155  |   3.5    3.6   3|    95     95   -0| 3.7  3.7
ims                  158  |  12.5   14.2  13| 11896  11399   -4|97.4 93.3
Measurement          157  |   0.5    0.5   4|   131    131   -0| 1.5  1.5
```

Figure 9

Perhaps figure 9 best represents the advantages of the pre-summarization technique. Since measurement data is collected with analytic modeling in mind, the modeler can very quickly build and validate models. Measurement and modeling data can be reviewed to determine if the appropriate modeling window was selected. For example, the **ims** application is well within accepted tolerances for model validation, as are the other workloads on the system. If the validation was unsuccessful the modeler can quickly create an alternate model derived from a different measurement window. The pre-summarization technique allows modeling tools to tightly integrate measurement and modeling data.

Summary

By basing system measurements on wait states rather than traces, we are able to implement some powerful techniques for parameterizing analytic models. The simplicity of the IN/OUT of system technique, combined with the pre-summarization methodology, allows a very low overhead continuous collection strategy to be implemented.

Continuous collections from which application specific models can be built, allows the capacity planner to pick the appropriate window from which to base models on. By viewing long periods of time the capacity planner can work with the users to select the interval which best represents the applications activity. This provides a significant advantage over detailed process collections techniques, since overhead is higher, and we are only able to collect data for short intervals. This limits our selectivity of the appropriate modeling window. It is a risky endeavor to base capacity planning models on non representative windows.

UNIX performance tool developers have the opportunity to significantly advance the state of the art. By leveraging UNIX wait state collection techniques, one can realize a significant reduction in modeling complexity while maintaining a high level of accuracy. This results in a reduction of the overall costs associated with modeling exercises, allowing capacity planners to spend more time focusing on the business aspects of their operations.

References

[Allen90] Arnold O. Allen, *Probability, Statistics and Queuing Theory with Computer Science Applications, Second Edition,* Academic Press, San Diego, 1990.

[Browning90] Tim Browning, *Forecasting Computer Resources Using Business Elements*, Conference proceedings of the Computer Measurement Group, (December 1990).

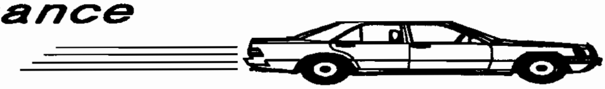
[Ferrari83] Domenico Ferrari, Giuseppe Serazzi, Alessandro Zeigner, *Measurement and Tuning of Computer Systems,* Prentice-Hall 1983.

[Lazowska84] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queuing Network Models*, Prentice-Hall, 1984.

[McGalliard89] James McGalliard, *Contracting for Capacity*, Conference proceedings of the Computer Measurement group, (December 89).

[Wade88] Gerry Wade, *Designing Performance Tools for the Every Day System Manager*, Proceeding of the 1988 Conference of HP Business Computer Users.

Performance Is Easy



or

(What the Gurus of Performance Don't Want You to Know)

by
Gerry Wade
Hewlett-Packard, Performance Technology Center
Roseville, CA

Interex Paper 2049

For years the art of Computer Performance has been the exclusive domain of highly trained specialists. The skills necessary to properly tune a computer system were portrayed as so difficult to learn as to be beyond the abilities of most people. This attitude was reinforced by the in depth training on operating system internals which was a prerequisite for studies into performance. While some tasks may still require this level of training, the vast majority of performance problems encountered today can be solved with a liberal application of common sense. This paper will attempt to outline a few strategies and concepts which can be used by the common computer system manager to get the most performance out of their systems using their own resources. There may still be situations which require the in depth training of the specialist and these will be illustrated. Even if a specialist is utilized, the common sense approach to a problem can serve as a good "bounds" or "reality" check on their work.

The Traditional Approach to Performance

When computer systems were still being developed, there were two types of programmers. The application programmers would be concerned with one or more programs which performed the "useful" work, that work for which the computer systems were purchased. Then there were the system programmers. They were concerned with the programs that performed the "necessary" work, the behind the scenes work that managed resources and made it possible for the application programs to peacefully share the same computer system. Let's face it, how long would a business STAY in business if it specifically purchased computer systems because they had a great memory management scheme? On the other hand, if a system didn't manage memory well then it couldn't handle the demands of the other programs efficiently and the cost of the computer systems would be too great to use.

[Have I alienated everyone yet?] The rivalry between application and system programmers is a long standing and heated one with each group claiming to be indispensable. The application programmers accuse the system programmers of not understanding end users or anything about the "real world" while system programmers accuse application programmers about not understanding how their programs impact the rest of the computer system and of being "shallow" and too "narrowly focused".

What does all this have to do with performance? I'm coming to that. Consider the situation where the computer systems are performing well for a time. Encouraged, the business decides to computerize even more of its tasks for even greater savings. (sound familiar?) After a while the once great performance of the system starts to degrade. Shocked that their once perceived omnipotent computer systems are actually mortal, the business asks their programmers to "fix it". I will just let you imagine the world class "finger pointing" war that ensues between the application and system programmers. ("My programs run great! They were performing just fine until all that other stuff was added so it must be a problem over there"! "Those programmers don't have any idea what they are asking this system to perform. If they just wrote more efficient programs then we wouldn't have any problems"! etc. etc. etc.)

It seems that the only place to solve a problem which affects the entire system is with the system programmers. Indeed, they are the ones who understand all that magic stuff going on inside the system so they should be able to get maximum performance out of it. While this might actually have been the right solution, it brought with it some excess baggage which we are still carrying today. Specifically, a system programmer was taught to solve a problem by first understanding it to a great depth. Understand how each piece interrelates with other pieces. Gather as much information about the situation as you can possibly gather. Formulate a theory that fits the measurements as to why the system is behaving the way it does. Finally formulate a solution which would make the theoretical model perform better, then recommend that action be taken against the real system.

Now don't get me wrong, I'm not saying that the approach of the system programmers is a bad one. I am merely saying that it has had some bad side effects. Specifically, the worth of such a programmer (now called an "analyst" since many did not write programs any more, just analyze existing ones) was based largely on the depth of understanding they possessed of the inner workings of the systems. It was to their best interest to make the process of solving performance problems appear to be quite difficult, and to protect their knowledge at all costs. Many times this protection was not even conscious, but rather an attitude they obtained while apprenticing themselves to their jobs. Since they had to learn all the system internals before studying performance, and since the performance lessons all leveraged on that internals training, it was easy to think that performance was a most difficult topic which could only be properly understood after many years of dedicated study. After all, isn't that what THEY had to go through to learn it?

The Problems with Experts

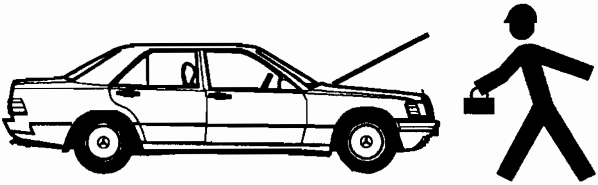
Guru's and Performance Wizards are expensive. They are expensive to train and even more so to keep current. As a result, not every business can afford to keep their own expert. Computer systems change. The detailed internals knowledge acquired by an expert must be continually updated or it can be rapidly outdated. Modern computer systems are increasingly complex. It is rare, if not impossible, to find someone who understands everything about all areas of the system internals. For that reason the experts tend to specialize in different areas. You may have one specialist on data bases and another on dispatching algorithms. While each specialist may know something about other areas on the system, they will focus on the one they know best.

This brings up the problem I call "The Specialist Syndrome". If you bring a problem to a specialist, they will most likely find the cause of the problem to lie within their speciality. This syndrome is not confined to computer systems. If you go to a foot doctor complaining about aches and pains in your back, they will no doubt find a problem with your feet which is at the root of your problems. The same problem described to a psychiatrist will turn out to be "all in your mind".

In computer systems, a performance problem will be interpreted based on the expertise of the analyst. It is not just a coincidence that most of the performance problems solved by the data base specialist turn out to be related to data bases. Please note that the Specialist Syndrome is often at odds with a phenomenon called the "It's Not My Problem" syndrome. This syndrome is the tendency of a specialist to shift the blame for any problem which they don't think they can remedy. (We can't let the expert mystique get tarnished by failure, can we)?

Practically speaking, the thing of which we must be wary is the tendency for a performance specialist to focus on areas where they have been successful in the past and try to make previous successful solutions fit each new situation. They can often overlook simple solutions by trying to force fit a solution from another problem.
(If you get an answer before you finish asking the question then you should beware falling victim to the specialist syndrome").

Do It Yourself Performance



What is the alternative to using experts for performance analysis?
The alternative is to do it yourself.
Isn't that Risky?
Yes, if you use the traditional approach (the approach used by the performance experts). BUT... (Are there any performance experts out there listening? NO, OK here comes the good stuff, the stuff the experts don't want you to hear about):

Don't tell anybody that I was the one who told you this but Performance Is Easy! That's right, all these years you were convinced that it was some dark mystery and all along it was nothing more than common sense! You're not convinced? I guess all those years of training are hard to set aside all at once. Let me feed it to you gradually. First of all, let me say that there ARE some cases that absolutely positively require the skills of a trained professional (a performance expert). Feel better now? Next, let me further state that a large majority of the performance problems you face are NOT THOSE CASES!

Let me further state that you should not necessarily attempt to become a performance expert in order to take over their tasks. By that I mean that is not necessary to pick up all the excess baggage they carry such as training on operating systems internals. Most of this stuff would just get in the way anyway. I believe it was that famous Karate instructor Mr. Mieygi who said something like "Know no Karate - good,
Know much karate - good, but Know little bit Karate - very bad". In other words, a little bit of knowledge can be a dangerous thing. It can lure you into thinking you understand the entire situation when in fact you don't. To paraphrase Mr. Mieygi let me say that you should either become a performance expert or not, don't try to be a little bit of an expert.

With that in mind, assuming that you are not now off reading the operating system internal manuals and brushing up on your machine instructions, let me give you some guidelines.

- You don't require vast amounts of internals knowledge in order to do performance. Often a general understanding of the types of rules used by the operating system and how you can interact with them is sufficient.

- You don't need massive volumes of data in order to analyze performance. You will find that a few well chosen metrics are all that are necessary in most cases.

- You don't need complex performance tools to measure what is happening on your system. In many cases you have the means to obtain the performance metrics you need readily at hand.

- Complex solutions are rarely correct ones. You will find that the best solutions to a problem are usually quite simple and easily understood if stated in simple terms.

- Often simply stating the problem in a clear and correct way can suggest the proper solution. Understanding what you want a computer system to do for you is often the biggest part of the problem.

A. Decide what performance means to you.

Does good performance mean no phone calls complaining about poor performance? Maybe it means getting 24 hours worth of work done in 24 or less hours time. Perhaps the real issue is reducing personnel turnover or lost sales due to slow system response times. Probably it is some combination of these and other situations which is truly important to your success. You will probably find that different users of the computer system will have different ideas of good and bad performance.

If you don't have a clear understanding of what good performance on your computer system means, then you will have no way of knowing when you achieve it. Moreover, the more concrete and measurable the statement of desired performance is, the better you will be able to measure your progress toward, or away from, it. You can see how much better it is to have a goal of "2 second response times on the order entry application", or "all night time batch jobs to be finished by 6:00 AM each day" rather than "no one is complaining". The reason no one complains may be that performance is good, or it may be that everyone has given up and left to find other work.

Be sure when you discuss the results of performance (response time, batch execution times etc.) that you also include the DEMANDS on that performance. For example, you should state a goal of 2 second response time for order entry where orders are being entered at a rate not to exceed 1000 per hour. Or, the nightly batch jobs will finish by 6:00 AM so long as the main data base does not exceed 10 Million line items in size and the nightly reports are less than 20,000 pages in length. Failure to link performance to demands will leave you vulnerable to increasing demands with no recourse to address performance issues. If the nature of your business demands that the performance be maintained in spite of changing demands, then make sure that everyone understands that as the demands change, the computer solutions may also be required to change.

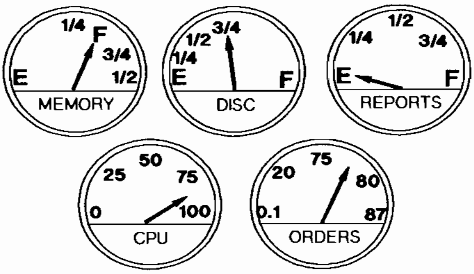
Once you understand what the desired performance is, you should communicate it to all concerned and determine if this is acceptable. I caution you that you may receive some negative feedback at this point so you should be ready to "calibrate" everyone. For example, if you just ask someone how fast they want the computer system to be then you will probably get an answer like "as fast as possible" or if you have a few smart Alecks "I want answers before I ask questions". I once saw a technique which worked wonderfully to get everyone reasonable. A computer sales representative was dealing with a customer who had the aforementioned attitude ("I want a computer system that goes infinitely fast, costs nothing, and will wax my car during the lunch break"). The sales rep. simply said "Speed costs money. How fast do you want to go?" In other words, you can have very high performance but at a very high cost. Or, you can have very low cost but with corresponding low performance. Cost and performance are linked and all you can do is choose the most appropriate level to suit your needs and pocketbook (and you have to wax you OWN car).

If you are discussing performance desires with those responsible for the budget then you shouldn't have any problems arriving at a reasonable performance goal. If you are discussing performance goals with someone who is not responsible for the budget, then you should probably talk in terms of "What is the worst performance you can tolerate and still get your job done?" Assure them that you will be trying to achieve the best performance possible, but you need to know at what performance level the business starts to suffer.

Once you get a good understanding of what your performance requirements are, then write them down. If you don't then you will no doubt find that they are changed when you come back later to examine them. It is always harder to hit a moving target...

B. Find "gauges" to measure demand and performance

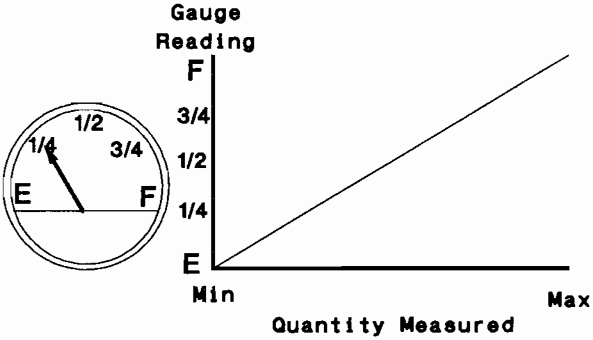
Gauges



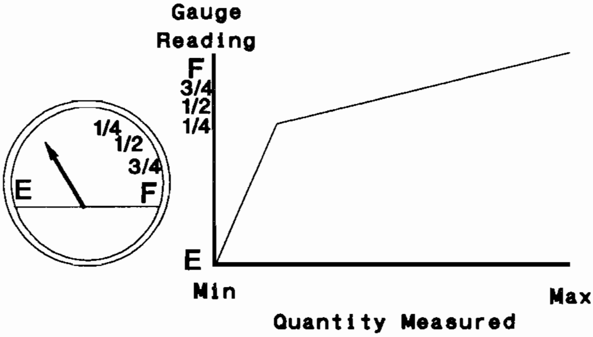
Once you decide what demands and performance levels are acceptable, you will need some way to accurately determine how your system is performing with respect to those levels. You will need a "gauge" for each demand and each performance metric in which you are interested.

Some basic guidelines for choosing gauges:

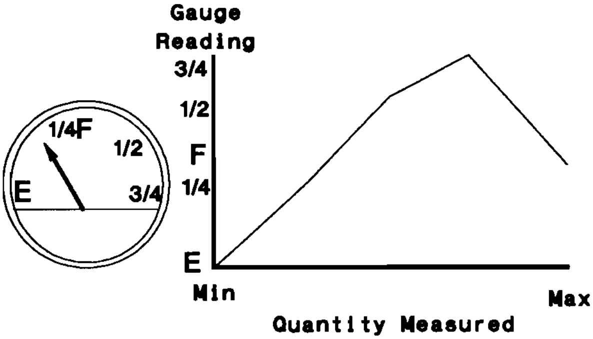
- A gauge should give you a quantifiable measure. Avoid gauges which give readings such as "pretty good" or "not so good" or "terrible" if you can find ones with a finer gradation such as "57% utilized" or "100 orders per hour". Avoid the temptation to go overboard in this area. Knowing a resource was 57% utilized will be as useful as knowing it was used 57.122345% of the time. The extra precision in the number is not helpful and can actually imply more accuracy in the predictions based on it than is justified.



- A gauge which responds linearly with respect to the item it is measuring is ideal. The units of the gauge are not important so long as it is easy to determine how much of the measured item is used and how much is left.



A non linear gauge can be used if no linear one is available. Care must be taken with such a gauge to avoid mis-interpretation. Having a gas gauge on your car which sits close to Full until you have only 1/4 tank left then drops suddenly is not an ideal gauge.



Even less useful than that is a gauge which "doubles back" on itself, with the same reading registering for more than one value in the measured item. In the example above, what is the quantity measured if the gauge is pointing straight up? (It could be either 1/4 or Full and you can't really tell which). This may seem an obvious guideline, but you might be surprised how many "experts" are still using such gauges today.

- Avoid extra gauges. Having two measures of the same metric is something like asking a man with two watches what time it is. He is never quite sure... If you do end up with multiple gauges, decide which one best fits your needs then use the others just for "backup" and not for quantitative measurements. (For example, if the primary gauge for memory usage says the system is short on memory, then a secondary gauge can be used to confirm the analysis, but you should not START with the second gauge).

- Avoid measuring metrics which you don't need. The tendency when performing performance measurement is to assume that "more is better". Indeed if you ask any performance expert, they will confirm this for you. Remember that the expert's philosophy of "gather all the info you can then try to fit a theory to the data based on a thorough understanding of how the system works". You will be able to take a much simpler approach to solving performance problems and extraneous gauges can distract you.

For example, how many of you have a vacuum gauge in your car? Ask any auto specialist and they can tell you many reasons why you should have one and the many wondrous things it can tell you. They can back up their claims by placing such a gauge on you car and telling you the state of many of the major systems in your automobile, just by how the gauge needle twitches when you step on the gas. If this is such a wonderful gauge then why doesn't every car have one? At one time Detroit DID put a vacuum gauge in many of their new cars. They found that drivers did not like them, they bounce up and down a lot and are constantly drawing attention to themselves, even when nothing is wrong. In order to draw proper conclusions from a vacuum gauge you must be able to visualize how the many different components of an automobile engine interact under different situations. In other words, a vacuum gauge is an inexpensive, very useful gauge for an automotive specialist, but it is less than useless to the ordinary automobile driver since it distracts them away from the gauges they do understand.

You will find many "vacuum gauges" in the computer performance world, and many performance experts that swear by each one of them. It isn't that these gauges are bad, merely that they are not appropriate without the training of an expert to back them up. For now you need to avoid the distraction they bring to your job.

- Specific gauges will be discussed later in this paper. For now you should think in terms of a gauge to measure each type of demand you will place on the system (Active interactive users, batch jobs per day, report pages requested, etc.) and a gauge for each result you are measuring (interactive response time, batch job execution time, report turn around time etc.). You may also find it useful to gave a gauge to measure the critical computer system resources which may constrain or bottleneck overall performance such as CPU, DISC, MEMORY and software locks and latches.

C. Adopt a Performance Strategy for the Future

If you only worry about performance when it's bad then you will spend your life being managed by performance instead of managing it. You should develop a strategy for performance management which warns you of problems before they become critical. It should allow for you to learn about your system, how it performs and specifically how it reacts to changes you make to it. This really isn't as hard as it at first seems. All it really takes is a little discipline and planning. If you have done steps "A" and "B" then "C" becomes rather easy.

1. - First, begin logging the demands and responses on your system as early as you can. Keep only as much detail as you need since too much detail will be difficult to analyze later. At least keep the primary values from your gauges every so often.

(How often should you take data and for how long should you keep it?) Here is a general rule of thumb. You shouldn't attempt to make predictions of a time scale which is shorter than the time between data points. Therefore, if you want to predict performance on a day by day basis then log data at least every day. If you want to make separate predictions based upon the hour of the day then you will need to log data at least every hour.

The second rule of thumb is that you should not attempt to make predictions which are further in the future than you have data in the past. For example, if you want to predict a month into the future then you should have a minimum of one month's data logged on which to base that prediction.

Combining the last two rules of thumb, if you want to predict a year into the future with resolution accurate to the hour of the day then you will need a minimum of 8760 (365 days times 24 hours per day) data points logged. At every data point, log the values of all your selected gauges, both demand and response metrics.

2. - Second, keep track of each change you make to the system which might effect performance. Especially note any changes which were made to improve performance. You should be able to measure the effect of each change by examining the performance data before and after the change was made.

3. - Before making the next change related to performance, decide what it is that the change is designed to do. Try to state the effect of the change in terms of the performance metrics you are collecting if possible. State also any side effects which may be caused by the change (See the section of TRADEOFFS later in this paper). Write down your predictions of your system's reaction to the change.

4. - Make the change to the system. If more than one change was desired then try to make each change independently if they are not interconnected. This will allow you to evaluate the effectiveness of each change rather then just their combined effect.

5. - Compare the system's actual behavior to the behavior you predicted. Ignore any minor differences, but try to account for any major ones. Especially note if the change had no noticeable effect or had an effect in the wrong direction.

6. - Adjust your understanding about how your system responds to your demands using the new data points.

IMPORTANT: It is this step which increases your KNOWLEDGE of your computer system. It is the only way true understanding of how your system performs given your demands on it!

7. - Now form a new theory about how to improve performance and, if needed, return to step 3 and repeat this cycle. Stop whenever performance is satisfactory or the cost to improve performance further is not worth the gain.

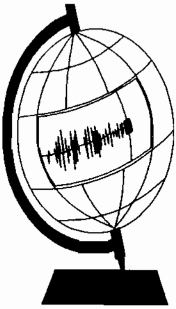
8. - If you can form no acceptable theories to improve performance and additional performance is still necessary, call for professional help. You will be prepared to shorten their task considerably by stating clearly what your goals are, and how you measure success. Make sure the expert knows you plan to measure the effectiveness of their suggestions and they agree with your measurement techniques. Ask for information as to WHY a particular recommendation was made, in plain language, so you can better handle this situation in the future.

You should find that you may need to rely on the performance experts fairly often at first, then as you gain an understanding of your system, their assistance will be required less and less often. Even so, don't hesitate to ask for a second opinion on any performance theory you have which you don't feel completely comfortable with or which will be especially costly to implement. [I told you that you would still have situations where traditional performance expertise would be required, didn't I? Well, this is when it is necessary].

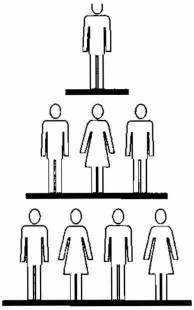
Now for a not so obvious observation:

As you monitor and record the response of your system to varying demands placed upon it, you will begin to develop a feeling for what is "normal" and when it starts to develop problems. If you then watch the indicators you are logging, you will be able to predict as well as any expert just when your system will begin to experience performance problems. You may even be able to predict how to adapt to those situations in time to AVOID having performance problems. At this time you can consider yourself in control of the performance on your system.

Types of Performance Data:



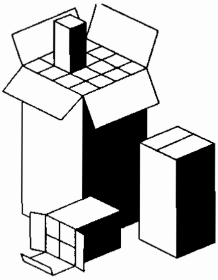
You can look at performance from several points of view. The first view is that from the overall system itself. This is usually called the **"GLOBAL"** point of view and it concerns itself with measuring the key potential bottlenecks on the computer system. If a resource which is shared by all users of a system is exhausted then probably all users will be negatively impacted. For this reason a global performance analysis is usually performed first. The most common global resources are: CPU, DISC, MEMORY, and Software Locks and Synchronizations.



The second way to look at a system's performance is to examine the performance of a single user (or **PROCESS**) on that system. It may be possible for individual processes to be experiencing vastly different performance on the same system, at the same time. If one process was able to monopolize all the CPU resources, for example, it might be getting superb performance while other processes had degraded performance as a result
Examining a single processes performance is usually no more difficult than examining the global performance metrics.

A process will consume quantities of the system resources. It will also have to wait for those resources if they are in use by other processes. At the process level you are concerned not only with how much of the key resources it USES, but also with how much time it spends WAITING for resources.

The complication with PROCESS data analysis is that typical systems like MPE and UN*X will probably have hundreds or even thousands of processes active at any point in time. If you do not know beforehand which processes you want to monitor, then you will have to log data for all of them. The volume of process data can quickly overwhelm your ability to store or analyze it.



The third way to examine computer system performance is by grouping the activities of similar processes into **APPLICATIONS**. In this way you can deal with an entire class of processes collectively instead of dealing with each process independently. You will usually want to improve the performance of an entire class of processes (say all order entry programs) rather than just one process out of thousands. By combining the activities of related processes we can isolate problems which might be unique to particular programs (such as data base locking situations which affect only programs which lock that data base).

While global and process types of data exist in their own right, application data is merely the intelligent combination of process data for related processes. Since the number of applications is usually much smaller than the number of processes on a system, then application data can be more efficient to store than process data.

Resource Characterization:

CPU:

Characterizing CPU usage is easy. All you need to do is to determine what percentage of the time the CPU was being used.The CPU gauge goes from 0% when no CPU is used to 100% when every potential cycle of the CPU was utilized. This gauge is linear and well behaved.

You might want to consider that some CPUs process instructions faster than others do. If you plan on upgrading from one CPU to another then their relative CPU speeds become important. If you do not plan on changing CPUs then you can skip this exercise. If two CPUs process instructions in roughly the same manner, then their speeds might be comparable using the number of such instructions they process in a given amount of time. The term for such a measure is usually Millions of Instructions Per Second or MIPS. You may have heard the acronym referred to as a "Meaningless Indicator of Processor Speed". This is because it is often inappropriately used to compare one computer against another with a different architecture, and hence a different way of processing instructions. Indeed the actual number of instructions necessary to perform the same task can vary widely between computer system architectures. It is important, therefore, to use MIPS only to compare two processors of the same architecture which process instructions in the same manner.

You CAN use a "relative processor performance" metric to compare one processor against another architecture but too many factors enter into such a comparison to place much validity on the results. Ask an expert (or better yet, two or three experts) for help if this situation is yours.

Given that we decide we can define the processing power of two computer systems, can we predict the impact of changing from one to another? The answer is yes, sometimes. Let me illustrate:

If you had processor A, a 1 MIPS processor running 50% busy then
upgraded to processor B, a 2 MIPS processor, then how busy would you expect processor B to be?

If A is 50% busy at 1 Million instructions per second, it is executing approximately 500,000 instructions each second. The other half second it is idle.

Now, if we place the same demands on processor B which executes instructions at 2 million per second, it would complete the 500,000 instructions in only a quarter of a second and then remain idle for the remaining three quarters of a second. Processor B would then be only 25% busy given the same demands.

This sounds easy right? Well you're right, this much IS easy. Now what would happen if we took processor A running at 100% busy and then upgraded to processor B? B would be running at 50% busy right? Probably not! Whoa, back up! I can still do algebra and it says that processor B would be 50% busy performing the same work that processor A takes 100% busy to accomplish. That's right, what is missing is that usually when a computer system is 100% busy, there is some demand which is not being satisfied. With more resources available, that demand can now be satisfied with the result being that processor B would actually be MORE than 50% busy, BUT IT WOULD ALSO BE SATISFYING MORE DEMAND (it has a higher throughput).

This is another of those areas where I recommend you consult an expert to verify your conclusions. Before you decide to upgrade your computer hardware, usually a very expensive step, have an expert double check to see it you are upgrading the proper components to the proper level.

DISC:

Characterizing disc resources is a bit more difficult. First, you have to decide whether it is the disc CAPACITY or it's THROUGHPUT which affects your performance. The capacity of a disc is the amount of storage it has, usually expressed in Megabytes. Running out of disc space can impact system performance. Indeed, it may even stop some of your applications from executing. (Someone once said that "the worst performance was no performance at all").

If the disc capacity is sufficient, then disc throughput can still cause a performance problem. Throughput is the rate at which data can be moved to and from the disc drives. Disc throughput is usually expressed in terms of transfers or IOs per second, or Megabytes per second. Since most computer systems can not operate on data directly on a disc drive, programs must wait for the data they desire to be transferred from the disc into main memory (or transferred from main memory out to the disc). The time a program waits for disc involves several factors.

Normally, data transfers between disc and main memory are grouped into discrete units. One IO will transfer a block of data at a time. The characteristics of the disc devices will force a wait for each transfer, to position the mechanical disc components, plus an additional wait while the data block is actually copied to or from the disc. The time required for a single IO will then be:

Service Time = Time to Position + (Bytes of data / Transfer Rate)

For example, to transfer a 1000 byte block of data to a typical disc it would take 25 milliseconds to position the disc plus (1000 bytes divided by 1000 bytes per millisecond) 1 millisecond to transfer it for a total of 26 msec..
To transfer 32,000 bytes of data in a single IO would take 25 msec plus (32,000/1000) or 57 msec.

What kind of gauges can be used for disc resources?

Disc capacity might be a simple 0-100% gauge for percent full, assuming all the disc space was available to whatever programs required it. If your system partitions disc capacity then you will probably want to have one gauge for each partition.

Disc throughput can be gauged by counting total transfers (IOs) per unit time. This will be a fairly linear gauge but you won't know the maximum value. Indeed the maximum number of IOs per second will depend upon the size of the transfers which can vary over time. Gauging by total bytes per unit time might be slightly better, but the maximum transfer rate can still be drastically affected by the size of each block transferred since there is a wait time for each block which is independent of block size. If the size of the blocks transferred to and from disc is fairly constant on your system, then either of these gauges can be used. You will have to determine the maximum gauge reading either empirically (watch the gauge under heavy load and see how high it goes) or theoretically. In either case be cautious of changes in systems or configurations which alter the size of the blocks transferred to disc.

<u>CASE IN POINT:</u> MPE/V transfers an average of 1 KBytes per block to and from disc. The rule of thumb is that a disc drive can perform about 25-30 transfers per second or 35-30 KBytes per second. MPE/XL transfers an average of 32 KBytes per block to disc. Would you be surprised to discover that the same disc drives on MPE/XL don't normally have a throughput of more than 10 transfers per second? How about if, at the same time, those same disc drives had a throughput of 320 KBytes per second? Would you say that the MPE/XL disc drives were running at 1/3rd the throughput of MPE/V (10 IO/sec vs 30 IO/sec) or would you say MPE/XL has ten TIMES the throughput of MPE/V (320 KBytes/sec vs 30 KBytes/sec). Both statements are correct, but very confusing!

Let me suggest a different gauge to use for disc throughput. Remember that an ideal gauge is linear from minimum and maximum and the minimum and maximums are known (like 0-100% busy for CPU). How about if we measure the amount of time that a disc drive is "busy" (unable to accept a new transfer request) versus the elapsed time. If we express the disc busy time as a percentage (busy time divided by elapsed time) then we have a **DISC UTILIZATION** metric which is linear from 0% to 100%. If a disc's utilization is 0%, we know that it is not doing anything. If it is 100% then we know that it is doing all it can and requesting any more data transfers to or from it will not make it go any faster. Disc utilization balances all factors affecting disc transfer times.

Extra Credit Homework Assignment:
Go back to your office and find the person most knowledgeable about your disc drives (or call your vendor CE). Now tell them that you intend to measure how busy a disc drive is by measuring what percentage of the time the computer system will not allow you to start another disc transfer to the disc. [DUCK] You should receive a stream of verbal abuse informing you that it is not FAIR to make such a measurement since it includes the software overhead in the operating system drivers, channel queueing delays and many other such things. They may insist that you should use the disc data sheet's specifications for average rotational delay, random seek times, and peak burst transfer rates instead. This technique will eliminate any overhead caused by operating system inefficiencies and so forth.

I don't recommend you continue the discussion (argument?) any further at this point. Thank the CE for helping out with their opinion. The reason we went through this exercise, if you were dumb enough to really DO it, was to illustrate that the programs you are interested in must wait for those operating system inefficiencies, queueing delays, and other stuff and so from their point of view it SHOULD be included in the time it takes to do a disc transfer! Disc Utilization is a measure from the program's point of view, and will most likely not be the same as the utilization of the physical hardware itself.

REAL Extra Credit Assignment:
Disc Utilization makes a very good gauge for measuring the throughput activities of a single disc drive. What should you do if your system, like most systems now days, has more than one disc drive? HINT: You CAN use one gauge for each disc drive, but that can be a lot of gauges. Is there some way to make a proper gauge which represents the throughput activities for all disc drives combined?

MEMORY:

Main memory utilization gauges used to be simple. You measured how much main memory was used versus the total for a 0-100% gauge just like CPU and DISC utilizations. HOWEVER ... Almost all modern computer operating systems have instituted a **VIRTUAL MEMORY** scheme. Virtual memory uses disc capacity as a slower, but much larger and less expensive, extension to main memory. Programs and data can not actually be accessed directly on the disc drives, but must be transferred into main memory when needed. Since not every part of every program is needed all the time, virtual memory can make it appear that your computer main memory is much larger than it really is without significant loss of performance.

With virtual memory schemes, it is not unusual for main memory to be 100% utilized with little or no degradation in performance due to memory constraints. (So much for the SIMPLE gauge). Measuring how full VIRTUAL memory is, while interesting, does not properly indicate when a lack of main memory will cause a performance problem.

What happens to a computer system when it does not have enough main memory to perform efficiently? If a program requires access to data which is not in main memory, it must be transferred from disc (swapped in). If main memory is currently fully utilized, then some other piece of a program or data area must be copied to disc to make room (it is swapped out). Swapping data in and out of main memory consumes two other system resources, it performs disc transfers which adds to the activities on the disc drives, and it consumes CPU resources to manage which pieces of data are in main memory and which are swapped out to disc. The virtual memory scheme will impact the program which is requesting data from disc by making it wait until that data is swapped in. It will also impact other programs on the system with the additional demands on system CPU and disc devices.

As long as a system has "enough" main memory, the overhead added by the virtual memory scheme is not high enough to impact overall performance significantly. Whenever a system has less than "enough" main memory, then the delays and additional CPU and disc consumption can make a dramatic impact on overall performance.

Now for the $64,000 question: "How much memory is 'enough'?"
A common, only semi-comical response to computer performance problems used to be "take two megabytes and call me in the morning". The reasoning was that if the system was performing poorly, it might be caused by having too little memory. Having too much memory never hurts, so keep adding memory until you are sure you have 'at least enough'.

You don't like that approach? How about the quasi-scientific approach. Postulate that there is some optimal "minimum working set size" for each program. If the program is afforded at least this much main memory then it will perform reasonably well. Now, if you add the minimum working set sizes for all programs on the system, add in the requirements for the operating system itself, and THAT is precisely how much main memory the system should have! This sounds good and it might even work IF...

IF the minimum working set size can be measured for each program.

IF the minimum working set size for a program is constant over time.

IF the mix of programs is well known (and constant?).

IF the operating system requirements can be determined.

IF processes SHARING code and/or data can be accounted for.

THEN it might work.

Not to be too cynical, this approach DOES work on some computer systems. Unfortunately the majority of today's multi-user multi-processing operating systems like MPE and UN*X don't yield to this approach very often.

If you can't measure the utilization of main memory directly, then what is the alternative? How about measuring it indirectly? You know that when you have enough main memory, the majority of data that programs require does not have to be swapped in (at least not once it has been swapped in the first time). If you do not have enough main memory then it is impacting your system's performance by forcing programs to wait for memory resources, and by swapping data to and from disc.

We could measure how much swapping is going on if we can separate the number of disc transfers which are for swaps versus those which are for normal user data (disc files). One memory gauge would be the number of swaps (or disc transfers) which are being performed in a given time. Like the disc throughput gauge based on transfers per second, this gauge starts at zero but has no upper limit. We would have to set some empirical limit on the number of swaps per second before we observed system performance problems due to memory size.

We could measure the amount of CPU consumed while managing virtual memory. This is CPU consumed for a "necessary" but not "useful" purposes (that is, it does not contribute directly to accomplishing the program's goals but is necessary due to main memory size constraints). We could also set an empirical limit on how much CPU was being consumed by the virtual memory activities before memory problems were probable.

If the operating system can detect when a program is forced to wait for data to be swapped in from virtual memory, and can make that data available as a process level metric (say the percent of time a process spends waiting for memory), then we might construct a gauge showing the average time processes are waiting for main memory resources. This might be a very useful gauge but not all operating systems are instrumented well enough to provide this metric. NOTE: This metric becomes even more important after we discuss the effects of buffering and anticipatory overlapped disc transfers in the tradeoffs section of this paper.

OK, memory gauges may be a bit difficult to obtain. We will most likely have to resort to measuring main memory usage indirectly, by the impact of virtual memory schemes on CPU and DISC utilizations. Even so, the proper amount of main memory will best be obtained empirically, by running the system until it has a detectable memory shortage (this sets the critical size for the given workload) then adding enough main memory to remove the shortage. I guess the original idea of "buy as much as you can afford so you won't HAVE a problem" isn't such a bad idea after all is it?

SOFTWARE LOCKS AND SYNCHRONIZATION:

It is possible for a group of programs on a computer system to have a performance problem even though no system resource is being over utilized. I have seen systems where the CPU was 20% utilized, each disc was less than 40% utilized, and no virtual memory swapping was occurring, and still the response time of the user programs was terrible. Tests showed that it wasn't just a poorly written application program (also known as a "hog") since with only a few users running the application, response time was fine. What is the problem?

The problem in this case turned out to be that the application programs were all locking the same data base in order to perform their work. Since only one program could have the data base locked at a time and furthermore since no program could perform any work unless the data base was locked, then necessarily only one of the application programs was allowed to execute at a time. If that program was waiting for a disc transfer then the CPU was left idle since no other program could use it until the first program unlocked the data base. When the program with the data base locked was using the CPU, no other program could have a disc transfer in progress since they didn't have the data base locked. Each program could run given this scenario, but only one at a time. The throughput of the system was being constrained by the software mechanism of locking a data base.

Before you say that all software locks should be eliminated, let me remind you that such locks are often required. If you allow two users to update the same item in a data base at the same time, then strange and not so wondrous things happen when you try to balance the data in that data base. Problems occur when the locking being performed forces too many programs to wait which could otherwise continue processing.

How many different software locks and synchronization mechanisms are there? Hundreds, or even thousands! Actually there is no practical limit. Is there any hope of constructing a gauge or gauges to measure program delays due to software locks? Yes and no. Yes, we can probably come up with one or two gauges which will show us in general how many processes are waiting for such events. We might even have some special purpose ones to break out waits for data bases from other types of waits. It is probably not reasonable to have a gauge for every different possible software lock which can exist on a system. (That's too many gauges to watch, even for a computer system).

How about if we pick a gauge for software locks which works like the last one discussed for memory? If we can determine whenever a process must wait for such a lock, and then count how much of its time it spends in such a state, then we can measure the percent of a processes life which it spent waiting for software locks. From a global prospective, this might translate into the average amount of time a process spends waiting for software locks, or the average number of processes which are waiting for software locks on the system. Please note that we are not interested in how many processes are HOLDING locks since locking an unlocked resource is usually very fast, we are merely interested in those processes which can not execute because someone ELSE is holding the resource when they need to lock it.

<u>A Sideline:</u> On MPE/V and MPE/XL systems, processes which are waiting for software locks and synchronizations are often called "IMPEDED" or "in the IMPEDE wait state".

QUEUEING FOR A RESOURCE

Now that we have an idea of the types of resources which can constrain a computer system's performance, we need to examine what happens when the resource we need is not available. In some cases, if the needed resource is not available then the requesting process is terminated. An example of this is running out of disc capacity when a program needs to create a disc file. More often, if the resource is likely to be available in a short period of time, then the process is temporarily suspended to wait for it. If more than one process is waiting for the same resource, they are said to be waiting in the queue for that resource.

Queues can occur for any resource, CPU, MEMORY, DISC and even for SOFTWARE LOCKS. Waiting in a queue delays the execution of a process so excessive queueing can be an indication of performance problems. There are many very good books on the subject of queueing theory and computer performance modeling based on queues. Most statisticians can give you a precise answer as to how long you will have to wait in any given queue. Listen carefully when they explain how they got their answers and you'll hear phrases like "If we assume..." and "normal distribution" and "exponential arrival rate" and so forth. These aren't necessarily invalid assumptions, but they aren't necessarily valid ones either. For now we need to understand only a few general concepts to work some magic.

First, if a process needs a resource and that resource is available, then no queueing occurs. It is only when a resource is being used by another process that a process may be delayed by waiting in a queue.
Question: If there is only one process running on the system, can there be any queueing delays?
How long a process waits in a resource queue depends on how many processes are in the queue ahead of the current process (the **"average queue depth"**), and how long each one of them will keep the resource occupied once they get it (the resource's **"service time"**). The average amount of time a process must wait to use a resource, then, is:

Queue Wait Time = Average Queue Depth X service time

How about a few examples?

If the average process spends 1 second of CPU time for each transaction, and if the average number of processes waiting for CPU at any given time is 15, How long should you expect a process to take to get its needed CPU time?

Queue Wait Time = 15 X 1 or 15 seconds
Add the time this process actually USES the CPU and the answer is
16 seconds! (close enough).

How about if the average time it takes to perform one disc transfer is 40 milliseconds and a process needs to perform 10 IOs. If the average queue depth for the disc is 5, how long must the process wait to get its IOs done?

Queue Wait Time = 5 X 40 milliseconds or 200 milliseconds.
Add the time to perform 10 IOs (10 times 40 msec = 400 msec)
for a total time of 600 milliseconds. Right?

Wait a minute, this assumes that the process only has to stand in line for its FIRST IO, then it gets to do all the rest of its IOs without standing in line any more. Since most disc drivers don't queue by processes, but rather queue by the IO request, probably a better estimate would be:

Queue Wait Time = 5 X 40=200 msec FOR EACH IO
Add the time to perform ONE IO (40 msec) = 240 msec TIME FOR EACH IO
Now multiply times the required 10 IOs = (10 X 240) or 2400 msec!

Time out for a little algebra: (remember high school??)

The three quantities, THROUGHPUT, UTILIZATION, and SERVICE TIME are all related. Service time is the average amount of time it takes to do one unit of work. Throughput is the number of units of work being done in a given amount of time and utilization is the fraction of time that work was being performed over a given amount of time. Stated algebraically:

```
SERVICE     = BUSYTIME / UNITS
THROUGHPUT = UNITS     / TIME
UTILIZATION= BUSYTIME / TIME
```

Now observe that

```
BUSYTIME    =  SERVICE X  UNITS
UTILIZATION= (SERVICE X  UNITS) / TIME
UTILIZATION=  SERVICE X (UNITS  / TIME)
UTILIZATION=  SERVICE X  THROUGHPUT
or
SERVICE     = UTILIZATION / THROUGHPUT
or
THROUGHPUT = UTILIZATION / SERVICE
```

If you have any two of the three quantities, then you can calculate the third.

Let's test this concept. If I had a gauge which showed a disc drive to be 80% utilized, and another gauge which showed that same disc to be performing 20 transfers each second, then how much time does it take the disc to perform each transfer (what is its service time)?

SERVICE = UTILIZATION / THROUGHPUT
= 80% / 20 transfers per second
(watch your units here, 80% is really 0.80)
= 0.80 / (20 transfers per second)
= (0.80/20) seconds per transfer
= 0.040 seconds per transfer
or 40 milliseconds per transfer.

Now does this make sense? From what we know about our disc drives, it takes about 25 msec to position and then 1 ms to transfer 1 KByte of data. 40 msec total time - 25 msec to position = 15 msec to transfer.
If we are transferring 15 KBytes of data on the average, then this all seems to be OK, otherwise something is wrong with one of our assumptions about the disc. If we knew the applications performed 1 KByte transfers, then we might have to conclude that the positioning of the disc was abnormally slow for some reason.

See how much useful stuff you can get out of a few simple gauges, some common sense, and a little high school algebra? (*And you thought this performance stuff was HARD*)!

These types of calculations work equally well for all types of resources where queueing can occur. You might try your own exercises on the utilization, service times and throughput of software locks, CPU, or memory.

Where Should You Get Your Gauges?

Now that we know something about the kind of performance Gauges we are looking for, how do we go about getting them? There are many sources of performance data on most systems, including some you may not have thought about yet. Here are a few options:

1. Best Buy: The best place to obtain useful performance data is close to the source. If you want to know how your important application programs are performing then the best way to get that data is to have the application itself collect it. Consider adding simple bits of code at the beginning and end of the code which performs a unit of work. (Remember that a unit of work can be defined by the application and might be an on-line transaction, or a batch update for example.) This additional code can collect the data for the performance Gauges you desire. At the very minimum I suggest you count the number of units of work accomplished, measure the CPU consumed for each unit, and measure the time it took to process each work unit. If the application waits for user input, separate the time waiting for an external event, like the user entering data, from that actually "in the system" carrying out the unit of work. Some sample code might be:

```
SUMUNITS := 0;              { initialize summary
variables }
SUMTHINK := 0;
SUMRESP  := 0;
SUMCPU   := 0;
TIME0    := TIMER;          { Start of user
THINK time }

{getuserinput}
TIME1    := TIMER;          { End of THINK,
Start of RESPONSE }
CPU1     := PROCTIME;       { CPU used BEFORE
this transaction}

{process the work, return the results}
TIME2    := TIMER;          { End of RESPONSE }
CPU2     := PROCTIME;       { CPU used at end of
transaction }

THINK    := TIME1-TIME0; { Individual
transaction data }
```

```
          RESPONSE := TIME2-TIME1;
          CPU      := CPU2 - CPU1;

          SUMUNITS := SUMUNITS + 1;{ Accumulate
          transaction sums }
          SUMTHINK := SUMTHINK + THINK;
          SUMRESP  := SUMRESP  + RESPONSE;
          SUMCPU   := SUMCPU   + CPU;
          TIME0    := TIME2         { End one
          transaction=start next }
          {return to getuserinput}

          ELAPSEDTIME := SUMTHINK + SUMRESP;  { Finish
          Program }
          WORKGUAGE   := SUMUNITS / ELAPSEDTIME;
          CPUGUAGE    := SUMCPU   / ELAPSEDTIME;
          AVETHINK    := SUMTHINK / SUMUNITS;
          AVERESP     := SUMRESP  / SUMUNITS;
```

Simply by accumulating the summary information and logging it periodically you can gain a lot of insight to the performance of the application and the system on which it is running. You may want to create library routines to START a transaction, STOP a transaction, and LOG SUMMARY results. These routines could then be easily incorporated into all your applications.

2. Second best buy: Measure the user work accomplished. This may be by instrumenting the program as in the above example, or by examining some other information. For example, if the order entry application kept a running order number which was incremented for each order processed, then you can interrogate the next order number at the end of each day to determine how many orders were processed that day. You may examine the current JOB and SESSION numbers each day to determine how many jobs and sessions logged on during the day. If nothing else then at least do a SHOWJOB every so often and record the number of users logged on to your system. Record the amount of work accomplished as the demand, or work unit gauge.

If you can't measure every unit of work done by the applications, you might be able to record a representative performance measure occasionally throughout the day. This can be as simple as doing one transaction and timing it with your watch, or it can be done automatically by having a special program perform one "dummy" transaction periodically and log the amount of time it took to complete. Many people use this dummy transaction technique for a single number performance management tool. (One gauge might be enough if it relates well to overall system performance).

3. Freebies: Many sources of performance data may be found free on the computer system you are using. Many systems will keep a summary of the amount of CPU consumed by different users (MPE REPORT command, UN*X accounting functions for example). You may also be able to enable system logging of key events. The MPE system logs can be a free source of valuable performance data if you only take the time to retrieve the data from them. Even certain system commands can be useful. I already mentioned using commands to show the number of users logged on each day. You may also find information on disc caching data comm, etc. all just one command away. You may also obtain much information by the simple process of observing. Many systems have a CPU usage indicator which can provide a very rough idea of overall CPU utilization. Many disc drives will blink a light when they are accessed. Some system managers have even been known to walk around the production floor and quietly time the response at their terminal users during different times of the day. At the end of a batch job, the amount of CPU used and the elapsed time for that job are often listed.

4. Off The Rack: You may purchase a performance tool to provide you with the performance metrics you need. In shopping for such a tool, keep in mind what you want it to do, how much detail you need, and how that data is presented to you. Try to buy the right tool for the job you need to accomplish.

5. Custom: You may decide to purchase an expert along with the performance tools. You may find that you only need to rent the expert and not support them for the rest of their careers. If you prepare for the expert by performing as many of the self analysis steps as possible, then you can greatly improve their effectiveness, reduce their time to completion (and hence their cost to you), and insure yourself the information you really need to manage your system. You might be surprised how much time it saves to decide things like what performance means to you and what your demands and expectations are BEFORE you call in an expert!

Tradeoffs

What do you do if you find a performance problem on your system? First, you need to decide if it is a global problem or isolated to one or more applications or programs. Then you need to isolate the most critical resource for this problem area. Globally, this will be the resource which is the most heavily utilized. From a process or application point of view, this will be the resource the application spends most of its time waiting on (other than time spent waiting on the user to enter the next transaction).

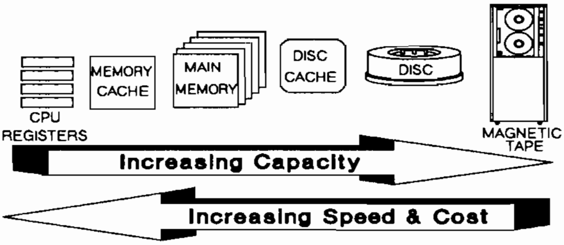
Once you pin the critical resource down, what can you do about it? You have three ways to relieve a bottleneck resource:

1. Get more resource (faster CPU, more MEMORY etc.)
2. Use less resource (re-code application, eliminate waste)
3. Make a tradeoff using a non critical resource.

The first two choices are fairly well understood. You may often make the decision on whether to use method 1 or 2 based on the costs in dollars and time to implement each one. Method 3, the trade off, can often improve performance without a lot of extra money or time. Beware that the trade off is "no free lunch". You are sacrificing one resource which you have plenty of for another resource which you are short of. If you aren't careful you can end up trading bottlenecks and often end up in worse shape than when you started. Here are some of the classical trade offs commonly used in the computer industry.

The most powerful trade off is to trade off storage capacity against speed and cost. If you look at a modern computer system you will see several different data storage techniques in use. The fastest and most costly technique are the working registers within the CPU itself. Main memory is slower then CPU registers but it is also less costly and so can afford to be much larger. We have already discussed how even less expensive disc storage is used to extend main memory size at the expense of slower access. Magnetic tape might be the slowest, least expensive, and the highest volume data storage technique commonly in use on modern computer systems. Some computer systems will interpose intermediate techniques between these storage techniques (memory cache and disc controller caches are examples of intermediate techniques).

COMPUTER STORAGE HIERARCHY



Another common trade off is to trade off main memory capacity in order to speed up access to slower disc devices through a technique known as "buffering". If a typical disc drive takes 25 milliseconds to position itself for a data transfer and then transfers 1 KBytes of data per millisecond, we can play some games by adjusting the size of each transfer. How long would it take for a program to access 8 Megabytes of data on such a disc?

CASE 1: Transfer data in 1 KByte blocks.
# of transfers = 8 megabytes / 1 KByte blocks = 8000
time for a transfer = 25 msec + (1 KByte/(1 KByte/sec))= 26
Total time = 8000 X 26 msec or 208,000 msec

CASE 2: Transfer data in 32 KByte blocks.
# of transfers = 8 megabytes / 32 KByte blocks = 250
time for a transfer = 25 msec + (32 KByte/(1 KByte/sec))= 57
Total time = 250 X 57 msec or 14,250 msec

Result, CASE 2 cuts the total time to transfer the data by a factor of 14 times, BUT it increased the demand on main memory by a factor of 32 times. If main memory is abundant then this can be a very good trade off. If main memory is a bottleneck, then this can be a catastrophe.

Another effect of buffering is that it can allow overlapped resource usage. A program can be utilizing the CPU to process data in the buffer while the disc drive is busy transferring data to another buffer in anticipation of it's being needed soon. Consider the following situation. The program mentioned above, that needs to process 8 Megabytes of disc data, also requires a total of ten seconds of CPU resource during this processing. If access to the Disc and CPU are not overlapped then the elapsed time for the program will be the time it takes to get data from the discs plus the time it takes using the CPU. (Using the 32 KByte buffer example above, this means the program would be finished in 14.250 + 10 seconds or 24.25 seconds.)

If, on the other hand, we transfer the first block of data (57 msec) then let the program start processing it while we simultaneously transfer the second block of data, we can overlap the CPU and disc transfer times so that the total elapsed time is roughly 14.3 seconds. In this case the total run time for a program would be the larger of its total disc transfer time and total CPU time instead of their sum.

Overlapping the disc and CPU portions of a program requires that the operating system can properly anticipate what disc data the program will require before it asks for it. If this works well, then the results can appear to be magic. If the operating system guesses wrong, it can actually make things worse than the non-overlapped case.

Just for reference: MPE/V is VERY GOOD at automatic file buffering and anticipatory buffer loading. The addition of disc caching in MPE/VE made it even better. If MPE/VE is very good at buffering, then MPE/XL, with its Mapped Files is simply phenomenal! UN*X traditionally hasn't been too concerned with overlapping disc and CPU due to its early development as a technical operating system. Later releases of HP-UX should improve this situation considerably as they will bring mapped files into the UN*X operating system.

The other big trade off area is one of **PRIORITIES**. For systems where more than one application is being run, it may often be desirable to designate one application, or group of applications to be of a higher priority than others. In this case if more than one process wants access to a resource, the one with the highest priority will get it in preference to the lower priority process.

A common way to use priorities is to balance the use of CPU when interactive and batch work is occurring on the same computer system at the same time. Traditional interactive applications do not require much CPU time as they spend most of their time waiting for user input. However, they are quite sensitive to the amount of elapsed time it takes to complete a single unit of work (the RESPONSE time). Probably even more important than the absolute length of the response time is the evenness of the responses. Uneven response can cause human operators to have difficulties relating to the computer. Batch work on the other hand, typically uses large amounts of CPU since it does not have to wait for user input, but it is not so sensitive to response time and could care less about the evenness of that response.

By giving the interactive users priority to the CPU over the batch jobs they can be assured of getting their small piece of CPU quickly enough for good even response times while not overly impacting the throughput of the batch jobs.

This is the IDEA. Let me illustrate a case where things didn't quite work out as planned. This was a combination production and program development shop. The programmers had all been cautioned to do their compiling in batch jobs so as to not impact the on-line production work. The problem came when the on-line production work got so heavy as to consume 100% of the available CPU. The programmers noticed that their compile jobs which normally finished in 10 minutes, were running for as long as 8 hours without completing! The system manager of this site called their vendor and asked how they might speed up the batch work on their system. The answer was that they could increase the priority of the batch jobs, or decrease the priority of some of the on-line work to allow the batch jobs access to the CPU.

This plan was implemented. Sure enough, with the batch jobs getting a slightly higher priority they were now completing faster. The next call to the vendor was to complain about (you guessed it) poor on-line response times! Since priority is a trade off, you can't forget that you can only make someone faster by making someone else slower. This is another example of that old proverb "there is no such thing as a free lunch".

It's important to remember for all trade offs, that they can make improvements in one area only by using a resource in another area. You don't actually gain more resource, only trade one against another.

What Do You Do If There is No Bottleneck?

What do you do if you get through all your performance analysis, eliminate all resources as potential bottlenecks and still don't have good performance on your system? Now it's time to start looking at the applications in more detail.

Do you know that there is some fundamental limit of performance for a given application? Have you ever considered if it is possible for your applications to give you the performance you desire? Consider the following statement:

The performance of a given application can never be better than the performance of that application when it is not contending for resources with other applications.

In other words, if it takes 10 seconds to do a transaction when the application in the only thing running on the system, then it would be unreasonable to expect it to take less than 10 seconds when the system is also running other applications. If fact, it would be reasonable to expect it to take longer than 10 seconds in this case. This may sound a bit simplistic, but have you taken the time to determine what the absolute best case performance is for your applications? Let me illustrate an extreme, but very real case.

An application performed on-line transactions which normally took ten seconds to complete. Whether running stand alone, or under normal system load, the response time for this application was ten seconds. The number of users for the application was small (only five users) and the time between transactions was very long (5-10 minutes). Occasionally the response time would be a bit longer, but there were no real problems. Now, an additional 20 users were added and the slow responses got more common. Analysis of this system showed no bottlenecks. The CPU was only 60% busy and no disc IO was occurring at all. The application didn't use any software locks and everyone was at a loss to explain the poor response times. It was not uncommon to see transactions with response times of several minutes!

Here is the type of analysis which was used to explain this phenomenon. First, the application was run and measured in a stand alone environment. Sixty transactions were performed and the amount of CPU and DISC consumed was measured. The number of disc transfers was low, but the CPU used was 600 seconds. Some quick algebra showed that each of the 60 transactions had a CPU service time of 10 seconds. Since this was also the response time, the transaction response time was entirely determined by the access to the CPU.

In production each user submits one transaction every 5 minutes. This transaction consumes 10 seconds out of every 300 seconds so the AVERAGE CPU utilization is only 3%. Even with 20 users the total CPU would be only 20 X 3% or 60% so this matches our measurement of live production. Still no bottleneck, right? From a GLOBAL point of view, that is right. From the application point of view, however, it is dead wrong. Think about what happens when a user enters a transaction. If no other transaction is in progress then the CPU is available so the transaction completes in the normal 10 seconds. If some other transaction is currently using the CPU then the application must wait for them to finish before it can use the CPU (remember queueing?). What are the chances that the CPU will be free when a transaction needs it? (This is queueing theory at its best folks)! Skipping the heavy mathematics, you should be able to agree that the chances of finding another transaction using the CPU are much greater the more users there are who are running the application. With the CPU being busy 60% of the time, the chances are pretty darn good that a transaction will have to wait for one or more other transactions to finish before it gets its chance at the CPU. This means the normal 10 second response time is lengthened to 20, 30 or even more seconds by the queueing during times when transactions "bunch" together.

So, what's the answer to this problem? There are several possibilities.

1. Restrict the number of users so the potential for simultaneous queueing is minimized. This means running with excess CPU most of the time.

2. Increase the CPU speed to handle the desired number of users. This also means running with an excess of CPU to avoid queueing.

3. Examine the application to determine if the CPU requirements can be reduced. Consider trade offs of CPU for other resources such as DISC or MEMORY which are currently underutilized.

4. Split the application into two pieces. An on-line piece which gives good response time to the interactive users and a batch piece which performs the CPU intensive work in the background, at a lower priority. Since no user is waiting for the batch work any more, it is no longer sensitive to response time, merely throughput. That means the system CPU can be run to its maximum with queueing among the batch jobs not causing problems. This assumes, of course, that the nature of the work is such that delayed processing is acceptable.

There are many more examples which could illustrate how a common sense approach to handling a performance problem can work well. Instead of taking your valuable time to relate them here, I will let you begin to develop your own success stories on your systems and you can tell them to me the next time we meet. (Keep a journal, you can sell it later and retire in luxury).

Summary

It isn't as difficult to analyze a computer system's performance as you might have thought. Many situations will yield to a little common sense and a logical approach to the problem. The biggest thing you can do to get your computer system performance under control is to take the time to determine:

What you need a computer system to do?
What types and volumes of work must it do?
What constitutes acceptable performance for each application?

How will you measure the demands and performance of your system?
What "gauges" will you use for these measurements?

Then begin keeping a log of the demands and your system's response as those demands change over time. Record any changes to system configuration or applications which might affect performance.

Start using a strategy for performance management based on measurements rather than feelings. Include a feed back loop where you predict the effect you expect from a change, then measure to see if that effect is seen. Adjust your understanding of how the system is behaving then repeat the process. Avoid making a change "just to see what happens". Make changes which you have reason to expect will improve your performance then measure to see if you were right.

Examine the key applications on your system to determine their specific demands for resources. Determine if your expectations for performance can realistically be met by the current configuration or if some changes should be made.

Consider trade offs in performance. Determine not only what should improve in performance, but also what should degrade.

Structure your approach to performance to learn. Ask for help when you need it, but ask for that help in such a way that you get what you need, and you understand what you get.
ollowing this advice may not make you a performance wizard, but you may appear to be one compared to how you used to approach performance issues. (You can keep this advice to yourselves if you want, but you will be furthering the myth of performance mystique. If you spread it around, then maybe no one will have to settle for poor performance because they don't know any better. Think about it).

Live Long and Prosper,

A Wizard who has seen the light.

Paper #2050

Facing the Fears of Moving from Host Based to Client/Server Computing

Ray Wilson
Washington Convention Center
900 9th Street, NW
Washington, DC
202-371-3289

Tony Jones
Hewlett-Packard Company
2 Choke Cherry Rd
Rockville, MD
301-921-6203

Many organizations find that their computing needs surpass their original computing architecture. The Washington Convention Center (WCC), such an organization, investigated ways to provide more effective computing resources. The WCC computing environment was changed from a host/terminal based architecture to client/server based architecture. WCC and HP worked together to provide a client/server based model which includes 50+ NewWave PCs, two HP 9000s, an HP Vectra running SCO Unix, all tied together over an Ethertwist LAN. In addition, an HP 3000 is available with RS/232C direct connections.

This paper will describe various issues related to the migration and recommend techniques which we found useful during the project. These issues include:

- Changing from terminal based perspective to client/server perspective.

- Changing from HP Word/3000 to Ami Professional on PCs

- Supporting 50+ PCs on a LAN vs. minicomputers/terminals

- Training end users

- Supporting multiple operating systems: MPE, HP-UX, SCO Unix, and MS-DOS.

Background

The Washington Convention Center (WCC) rents exhibit space and meeting room space. One of the main purposes of the WCC, as part of the Washington Convention and Visitors Association (WCVA), is to bring external customers into the city for generation of revenue. Our clients include:

- Large and National Associations including Political functions
- Trade shows
- Public events requiring formal admission
- Municipal groups

Our responsibilities are multifunctional. To attract the best conventions and shows (and thus more revenue), we provide services which aim to fulfill the needs of our clients. These services include:

- Administrative
  - Fax
  - Copying
- Utility Service Delivery
  - Electrical
  - Telephone
- Engineering
  - Compressed Air
  - Water
  - Drainage
- Catering, Food service
- Internal needs
  - Telephone for staff
  - Engineering for staff

The WCC layout, including over 280,000 sq ft of exhibit space, spans 4 square blocks and encompasses 2 floors. Because of its size, providing these services required a computer system capable of managing multiple activities.

History of computer environment

In 1985, we started off with an HP3000/42 with five (5) accounting application users and a plan to add a Sales and Event Management application. Convention Center Sales and Event Management systems assist the staff with the sale and coordination of leased space commitments and the planning and management of event service activities.

All terminals were connected to a single host. We started out with 5 users with the thought of adding another 10 users for sales and event management. For our needs, the HP3000/42 was considered a 15-25 user computer. Human resources was added. The HP3000 was additionally used for Telephone Call Accounting. The Washington Convention Center receives long distance and local call revenue for exhibitor calls processed using its digital PBX equipment.

In 1987, word processing (HP Word/3000) was added. The users were satisfied with Word. It was well received since it was better than using a typewriter. Daisy wheel printers were preferred since laser printer output appeared to be computer generated. Daisy wheel printers, at the time, had more available fonts.

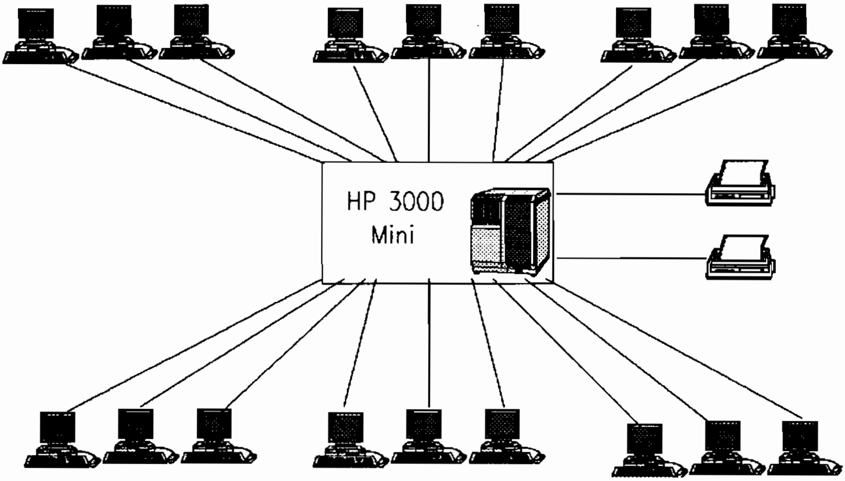


Figure 1. Initial Host Based Configuration

At this point, the typical system user load consisted of:

- 5 concurrent Accounting System users
- 1 Human Resources Management System user
- 1 Batch job: Hourly call detail record processing
- 1 Call Accounting System user
- 18 Word users

Management saw the administrative productivity gains. They wanted to experience the same gains in their job responsibilities.

End user oriented information management was added: (HP ListKeeper/3000). Administrative services used it to keep track of supplies. The Receiving department still uses ListKeeper to track due dates on purchased items.

Access to graphics with HP Draw/3000 allowed for charts, forms design, and provided a rudimentary CASE tool (data flow, and functional process flow).

Individual users initially had access to HP2628 terminals. We later added HP150 PCs with Lotus 1-2-3 and Executive Memomaker. The Budget and

Contracts staff proved to be the most productive users. The budget management process was handled by 2 budget analysts and a manager. The Convention Center's experience using and administering both Word and Executive Memomaker concurrently later influenced the decision to select a fully "what you see if what you get" (WYSIWYG) and scalable word processor that would allow users with differing needs to use the same tool.

Communication

The type of interface to the HP3000 was direct connect RS/232. 45 users were wired through AT&T System 85 PBX digital ports.

Due to the size of the site and convenience, the staff primarily communicated via telephone and hand held radios. Electronic mail (HP DeskManager) was added to facilitate communication but, the terminal interface screens never became popular. Electronic mail was not well accepted due to the sheer convenience of the aforementioned modes of communication.

E-Mail is useful where no other convenient method exists. It provides hard copies to individuals and can be used to reach a larger group of people all at once. A drawback is that users need a terminal to communicate.

We learned that unless tools fit a person's work habits and job function, they would need to be changed (much easier than changing people, work habits or job functions).

In 1989, it became obvious that PC based word processing surpassed host based word processing. The most regarded PC word processing applications were WordPerfect and MS-Word. Increased usage of host based word processing led to sluggish performance. Users also wanted to put charts into documents in an efficient manner. A battle existed between word processing and other terminal based applications. Without a change, productivity would soon suffer.

In addition to the poor performance of the single host, the state of the art of data retrieval and analysis tools did not allow timely delivery of information in a usable form.

The first solution considered was to provide personal computers for everyone in the work force. Budgeting for them became a new issue. More powerful Information Management was needed to handle the increasing volume of data tracked by Image/3000 and KSAM/3000. Analysis became more sophisticated, requiring forecasting and charting ability. In hindsight it was the ability to deliver powerful analysis and data management tools that operate with a coherent vision that convinced executive management of the inherent value of client/server computing.

The WCC had to make a decision on the type of applications:

- Conventional wisdom (character based applications)
    - Lotus 1-2-3
    - WordPerfect

-or-

- Windows based solution
    - MS-Windows 2.11
    - MS-Excel
    - WordPerfect 5.0

The decision was made to buy PCs: 1 HP Vectra RS/20, 2 HP Vectra QS/16, 1 HP Vectra ES/12. The PCs were distributed to:

- Information Services (I.S.) staff
- Budget and contracts

Information services grew from 3 to 5 people. All received PCs. The budget department grew by 2 people. This resulted in 5 RS/20s. Additional end users anxiously awaited arrival of their personal computers.

In 1991, after appropriate requirements analysis and load projections, we developed a network design architecture with the assistance of the Apex Group. This architecture is called the Office Systems Automation Network (OSAN). From the users' perspective, the OSAN provides word processing, electronic mail, printing, and desktop data management. The central server is an HP9000/822.

Today: 1992

An HP9000/817 was added to the network. A primary reason for choosing the '817 was the ability to have it change rolls with the '822 or reduce the load of the '822. An HP Vectra running SCO Unix was later added for application development.

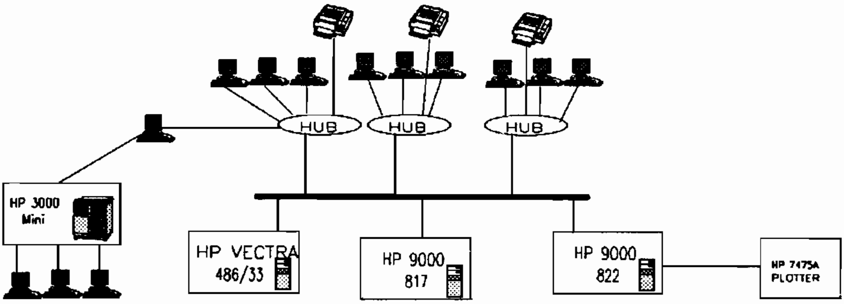


Figure 2. Current Client/Server Configuration

Currently, 74 personal computers are used by the WCC organization:

- 2 in Training lab
- 2 running printers, pending full conversion to the network
- 1 in Development area used to stream backups and monitor disc space on HP3000
- 1 Computer Aided Design (CAD) station with 20" monitor and LaserMaster printer
- 1 for Security staff
- Rest are for end users
- 2 laptops

A LAN based printer allows for printing in a quicker and more efficient manner. In comparison, it took 20 minutes to print a CAD drawing using a standard LaserJet III with 2Meg upgrade, one minute with a LaserMaster printer (with a special video interface). A LAN based LaserJet IIISi accomplished the same task in less than one minute!

At this time, unfortunately, there was no protocol for the LAN based printer to notify the user of lack of paper, or of potential mechanical complications (i.e. paper jam).

Now that we know the background, how did we make the transition?

NewWave was used to enhance the view of the organizational network. NewWave is an important piece of our architecture because of its object orientation and task automation capabilities. The object orientation allowed users to concentrate on organizing information without needing previous knowledge or expertise in the MS-DOS file system. Users quickly started to take advantage of the 32 character title for objects along with creating folders to organize their information in the most appropriate manner. One NewWave application noted to simplify day to day activities is "Printing a document".

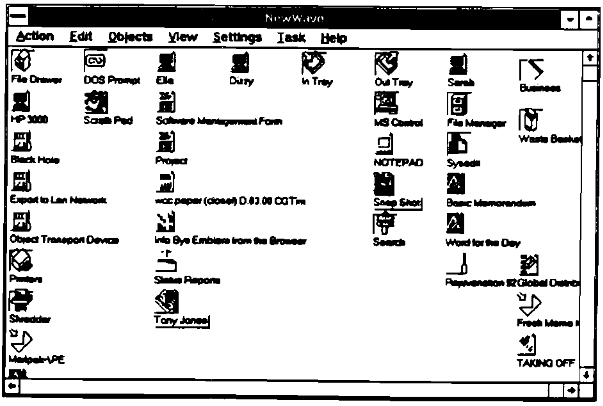


Figure 3. NewWave Based User Enviroment

Before the network was installed, users had to carry the documents to one of several print stations. A print station consisted of a NewWave based PC with a LaserJet printer. To facilitate transferring a document from a user PC to the print station, a NewWave Object Storage folder labeled 'Object Transport' was available on each PC. The folder pointed to a floppy drive. Users simply placed (drag and dropped) the item into the 'Object Transport' folder, resulting in storage of the item on the floppy disc. There were no keystrokes involved in the action, no need to enter file names, nor was it necessary to deal with the MS-DOS file system. The user removed the floppy and carried it to the print station, inserted it in the drive, then double clicked on the 'Object Transport' folder to view its contents. To print an item, the user simply dragged the item from the folder and dropped it on the printer tool.

By supplying a simple procedure to the end users, we reduced training and support costs.

Changing from terminal based perspective to client/server perspective

To address information management needs, we chose HP NewWave Access, a client/server based product which functions as a powerful ad-hoc reporting tool. We use it to retrieve information from the HP3000 databases inherent to the applications. In the future, we plan to use it to access HP9000 Informix databases. It also supports the NewWave agent which can lead to more powerful and current automated reports.

To address communication needs, HP NewWave Mail (client) and HP OpenMail (server) were added to the PCs and the HP9000. Users simply drop the item into their Out Trays, drop in distribution lists, and OpenMail sends the items to their destinations. Roles still exist for voice mail, E-Mail, telephone, and meetings. Ray uses NewWave Mail and a NewWave Text Note when he wants people to reflect on information. Ray also uses a "CC" list so people can see who else is aware of the situation.

To access host based applications, WCC chose Tymlabs Business Session Terminal emulator. Users concurrently access the HP3000 and HP9000s. One terminal object accesses the HP3000 via RS/232 direct connection, and the other terminal objects connect over the LAN. For example, the Information Services staff members typically have 3-4 terminal objects.

A drawback to direct connections is that one must reconfigure the connections with every new change. With LAN based connections, one just adds the new computer to the LAN resulting in immediate access to all end users on the LAN.

Backing up PCs is handled by agent tasks. In addition, to reduce disc space needed on the server during 'back ups', the agent tasks only 'back up' NewWave objects. If necessary, the rest of the information on the hard disc (applications) can be reproduced by re-installing the software.

Software configuration management on multiple PCs became an issue. We developed a series of batch files and check lists for building the PCs and installing the software. Certain configuration files reside on the server. We use the server as a synchronized distribution mechanism. To maintain a consistent set of network related information, the PCs retrieve '/etc/hosts' and '/etc/services' from the HP9000. To synchronize the time across the computers on the network, each PC uses a batch file to get the time from server and sets its internal clock. The batch file uses a 'remsh' command to get the time. In addition, we use the time stamps to assist in **troubleshooting.** Two files are always created upon boot up: 'now.set' and 'last.set'. The file 'last.set' has 2 lines in it: the last time and the current time. This gives us an idea of how long the PC has been operating.

It was important that we chose HP-UX, LM/X and Arpa Services for PCs because the PCs can use HP9000s as file/print and application servers.

Changing from HP Word/3000 to Ami Professional on PCs

What did we learn during the transition?

Ami Professional gave us better control of the documents and a better sense of what final documents would look like before actually printing them. In addition, Ami had the capability to integrate with other applications. Since Ami is a NewWave aware application, users could drop Ami documents on the printer and integrate other NewWave objects into the documents.

In the future, we would provide more customization. We would remove the original Ami masters (templates) which were not relevant to the WCC, customize the style sheets and provide additional masters. We learned how to be more flexible taking advantage of the new knowledge and expertise gained by the end users.

Supporting 50+ PCs on a LAN vs. minicomputers/computers

What did we learn during the transition?

The PCs run the applications from the file server. This allows us to have a single version of the software available for all users. By running network based applications, we can concentrate on improving network access times for all users. This also leads to reduced support costs.

Managing multiple computers is usually more difficult than managing a host with a set of terminals. One has to contend with various software configuration issues, hardware configuration issues, and the potential need to update software resident on multiple PCs. Software updates sometimes require that you visit each of the users in order to update their hard discs. We are investigating the use of remote control packages to assist in managing the PCs.

Network printers are physically located near the users and are logically attached to the servers. All users have access to at least 3 printers. One printer is the main printer in the computer room, and the other two selected printers are the closest printers to the individual user.

Training end users

What did we learn during the transition?

It was important that we make the investment in training end users and the support staff. One Information Services (I.S.) staff member is responsible for developing a syllabus, training material and delivering training sessions. We started with a class room setting and the I.S. staff member lectured. There were labs and experimentation during class time.

Having the users spend time in training led to a short term loss in productivity, but there is a long term payback. Users develop more advanced understanding of the tools and can produce higher quality results in a shorter period of time. This leads to **reduced support costs**. Every user has a base set of knowledge. By teaching every user troubleshooting techniques, solutions are explored before contacting the I.S. staff.

In the future, we plan to take a completely new approach. We will go from a classroom setting to pure hands on lab sessions. Each session has a test, giving users and their managers feedback. Through this method, everyone develops a vested interest in learning and applying the material. More PCs must be dedicated for training and everyone must make the commitment that those PCs are specifically for this function.

Supporting multiple operating systems: MPE, HP-UX, MS-DOS, and SCO Unix

What did we learn during the transition?

Initially, it is more difficult to manage multiple operating systems. However, by taking advantage of each system's strengths, this can lead to increased performance at lower costs. In client/server architectures, specific computers can be optimized for particular functions. The support staff is important! Members need to be highly motivated and willing to develop general familiarity with all of the servers. Likewise, management must be prepared for the support staff to take the time to develop expertise in multi-platform environments.

Support from vendors

What did we learn during the transition?

Vendor support is critical. It can make or break an application. Unfortunately, calling some vendors would lead to: "why would you want to do that?". Vendors must keep an open mind and allow the customer to explore the potential of the product. Some problems can take days to resolve. It is important for the vendor to treat clients as real people.

Policies

What did we learn during the transition?

Security became an issue in the new environment. Unfortunately, users had to stop using the Dexotek Safe folders because forgetting the password would render the information inaccessible (even to the I.S. staff). In addition, it became increasingly more cumbersome to retrieve data secured on a user's local hard disc when the person was not in the office and the information was needed by another user. Finally, there is the remote possibility of the malicious user securing information and leaving without telling anyone the password.

WCC chose not to use the NewWave password since it was considered a hassle to remove it. Instead, the After Dark utility password scheme was used since it has 2 passwords: master password and end-user password. The I.S. staff controls the master password. "If you do use a password, make sure your manager also knows it".

Each PC is a "Personal" computer within limits. WCC supplies PCs as office tools to conduct business related activities. The information stored on the machines is considered the property of the WCC. For personal items, one can use Object storage and put it on floppy.

Users are told not to install applications because it may lead to an awkward support issue. Overall, the original applications were too interdependent and the new applications could affect the other applications.

Application suite (End users)

Here are the applications which are available to the end users:

Word Processing
Ami Professional 2.0 NewWave version
Aldus PageMaker (newsletter and training material). 1 copy

Graphics
Micrografx Designer
Micrografx Charisma
Windows CAD/Drafix
MS-Excel
Lotus Ami Professional

Mail
HP NewWave Mail for OpenMail

Utilities
Dexotek utilities
Berkely AfterDark

Scanning software
Expervision scanning software. Primarily used for OCR (HP ScanJet IIc)

Database
Borland ObjectVision 2.0. In R&D for discrete data capture (lot of potential)
Superbase4
HP NewWave Access (to HP3000/52 over RS/232)

Personal information manager
PackRat

Spreadsheet
MS-Excel

Networking
HP LAN Manager basic clients
HP Arpa Services 2.1

Foundation
MS-DOS 5.0
MS-Windows 3.1
HP NewWave 4.0

CASE
System Architect by Popkin

Project Management
MS-Project

Terminal Emulation
  Tymlabs Session for NewWave

Glue type applications
  MS-Visual Basic
  NewWave and Excel used together

No character based DOS applications

Servers

Objective to have all PC applications running off server. End user PCs only have 40Meg hard discs.

HP9000/822 is PC server
  First Unix computer brought into department
  OpenMail
  LAN Manager/X (LM/X)

HP9000/817
  LM/X
  Informix (upon completion of application)
  Fourgen Financial (Accounting, Informix based)
  Put Drafix (PC software) here instead of on '822

HP3000/52
  Accounting
  Showtime, Call Accounting, FAMIS Accounting
  Profile Human Resources

SCO Unix (HP Vectra)
  Informix
  Fourgen development for HP9000/817 target
  (The developer version costs less for SCO than '817. WCC chose to use '817 as production computer with run-time environment.)

Application Development

  Fourgen now
  SCO computer for development
  '817 for production

End user shared resources configuration

Resource	Usage
P	public ('822)
X	Information services research. Install s/w from back up
W	PackRat database drive (32 user license)
LPT1	LaserJet IIID driven by '822 over LAN.
LPT2	LaserJet IIISi driven by '822 over LAN
LPT3	LaserJet III driven by '822 over LAN.
COMx	Plotter: HP DraftMaster IIRx driven by '822 via serial port. HPGL 2 language, but not getting full benefits from device since have to use standard HPGL driver.

NewWave Agent Tasks

Back up NewWave objects on PC to server
Empty trash
File today's mail in Pack folder and store on server.
Create documents
Start mail transfer

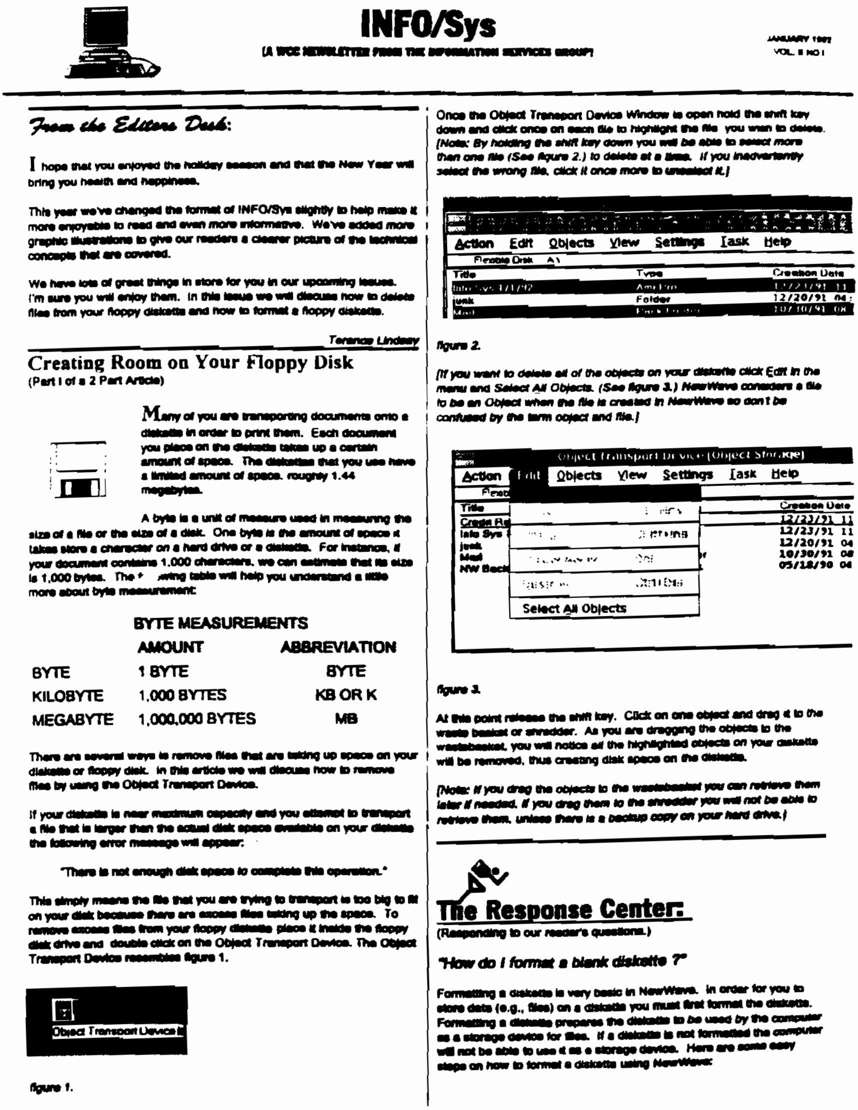
Acknowledgements

The success of this project is largely due to the commitment and support of the entire Washington Convention Center management and staff. We would like to thank the following contributors:

Terence Lindsay	Development of Training Material and User Support Systems
Mark Glover	Illustrations and User Support Systems
John Johnson	Network Design and System Administration
Shu Fang Gan	Applications Development and Integration
Steve Arnett	Systems Architecture Design and Requirements Specifications

Finally, Tony would like to thank his wife, Sheila Butler, for her valuable suggestions regarding this paper.

Sample newsletter



INFO/Sys

(A WCC NEWSLETTER FROM THE INFORMATION SERVICES GROUP)

JANUARY 1992
VOL. II NO I

From the Editors Desk:

I hope that you enjoyed the holiday season and that the New Year will bring you health and happiness.

This year we've changed the format of INFO/Sys slightly to help make it more enjoyable to read and even more informative. We've added more graphic illustrations to give our readers a clearer picture of the technical concepts that are covered.

We have lots of great things in store for you in our upcoming issues. I'm sure you will enjoy them. In this issue we will discuss how to delete files from your floppy diskette and how to format a floppy diskette.

Terance Lindsey

Creating Room on Your Floppy Disk

(Part I of a 2 Part Article)

Many of you are transporting documents onto a diskette in order to print them. Each document you place on the diskette takes up a certain amount of space. The diskettes that you use have a limited amount of space, roughly 1.44 megabytes.

A byte is a unit of measure used in measuring the size of a file or the size of a disk. One byte is the amount of space it takes store a character on a hard drive or a diskette. For instance, if your document contains 1,000 characters, we can estimate that its size is 1,000 bytes. The f   wing table will help you understand a little more about byte measurement:

BYTE MEASUREMENTS

	AMOUNT	ABBREVIATION
BYTE	1 BYTE	BYTE
KILOBYTE	1,000 BYTES	KB OR K
MEGABYTE	1,000,000 BYTES	MB

There are several ways to remove files that are taking up space on your diskette or floppy disk. In this article we will discuss how to remove files by using the Object Transport Device.

If your diskette is near maximum capacity and you attempt to transport a file that is larger than the actual disk space available on your diskette the following error message will appear:

"There is not enough disk space to complete this operation."

This simply means the file that you are trying to transport is too big to fit on your disk because there are excess files taking up the space. To remove excess files from your floppy diskette place it inside the floppy disk drive and double click on the Object Transport Device. The Object Transport Device resembles figure 1.

figure 1.

Once the Object Transport Device Window is open hold the shift key down and click once on each file to highlight the file you wan to delete. *[Note: By holding the shift key down you will be able to select more then one file (See figure 2.) to delete at a time. If you inadvertently select the wrong file, click it once more to unselect it.]*

Title	Type	Creation Date
Info Sys 1/1/92	Ami Pro	12/23/91 11
junk	Folder	12/20/91 04:
Mail	[illegible]	10/30/91 08

figure 2.

[If you want to delete all of the objects on your diskette click Edit in the menu and Select All Objects. (See figure 3.) NewWave considers a file to be an Object when the file is created in NewWave so don't be confused by the term object and file.]

Title	Creation Date
Credit R	12/23/91 11
Info Sys	12/23/91 11
junk	12/20/91 04
Mail	10/30/91 08
NW Back	05/18/90 04

figure 3.

At this point release the shift key. Click on one object and drag it to the waste basket or shredder. As you are dragging the objects to the wastebasket, you will notice all the highlighted objects on your diskette will be removed, thus creating disk space on the diskette.

[Note: If you drag the objects to the wastebasket you can retrieve them later if needed. If you drag them to the shredder you will not be able to retrieve them, unless there is a backup copy on your hard drive.]

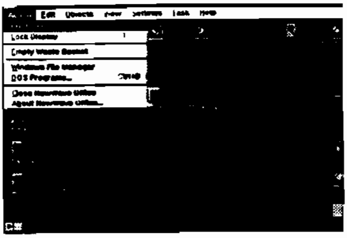
The Response Center:

(Responding to our reader's questions.)

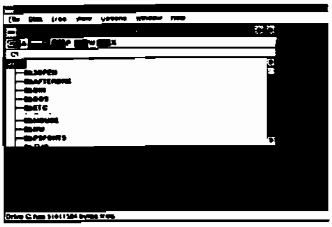
"How do I format a blank diskette ?"

Formatting a diskette is very basic in NewWave. In order for you to store data (e.g., files) on a diskette you must first format the diskette. Formatting a diskette prepares the diskette to be used by the computer as a storage device for files. If a diskette is not formatted the computer will not be able to use it as a storage device. Here are some easy steps on how to format a diskette using NewWave:

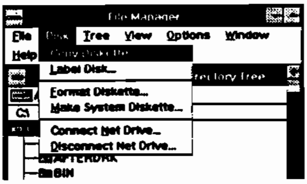
1. Locate the word Action in the NewWave window menu bar. Click once on it with your mouse. The following menu will appear:



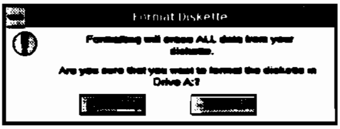
2. Click on the Windows File Manager line in the menu to open it. Once it is opened, you will be able to see a directory tree of your hard disk similar to the one below.



3. Now click on the word Disk in the file manger menu bar. The following submenu appears:



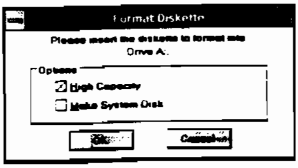
4. At this point choose Format diskette. The following window will appear:



Formatting will erase ALL data from your diskette.

Are you sure that you want to format the diskette in Drive A:?

Click on format. The following window will appear next:



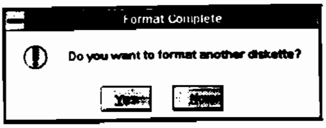
Please insert the diskette to format into Drive A:.

Options

☒ High Capacity

☐ Make System Disk

Because you will be using High Density Diskettes, therefore make sure the High Capacity Box has an "X" in it. Make sure a diskette is inside the floppy disk drive. Now click OK and the formatting process will begin. Once the process is finished the following window below will appear.



Do you want to format another diskette?

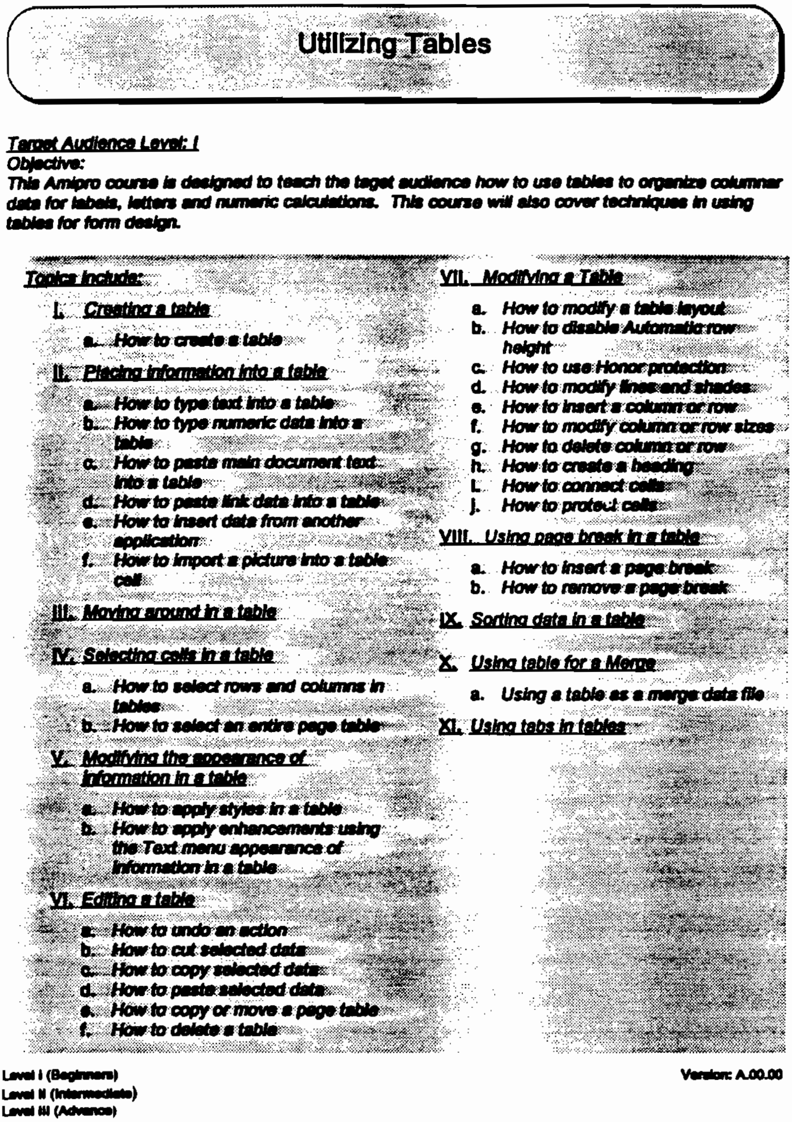
At this point you can format another diskette by responding yes or exit out of the format procedure by responding no and closing the file manger window.

Byte Line

If you are having problems with your PC, don't take matters into your own hands... call ISGI... x3211

INFO/Sys NEWSLETTER is a publication of the Washington Convention Center Information Services Group.

Publisher:	Ray Wilson
Editor:	Terence Lindsey
Contributing Editor:	Michael Greene
Contributing Editor:	Theresa DuBois



Utilizing Tables

Target Audience Level: I

Objective:

This Amipro course is designed to teach the taget audience how to use tables to organize columnar data for labels, letters and numeric calculations. This course will also cover techniques in using tables for form design.

Topics include:

I. *Creating a table*

- a. *How to create a table*

II. *Placing information into a table*

- a. *How to type text into a table*
- b. *How to type numeric data into a table*
- c. *How to paste main document text into a table*
- d. *How to paste link data into a table*
- e. *How to insert data from another application*
- f. *How to import a picture into a table cell*

III. *Moving around in a table*

IV. *Selecting cells in a table*

- a. *How to select rows and columns in tables*
- b. *How to select an entire page table*

V. *Modifying the appearance of information in a table*

- a. *How to apply styles in a table*
- b. *How to apply enhancements using the Text menu appearance of information in a table*

VI. *Editing a table*

- a. *How to undo an action*
- b. *How to cut selected data*
- c. *How to copy selected data*
- d. *How to paste selected data*
- e. *How to copy or move a page table*
- f. *How to delete a table*

VII. *Modifying a Table*

- a. *How to modify a table layout*
- b. *How to disable Automatic row height*
- c. *How to use Honor protection*
- d. *How to modify lines and shades*
- e. *How to insert a column or row*
- f. *How to modify column or row sizes*
- g. *How to delete column or row*
- h. *How to create a heading*
- I. *How to connect cells*
- j. *How to protect cells*

VIII. *Using page break in a table*

- a. *How to insert a page break*
- b. *How to remove a page break*

IX. *Sorting data in a table*

X. *Using table for a Merge*

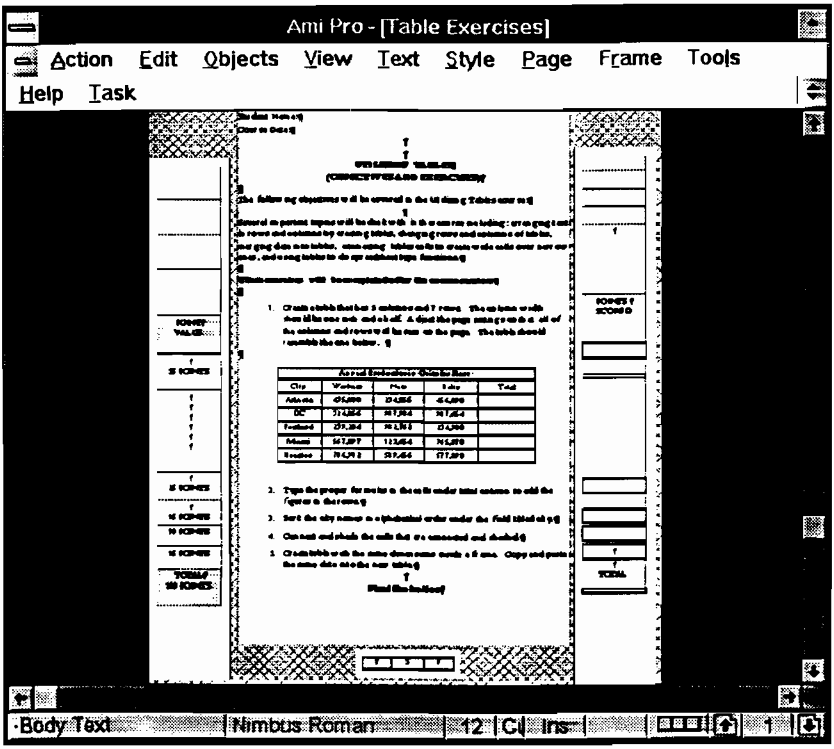
- a. *Using a table as a merge data file*

XI. *Using tabs in tables*

Level I (Beginners)
Level II (Intermediate)
Level III (Advance)

Version: A.00.00

Sample Test


Ami Pro - [Table Exercises]
Action
Edit
Objects
View
Text
Style
Page
Frame
Tools
Help
Task
Body Text
Nimbus Roman
12
Ins
1

Paper 2051
HP- UX and OSF in the Commercial Market

Lisa Burns
Hewlett-Packard Company
19447 Pruneridge Ave., Mailstop 47LA
Cupertino, CA 95014
408-447-6628

HP has been involved in UNIX for ten years. Over those years HP has provided numerous value-added features and has enhanced the quality of its UNIX offering (HP-UX). This investment has paid off for HP's customers: HP-UX is now the most robust UNIX environment for large, mission-critical applications.

The Open Software Foundation (OSF) offers an excellent, broad array of technology. This includes widely accepted technologies for distributed computing environments and other commercial computing applications issues, as well as a standardized operating system, OSF/1. HP, as one of the founders of OSF, views OSF technology as an important part of its strategy.

This paper describes the benefits that OSF technology brings to the commercial market, the benefits HP-UX brings to the commercial market, and HP's strategy for incorporating OSF technology into HP-UX. The paper will then describe what customers can expect from HP-UX in the areas of portability, migration and interoperability as this technology is incorporated. Now and in the future, HP-UX will provide a high degree of compatibility with other OSF vendors while maintaining the quality and value-added technology that is present today in HP-UX.

Benefits of OSF Technology

Though it was initially formed to focus on operating systems, OSF now offers a broad array of technology. The advanced technical features of each of these offerings are impressive by themselves. When combined with the broad acceptance that these technologies have received in the marketplace, the feature set becomes even more appealing, and provides key benefits to the customer.

OSF provides consistency for an entire environment, including user interfaces, distributed computing tools and technologies, and the tools to manage that environment. As more and more customers connect their computers together to run large, distributed applications, such tools become critical.

The following is a partial list of OSF technologies:

Layered Products Technology

Motif -- Motif is a de facto industry-standard user interface that is already being used on a variety of UNIX and non-UNIX operating systems. Motif allows application designers to easily link their code to windows display panels, which include open and close buttons, scroll bars and other familiar features. This consistent user interface limits application training for users and improves their productivity. In addition, Motif minimizes the programming effort for Independent Software Vendors (ISV's).

Logical Volume Manager (LVM) -- LVM is provided as a layer between the file system and devices. It allows concatenation of several physical disks into one larger logical volume.This allows customers to have files larger than the available space on one physical disk. Also, if the file system running above LVM allows it, customers can increase the size of the file system without reconfiguring and rebuilding the operating system. LVM also provides disk mirroring, the ability to duplicate all writes to a disk on a second, or even a third disk. If one of the disks fails, the other remains on-line, therefore no data is lost.

Distributed Computing Environment (DCE) --DCE was recently released by the OSF and is already becoming the distributed computing environment standard for both UNIX and non-UNIX systems. Industry commitment to DCE is strong. For example, HP, DEC and IBM have committed to porting DCE to both their UNIX and proprietary operating systems. In addition, several large customers, including Boeing, Nippon Telephone and Telegraph (NTT), and the European Community (EC) have stated that DCE is their corporate standard.

DCE is a robust environment that allows for secure, reliable distributed computing and distributed file systems. Besides providing security and reliability, DCE supports location (system and network) independence, meaning data or services can reside anywhere in a customer's enterprise and still be easily accessed. Security, reliability, and location independence are critical in the large heterogeneous distributed environments customers have today.

Distributed Management Environment (DME) -- DME is a comprehensive network and system management framework to manage a distributed system. When released, DME will provide tools to configure a distributed environment, to install, update and manage software and software licenses, and to manage distributed printing. Also, DME will provide a set of development tools for distributed applications. DME will be well-integrated with DCE, and will use DCE services.

File System Layout -- OSF/1 uses the industry standard file system layout modeled after AT&T's System V version 4 (V.4) file system layout. The V.4 layout includes both standard locations within directories for executable, configuration files, and user data, as well as a standards for installing the system and starting it up via scripts.

OS-Related Technology

Multiprocessing -- OSF/1 features multiprocessing (MP). MP Allows processes to run concurrently by assigning tasks to each of two or more processors. Large On-Line Transaction Processing (OLTP) environments may have hundreds of processes running concurrently. In these cases, MP capability can greatly increase throughput on a machine.

Threads -- OSF/1 also incorporates kernel threads. Threads are a facility allowing a given process to be divided by the application into a set of sub-tasks that can execute concurrently rather than serially. The application designer identifies these sub-tasks and assigns a thread to each one. Threads can easily share resources such as address space and file descriptors.Threads have lower startup, shutdown and context switch overhead than a regular UNIX process, and are therefore sometimes referred to as "lightweight processes."The lightweight processes can be scheduled independently on a single processor machine, allowing one thread within a process to execute while another waits for I/O or other resources. On an MP machine, several threads within the same process may execute in parallel, each on its own processor. All of these features of threads improve throughput significantly, on both uniprocessor and multiprocessor machines.

There are two types of threads: user space threads and kernel threads. In user-space threads, multiple threads of execution can reside within the same process and share common resources. However, all the threads in that process are viewed by the operating system as one process, and so only one thread can be run at a given time, even if the customer has a multiprocessing machine. Kernel threads provide all the benefits of user space threads in addition to allowing several threads within one process to be scheduled in parallel on a multiprocessor machine. Kernel threads provide more parallelism than user threads but incur slightly more system overhead as well.

Standard Interface --As OSF's Application Environment Specification (AES) is implemented by more and more vendors, it will provide a consistent interface for end users, programmers and systems administrators. In the heterogeneous UNIX environments that are common today, this consistency will greatly reduce training costs and improve productivity significantly.

HP's role in OSF technology

The benefits of OSF technology to the industry and to customers are many. HP is a leader in supporting OSF, and has provided much of the critical functionality upon which OSF's current technology offerings are based.

For Motif, OSF accepted pieces of HP's New Wave technology in its request for technology (RFT) process. HP provided these pieces and did the integration of the final Motif version for OSF. HP continues to be very active with OSF in further enhancement of Motif.

For DCE, HP provided a critical, central piece of functionality, the Remote Procedure Call (RPC) component. RPC is based on HP's NCS product, and allows developers of distributed applications to call procedures on remote machines as easily as if they were executing locally. HP technology also provides the security, user management, and diskless building blocks of DCE.

For DME, OSF has accepted HP OpenView Network Management server software as the framework for DME. Also licensed from HP is the NetLS (Network License Server) for network-wide software licensing. Finally, OSF accepted HP's software distribution utilities which perform installation and update of both system and application software from a single location.

Benefits of HP-UX

HP has a substantial investment in HP-UX, and we want to maintain and grow the commercial functionality that we have developed over the last ten years as we incorporate OSF technology into our operating system. Key value-added features that are available today on HP-UX are listed below.

Scalability -- The scalability of an operating system is a measure of its performance and quality across a range of machine sizes. Many vendors in today's

UNIX market provide offerings which address a certain range of machines very well. However, these vendors do not offer the same operating system for other types and sizes of machines. Typically, a vendor has an excellent workstation offering or an excellent multi-user offering but not both.

HP-UX, in contrast, runs on the widest range of UNIX-based systems of any operating system in today's market. From desktops to mainframes, HP-UX delivers excellent price/performance. HP's product line extends from workstations, to low-end PC servers to mainframe class multiprocessing machines.

High Availability-- For mission-critical OLTP applications, HP-UX offers SPU Switchover on the Series 800 systems. Switchover transfers an application to a second processor in the event of a hardware failure, allowing data entry and processing to continue without interruption.

Powerfail Recovery -- Also for mission-critical applications, HP-UX on the Series 800 protects customer data from short power outages without the need of a UPS.

System Management Tools --HP-UX includes a large suite of tools for common administration tasks. System Administration Manager (SAM) is a menu-driven administration facility that is bundled with HP-UX and performs a variety of administration tasks such as maintaining user accounts and configuring the kernel. Advanced tools also exist for high speed backup (Omniback), detailed performance analysis (PerfView), and printer administration (OpenSpool).

On-Line Diagnostics -- HP-UX provides comprehensive on-line diagnostic tools. SupportWave, provides a graphical user interface and allows users to verify that all the devices, I/O cards, floating point processors are properly installed and enabled. Other diagnostics that ship with all HP-UX releases are available to HP service personnel for use when trouble-shooting hardware failures. Predictive support, available on the HP-UX commercial services, allows HP support personnel to detect component failures before they cause system disruption.

Complete Networking -- An environment is more than an operating system and includes various networking requirements a customer may have. HP offers a complete set of networking products that cover the UNIX networking standards, OSI networking standards, and IBM protocols.

Standard Conformance -- HP's commitment to standards is central to our product strategy. We have over 300 HP professionals working in industry groups to refine existing standards and to develop new ones. We not only define standards, we are typically the first to implement them. HP was the first vendor to pass the

System V verification suite from AT&T and the first full-line vendor to ship OSI networking.

HP's Approach to incorporating OSF Technology

HP wants to bring to the customer the benefits that OSF/1 has to offer, while maintaining the value added functionality and application availability present in HP-UX. To achieve this objective, HP will evolve OSF technology into HP-UX over time. Specifically, HP will do the following to address the key OSF/1 benefits:

Multiprocessing -- HP-UX provides symmetric multiprocessing tuned and optimized for our Precision Risc family of computers. This technology is available today.

Threads -- HP's current plans will add user-space threads to HP-UX in our next major HP-UX release. HP-UX's context switch overhead, even for traditional UNIX processes, is already much lower than other UNIX implementations due to our Precision Risc architecture. The addition of user threads will provide an even faster context switch. Because of this, the need for kernel threads on HP-UX is not as great as on many other vendors' machines. We are currently evaluating the schedule for implementing kernel threads in HP-UX.

Logical Volume Manager (LVM) --Beginning with release 9.0, LVM with enhancements will be available on the Series 800 as a standard bundled offering.

Standard Interface --HP is committed to meeting all the key industry standards including OSF's Application Environment Specification (AES). Over time, this will include a common commands set, as well as the Application Programmer's Interface (API).

File System Layout -- HP's current plans include a move to the V.4 file system layout in our next major HP-UX release.

Portability and Migration

Customers may be concerned about the impact of incorporating OSF technology on their applications and on the administration and use of their existing systems. HP is committed to providing an easy upgrade path for customers as we release these new features. In general, current HP-UX customers should see little impact in the future. Customers moving to HP from other OSF vendors should also have an easy migration. Changes which both types of customers may notice are listed below:

End Users --As HP moves to standardize on a common commands set with other OSF vendors, current HP-UX end users should see little change. HP VUE, the Motif-based workspace manager, will continue to be supported on future HP-UX releases. HP-UX end user commands are already close to the OSF/1 command set, and HP is currently investigating any differences so that migration can be as smooth as possible for customers.

For end users moving to HP's operating system from another OSF offering, there should be little change as well, since the command set will become common. If the end user is moving from another Motif-based user interface, HP VUE will present little change. HP VUE is already available on IBM platforms.

Programmers -- As HP-UX incorporates new technologies, current HP customers should see little change. HP is working to minimize the effect of both OSF and V.4 on customer applications. If the application is coded to industry standards (POSIX, X/Open, OSF AES), little if any work should be required to move to new HP releases. For the V.4 file layout, customers may see an impact if path names are included in programs or in scripts. HP is investigating migration tools to ease this transition.

For customers moving from an OSF operating system, few changes should be required to move applications to HP-UX. Again, if the application is coded to industry standards, customers should be able to port smoothly. We anticipate some minor differences in compilers and build tools between OSF vendors. Each OSF vendor will provide their own compiler that is highly tuned for their architectures. This allows programmers and system vendors to provide the user with the best performance. However, to the extent that a programmer uses functionality not specified by X/OPEN, he or she may notice differences between vendors' implementations of these tools.

System Administrators -- For current HP-UX system administrators, the incorporation of new OSF technology into HP-UX should not be a large change. Support for SAM will continue on HP-UX, so administrators can continue to use this tool rather than writing their own administration scripts.

Since OSF does not currently specify an administration standard, systems administrators will probably see the most differences when moving from one OSF implementation to another. For example, while HP offers SAM, IBM offers an analogous tool called SMIT. The release of DME technology from OSF may allow vendors to converge on administration tools. HP anticipates integrating SAM and other tools into DME once it is available.

Interoperability

The average UNIX customer is not going to have just one type of UNIX machine. Today, HP-UX machines coexist with other vendors' UNIX machines, proprietary operating system machines and MS-DOS and Apple PC's. Fortunately, there are several technologies that address interoperability and are operating system independent:

ARPA Services -- These services sit on top of TCP/IP networking protocols, and include various services to allow logging in to a remote machines as well as file transfer. ARPA services are available on virtually all UNIX systems and are available for many proprietary systems as well as PCs. They are supported today on HP-UX and will continue to be supported in the future.

Network File System. -- NFS is a distributed file system that allows users on different machines to share files. NFS is available across UNIX and non-UNIX machines. PCs can also be a NFS client. NFS is already available on HP-UX today and will be supported into the future. Customers using OSF implementations will see easy file sharing across OSF vendor platforms.

DCE -- DCE is new, but we expect it to become the industry standard distributed computing environment over the next few years for both UNIX and non-UNIX machines. Several OSF vendors, including HP, IBM and DEC, have committed to implementing DCE as soon as possible. HP's current plans call for an intermediate HP-UX release of the DCE Toolkit, followed by a production implementation in our next major release of HP-UX

DME -- DME is still being designed by OSF, but we expect DME to become the industry standard approach for administrating UNIX and non-UNIX machines. HP will investigation production implementation of DME as this technology becomes available from OSF.

Conclusion

For years, UNIX customers have struggled with the challenge of operating distributed multi-vendor computing environments. OSF's DCE, DME, LVM, Motif and other tools provide powerful solutions to the distributed computing problem. HP's product offering has provided the most robust UNIX environment to HP-UX customers for over ten years. We are committed to delivering this OSF technology within HP-UX as soon and as smoothly as possible. Customers will see easy migration paths, improved application portability and user productivity as this occurs. The combination of HP's unique, value-added features and OSF's powerful, new distributed computing technology opens a new horizon of computing power for HP customers.

****

The author wishes to acknowledge Walt O'Maley for his work in preparing the first draft of this paper.

PAPER #2052

Innovative Use Of Client/Server Architecture In System Support

Richard Lasslo
Hewlett-Packard Company
100 Mayfield Ave
Mountain View, California 94043
(415) 691-5480

ABSTRACT

This paper describes how HP's support application, RemoteWatch, utilizes an innovative client/server architecture. The client/server architecture provides efficient and easy methods to distribute an application between different tasks and between different computers. RemoteWatch maintains small daemon (server) programs on each computer that is being supported. The RemoteWatch graphical user interface can access all support data and programs via these daemons. RemoteWatch also displays a graphical map of the configuration of these remote computers. The RemoteWatch support application, using the client/server architecture, provides centralized support of multiple systems on a network.

INTRODUCTION

The HP RemoteWatch support package, developed to support HP-UX computer systems, required a centralized user interface to allow a customer to access all of their supported systems on a local area network from a single point of access. This was achieved by designing a centralized user interface application based on an innovative, distributed server/client architecture. This application nicely integrates together four powerful features: an OSF/Motif X-based graphical user interface, a distributed client/server architecture, the Unix Internet daemon and TCP/IP sockets. This application communicates with the RemoteWatch support programs and data located on each supported machine.

The HP RemoteWatch support package also required the ability to quickly obtain a graphical map of the configuration of the supported workstation cluster and display it via the graphical user interface. This was achieved by the design of a special condensed mapping protocol that can quickly be transmitted over the network.

This paper describes the development and design of this RemoteWatch centralized user interface application.

To help understand the requirements of this application, the first section of the paper describes the functions and features of the RemoteWatch support package and its requirements for the development of the centralized user interface application.

The second section of the paper discusses client/server architecture models including the classical client/server model and the model that the RemoteWatch centralized user interface uses.

The third section of the paper describes the overall design of the user interface application. The application consists of two components, the user interface based *client* program and the remote *server* program. This section describes these two components, discusses the flow of communication that passes between the client and server programs, and discusses how the messages are used.

The fourth section describes the design of the centralized user interface client program in more detail, including the format of the commands sent from the client to the server and a code example describing the method that the client uses to connect with and send commands to the server program.

The fifth section describes how the response messages sent from the server program to the Motif X–based graphical user interface client are received. The differences between X–based programs and non X–based programs are discussed and the solution used is described with a code example.

The sixth section describes the design of the server program in more detail including the Unix Internet daemon program and an example of code that describes the method the server uses to read a command from the client.

The seventh section describes the centralized user interface icon based map and the protocol developed to transmit the map information between the server and client programs.

The eighth section discusses security issues involved when designing a client/server program and discusses some of the security measures implemented.

The final section is the conclusion and summary of the paper.

REMOTE WATCH PACKAGE

The RemoteWatch support package resides on the customer's system and is provided as part of the HP SuccessLine support contract.

The RemoteWatch support package provides proactive support for customers using HP computer systems. The RemoteWatch package consists of two major components: the monitoring software and the centralized graphical user interface.

The monitoring software component is a set of support programs residing on the customer's systems that automatically monitors the systems on a daily basis. These programs check for configuration changes, hardware and software failures, security problems, overgrown log files and for many events or conditions that indicate the system requires attention.

The centralized user interface client/server application provides a single point of access for maintaining an entire network of systems, each running RemoteWatch. The graphical user interface provides the ability to easily view detailed support information on any system on the network. This information includes all support data, disk and LAN statistics, current report files and historical report files on system problems, security reports, and other valuable information.

The graphical user interface provides an icon map representation of the configuration of each supported system. This icon map displays all system hardware including disk drives, tape drives, cluster server nodes, client nodes, printers, and CD ROM drives, The map also displays information on CPUs, operating systems, and connectivity between nodes in a cluster,

The graphical user interface also provides a system access map which contains a user defined map of all the systems that can be accessed from the centralized user interface. A simple click of the mouse on a system icon displays an icon map of the system and its hardware.

REMOTEWATCH REQUIREMENTS

The following were the requirements for the centralized user interface application:

- Access data files on remote machines without need of account and password.
- Access and execute support programs on remote machine.
- Interface remote support servers directly to a graphical user interface.
- Reliable data transfer between graphical user interface and support servers.
- Message protocol specialized for the RemoteWatch support processes.
- Security measures to prevent security breaches by malicious clients.
- Configuration icon map protocol for sending map information.

All of these requirements were satisfied by an innovative combination of a distributed client/server architecture, a Motif/X based graphical user interface, and several other features described in this paper.

BASIC CLIENT/SERVER FUNDAMENTALS

The classical client/server model consists of one or more server programs providing services to client programs (see diagram 1).

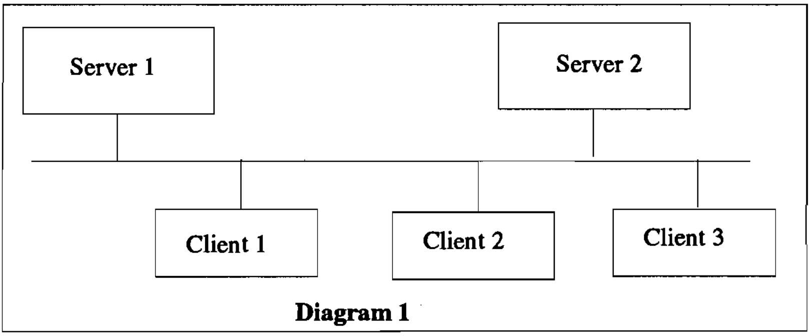


Diagram 1

These client and server programs can be located and executing on different computer systems located anywhere on the network.

A client program typically initiates communications between the client and the server and the client makes requests to the appropriate server for a service.

An example of a typical server program is a printer server. The server program resides on a computer that has one or more printers. When a client program (especially a client program that is located on a computer that does not have a printer) wants to print a document, the client program sends a request to the print server to print the desired document. The print server then receives the print file from the client and sends it to the desired printer.

REMOTE WATCH CLIENT/SERVER MODEL

The client/server model used by the RemoteWatch centralized user interface application is similar to the classical model (see diagram 2). However it has been modified to provide the mechanism to access multiple RemoteWatch supported systems from a central user interface application. In the RemoteWatch client/server model, there is only one client program. The client program is the centralized user interface. There are multiple identical support server programs each one located on one of the supported systems.

Whenever the graphical user interface (the client program) requires support data from a remote computer, the graphical user interface program (client) sends a request to the server program located on the desired supported machine.

A server program is often called a daemon program on Unix computer systems. The term daemon and the term server program can be used interchangeably.

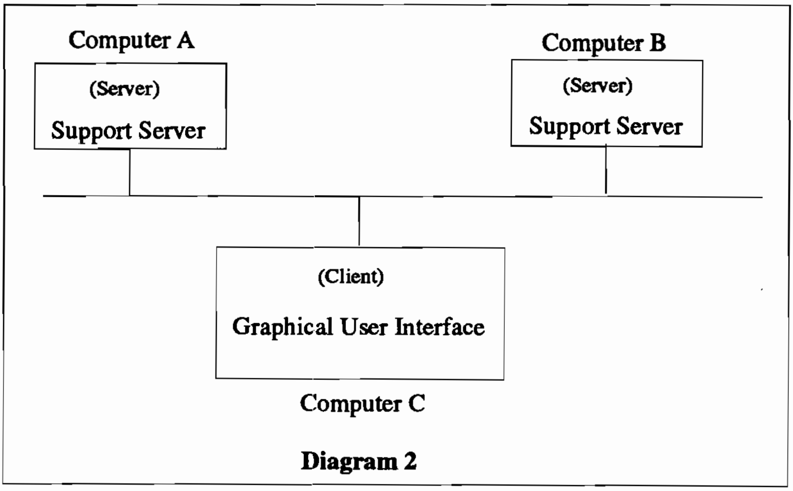


Diagram 2

In the above diagram (diagram 2), the graphical user interface (client program) is shown residing on a different computer than the server (daemon) programs. The graphical user interface can also be located on one of the computers that contains a support daemon program. The client can request services from a server on the same computer.

This architecture provides centralized access by the graphical user interface (client) to any computer systems located on the network that contain the server program (support daemon).

Since the architecture utilizes the network, the computer systems that are being supported and monitored by the centralized user interface can be geographically located anywhere as long as it is connected to the network. This provides a way for a large company with many sites to access and support computer systems from one central location. An example of this is a centralized graphical user interface executing on an HP system in Mountain View, California that can access support server programs on systems in England, France, Italy, Germany, the Netherlands, and Canada.

DESCRIPTION OF OVERALL DESIGN OF APPLICATION

The centralized graphical user interface consists of two separate components: the centralized user interface client program and the remote server program. The centralized user interface program is the client application and provides the single point of access to all support systems. The server program resides on each supported computer system and provides services for the centralized user interface.

The following diagram (Diagram 3) describes the individual components and layers found in each application.

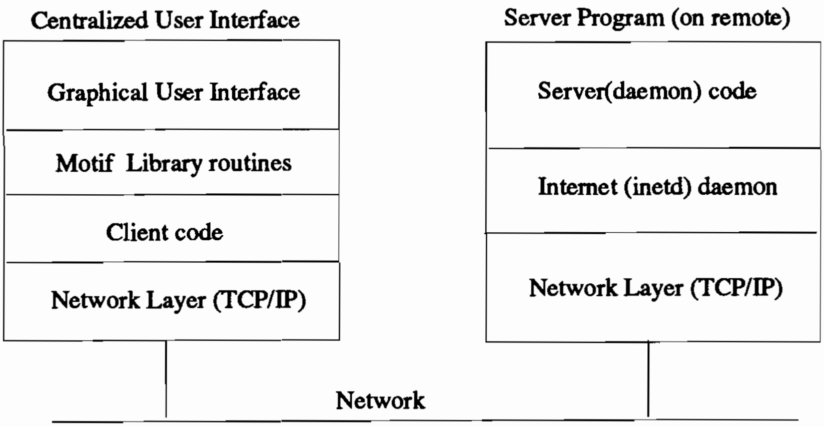


Diagram 3

The centralized user interface contains a set of layered components consisting of the graphical user interface, Motif widget/X–interface library routines, the client code and the TCP/IP network layer.

The server program consists of the server code interfaced to the Unix Internet daemon (inetd), and the TCP/IP network layer.

COMMUNICATIONS BETWEEN CLIENT AND SERVER PROGRAMS

There are two basic categories of transactions between the client and server: single transaction and multiple transactions.

A single transaction consists of a single command request from the client to the server. The client expects a response message from the server for each command request.

There are several types of single transactions as detailed below:

1. Request for directories of support files.
2. Request for contents of a support file.
3. Request to execute a remote support program.

4. Request for other support information (such as configuration map data).

A typical example of use of a single transaction is when the client program requests the contents of a support report file from the server program on a remote system.

Designing a graphical based user interface client application that is capable of sending a command request to a remote computer and receiving a response message from the remote system required some special techniques. These techniques of communications between the user interface and TCP/IP sockets at the network layer are described later in the paper.

MESSAGE PROCESSING

When a response message is received by the user interface, the user interface calls the appropriate routine to handle the response message. There are several ways the user interface can handle these messages. The different ways are described below:

If the message is the result of a command request to obtain the contents of a support report file, the graphical user interface calls a routine that sends the message directly to the output display terminal area of the graphical user interface.

If the message is the result of a command request to obtain a directory of report files that are available on the remote system, the graphical user interface formats the message containing the directory and creates a graphical listbox containing a list of report files that can be selected by the user. When the user clicks on a desired filename, the graphical user interface automatically sends a command request to obtain the report file from the remote computer and displays the report file directly to the output display terminal.

If the message is the result of a command request to obtain the description of the configuration icon map, the graphical user interface interprets the ASCII description response message and builds a graphical tree widget based icon map that represents the workstation cluster configuration.

CENTRALIZED USER INTERFACE

The centralized user interface contains the Motif based graphical user interface code and the client part of the client/server architecture. The graphical user interface consists of a combination of Motif widgets to provide a standard Motif based application. There is also a custom tree widget developed to display the iconic map of the hardware configuration of workstation clusters (See Image 1). The design of the Motif section of the graphical user interface is not discussed in this paper.

Interfacing the Motif/X based graphical user interface to the client/server architecture and the TCP/IP network layer required the design and development of two interface methods. The first method is designed for outgoing command requests from the client (user interface) to the server. The other method is designed for receiving the response messages from the server program. Each of these methods are described in the following sections.

SERVER COMMAND FORMAT

The format of the command requests sent from the graphical user interface client program to the server program was designed to meet the requirements of the RemoteWatch support package. The requests required the ability to pass multiple parameters of variable length. There was also a need to keep the commands short and brief to provide fast performance.

The format of the command request is

|length(3 bytes)|command Byte | First parameter | Second Parameter ... |

The entire command string consists of all printable ASCII characters. No control characters and non-printable characters are transmitted. This was done to prevent the possibility of special non-printable control characters from being intercepted by some of the existing network programs and hardware.

Each component of the command request is separated by a blank character. The first three characters contain the length of the message in ASCII. The server program uses this length field to determine how many characters it should expect when receiving a command request.

The command byte is a single character that represents the command requested. There are presently 32 defined commands. The parameters are variable length ASCII strings containing information that is relative to the command, such as file names or directories.

When a server receives a command request, it is expected to return a response message. The response messages are variable length ASCII strings. There is no length or command byte fields required in the response message. The client program detects when the entire message has arrived by noting when the TCP/IP socket is closed by the server.

An example of sending a command requests is shown later in this paper.

CONNECTING AND SENDING COMMANDS TO THE SERVER

When a client program wants to send a command request to a remote server program it must connect to the remote computer system and send the appropriate command request.

The RemoteWatch client uses TCP/IP sockets to connect to the server program. The client connects to the desired server machine at the port address predefined for the RemoteWatch server. This port address is the same address for all RemoteWatch server programs.

The following is a small code fragment to demonstrate how the client uses TCP/IP to connect to the server program.

```
connectToServer(remoteSystem)
char *serverName;
{
    static  int    tcp_socket;                          /* socket id */
    static struct hostent         *hostInfo             /* remoteHost info structure */
    static struct servent         *serverInfo           /* Server info structure */
    static struct sockaddr_in   socketaddr_in;  /* socket structure */

    /* obtain internet address of remote system */
    hostInfo = gethostbyname (remoteSystem);

    /* obtain the port number for the remote watch server */
    serverInfo = getservbyname("remoteWatchServer" , "tcp");

    /* create an Internet socket */
    tcp_socket = socket(AF_INET,   SOCK_STREAM, 0);

    /* store family, internet address, and port number in socket structure */
    socketaddr_in.sin_family          = AF_INET;
    socketaddr_in.sin_add.s_addr = ((struct in_add *)(hostInfo->h_addr))->s_addr;
    socketaddr_in.sin_port             = serverInfo->s_port;
    /* Connect to server on remote system */
    if (connect( tcp_socket, &socketaddr_in, sizeof(struct sockadd_in)) == -1)
     {
        return -1;  /* connection failed */
     }
     else
        return tcp_socket;
 }
```

The connectToServer routine determines the Internet address of the remote system and the port number of the RemoteWatch daemon, creates a TCP/IP socket and then attempts to connect to the server program located on the remote system. The connect() call is a Berkeley interprocess communication call (ipc) that attempts to connect to the server located on the remote system using the TCP/IP network layer.

If the connection is successful, the client can communicate to the server program using the TCP/IP socket. The client can do standard Unix read(), write(), send(), and recv() calls to the file descriptor located in the tcp_socket variable.

The following is an example of sending a command request to the server;

```
sendRequest(tcp_socket, commandRequest)
int tcp_socket;
char *commandRequest;
{
   send (tcp_socket, commandRequest, strlen(commandRequest),0);
}
```

The variable commandRequest contains the ASCII command string.

RECEIVING RESPONSE MESSAGES FROM SERVER USING MOTIF

Receiving the response message from the server program is more difficult than sending a command request to the server. This is due to the nature of an X–based user interface.

Once the centralized user interface (the client) sends a request, it must wait for the response from the server. This is difficult to do in an X–based program since an X–based program cannot stop and wait for characters to arrive.

A typical computer program is sequentially driven. The order of execution of the code is fixed by the sequence of code statements. A sequentially driven program usually is not concerned with handling real time events. In contrast, an X–based graphical interface program is an event driven program. The order of execution of code depends on the order of events that occur. Examples of events are mouse button clicks, keyboard keys being pressed, windows being uncovered, etc. The X–based program must be able to handle any of these events that occur and in any order.

Another difference between the X–based program and the sequential program, is that the X–based program must always be in an event monitoring loop. This loop monitors for any event and handles the event. If the X–program leaves the monitoring loop for any significant length of time, it will not be able to react to mouse clicks, keyboard keystrokes, etc.

The solution for integrating the X–based user interface to the server program is to make the arrival of characters from the server an event that the X event loop can detect. Whenever characters arrive, the X–event loop is notified and the application can handle the event similar to the way it handles other X events.

This solution is used in the RemoteWatch user interface application. Whenever one or more characters arrive from the server program, the X event loop is notified and the X program will handle the event.

After a command request is sent to a server, the X–event loop is told to monitor for response characters arriving from the server. As characters arrive, the X–event loop calls a special routine that stores these characters into a buffer. When the complete message has arrived, the X–event loop calls the appropriate routine to interpret and handle the message. When the server finishes sending the response message, it closes the socket connection between the server and client. The X–event loop can detect when the socket closes. This is the signal that tells the X–event loop that the entire message has arrived. While the X–event loop is waiting for all the characters of a message to arrive, it is able to handle all other X events.

The following is an example of the code that instructs the X–event loop to monitor for incoming characters.

```
/*  Tell the X event loop to look for characters arriving on the socket socketId      */
/*   Set the X–event loop to call  the readCharacterRoutine, when characters arrive     */

XtAddInput( socketId,  XtInputReadMask,  readCharacterRoutine, NULL);
```

The following is an example of the readCharacterRoutine() code that is called by the X–event loop when characters arrive. This routine reads the characters and stores them in a buffer until all the characters have arrived. When the routine detects an end of message, it then calls a routine to process the entire message.

```
/* Routine that is called when one or more characters arrive from the server */
readCharacterRoutine(client_data, fid, id)
caddr_t      client_data;
int          *fid;
XtInputId    *id;
{
     int   nbytes;
      char buf[5001];
     nbytes = read(*fid, buf,  5000);

     if (nbytes > 0)
     {
          /* Append received characters into a storage buffer */
           appendToBuffer(buf,nbytes);
      }
      else
     {        /* This indicates end of message,  Process the entire message */
              processMessage(buf,nbytes);
      }
}
```

There is no guarantee that the entire message will arrive at the socket at the same time. The readCharacterRoutine is designed to be able to handle when just a few characters arrive at a time in the socket from the server. Each time characters arrive, the readCharacterRoutine() appends the characters to a large buffer until the message is complete.

SERVER PROGRAM

The RemoteWatch server daemon program resides on each supported computer system on the network. The server program is different from the centralized user interface client program because the server program is not X–based. It is a sequential program that can wait on a read() call for characters to arrive without having to handle other asynchronous events.

The server program is also different from the user interface client program because the server program uses the powerful Unix utility, Internet daemon (inetd), to handle its network functions.

The Internet daemon, described below, can be instructed to listen for connection requests coming from a client program. When a client connects to the system, the Internet daemon automatically starts the requested server program and attaches the socket from the client program to the server program.

The server program accesses the socket by executing read() and write() calls to standard input and standard output. Writing to standard output sends characters down the socket to the client program. Reading from standard input, reads characters from the client program. The server program does not have to know anything about the sockets since the Internet daemon automatically connects the socket to standard input and standard output.

THE INTERNET DAEMON UTILITY

The Internet daemon (inetd) is a powerful Unix utility which provides a simple method of creating and installing custom server programs. The inetd program handles many of the Unix client/server programs such as telnet, rlogin, ftp, remsh, etc. The RemoteWatch package takes advantage of this utility for the RemoteWatch support server.

The Internet daemon is called the Internet super–server because it invokes all other internet server processes when needed. The Internet daemon listens on the network for any request to connect to any of the available servers on the system. When a request arrives, the Internet daemon invokes the requested server and automatically attaches the socket from the requesting client to the standard input and output of the server.

When the centralized user interface client wants to send a command request to a remote support server program, it connects to the inetd program on the remote system and requests the RemoteWatch support server (see code example connectToServer() in the section on **CONNECTING AND SENDING COMMANDS TO THE SERVER**). Once the inetd daemon starts the RemoteWatch server, it sends the client program notification of the success of the connection. The centralized user interface then can send the command request directly to the waiting server program.

The beauty of the internet daemon is that it takes care of setting up and interfacing to the client socket. It automatically attaches the output of the socket from the client to standard input of the server and attaches the standard output of the server to the input of the client socket. Thus the server program does not have to contain any socket creation and handling calls. The server program calls *read* and *write* calls to standard input and output to read and write characters to the client program.

SERVER READING COMMAND FROM CLIENT

When the Internet daemon starts the RemoteWatch server program, the centralized user interface will shortly be sending a command request through the Internet socket. The server program waits on a read() call on the standard input for the arrival of this message.

Because there is no guarantee that all the characters of a command request will arrive at the same time, the read must be done in a loop until all the characters have arrived. The first three bytes of the command request format contains the length of the message. Once the length of the message is determined, the read routine loops until it receives the required message length.

Below is an example of the code to read a command request sent from the client:

```
#define   STDIN   0

char    buffer[500];  /* buffer for incoming command request */
char    len[4] ;             /* contains the message length */
int     numCharReceived,  num_in;
int     messageLength;

/* First determine command request packet length  by reading the first three characters */
numCharRecieved = recv(0, len, 3, 0);

/* Since all three characters may not have arrived all together, loop until they have */
while (numCharRecieved < 3)
{   num_in = recv (STDIN, &len [numCharRecieved], 3 – numCharRecieved, 0);
    numCharRecieved = numCharRecieved + num_in;
}

len[3] = '/0';
messageLenght = atoi(len);
```

```
/* Receive the command request from the client */
numCharRecieved = recv (STDIN, buffer, messageLength, 0);

/* Since not all the characters may not arrive at the same time, loop until they do */
while (numCharRecieved < length)
{   num_in = recv (STDIN, &buffer [numCharRecieved], length – numCharRecieved, 0);
    numCharRecieved = numCharRecieved + num_in;
}

/* Entire command request received, call dispatching routine to interpret command */
```

Notice that the server reads characters sent by the client by using the recv() call and reads from STDIN. STDIN is the standard input and is defined to be 0.

ICON BASED CONFIGURATION MAP

One of the major features of the RemoteWatch centralized graphical user interface is the use of an icon based configuration map to visually describe the configuration of the supported workstation cluster (see image 1).

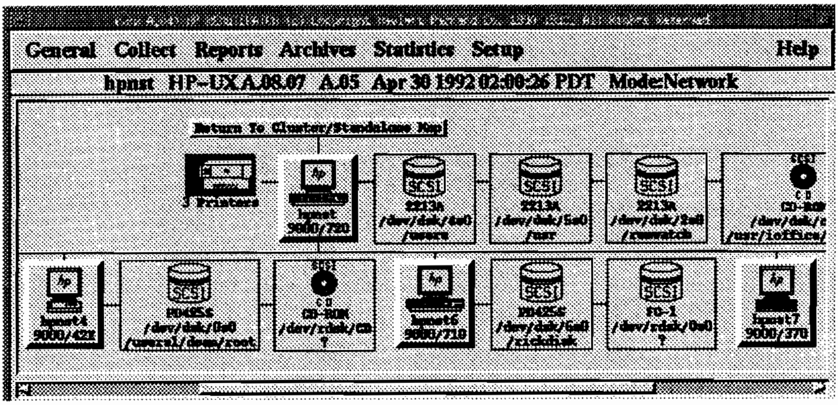


Image 1

When using the RemoteWatch centralized graphical user interface, when a user wants to access a remote system from the centralized user interface, the user first selects a remote system. The user interface automatically displays an iconic map representation of the configuration of the systems hardware.

To create this map, the user interface requests icon map configuration information from the RemoteWatch server on the remote system. To provide quick response time, a concise icon map description language was developed. This allows small complex map descriptions to be quickly sent across the network.

The format of the map description language is as follows:

DATE	date time clusterServerName
SERVER	os name version cpu
CLIENT	os name version cpu
DISK	interfaceType model mountPoint deviceType accessMode Vendor
LAN	internetAddress interfaceName
LP	printer information

When the centralized user interface client program receives the map description in a response message, the centralized user interface interprets the contents and builds a graphical iconic map representation of the workstation cluster.

SECURITY ISSUES

It is very important to consider security when designing a client/server system. A malicious person can develop their own client and attempt to send potentially damaging commands to the server.

Several security protections were designed into the RemoteWatch client/server protocol. The following security features were developed:

1. The RemoteWatch server can only execute a limited set of predefined RemoteWatch commands on the remote system. This prevents the malicious client from executing potentially dangerous commands such as a command to remove files or modify sensitive files such as /etc/passwd.

2. When sending a legitimate RemoteWatch command request, if any of the parameters in the command request string are filenames, the filenames must be relative to a predefined path. No absolute paths to a file are allowed. Also the filename may not contain any .. (double dot) members. If the RemoteWatch server detects any of these conditions, it rejects the command. This precaution prevents the malicious client from attempting to access files other than RemoteWatch files.

3. A user can restrict the RemoteWatch support server on a system from being

accessed by any centralized user interface or client program. The file /etc/inetd.sec can be edited to deny access to specific clients on other systems.

CONCLUSION

This paper has shown that by using an innovative combination of a Motif/X–based user interface, client/server architecture, the Unix Internet daemon, and the TCP/IP network layer, a centralized user interface application can be developed for use in a distributed system support package.

The design of this centralized user interface can be modeled and used for other similar applications that require the unique combination of graphical user interface and distributed client/server architecture.

PAPER #2053

Enterprise Networking: Technology and its Impact on Business Applications

Sam Sudarsanam
Professional Services Division
Hewlett-Packard Company
100 Mayfield Avenue
Mountain View, CA 94043 USA

Mohammad Malik
Worldwide Customer Support Operations
Information Technology
Hewlett-Packard Company
100 Mayfield Avenue
Mountain View, CA 94043 USA

Introduction

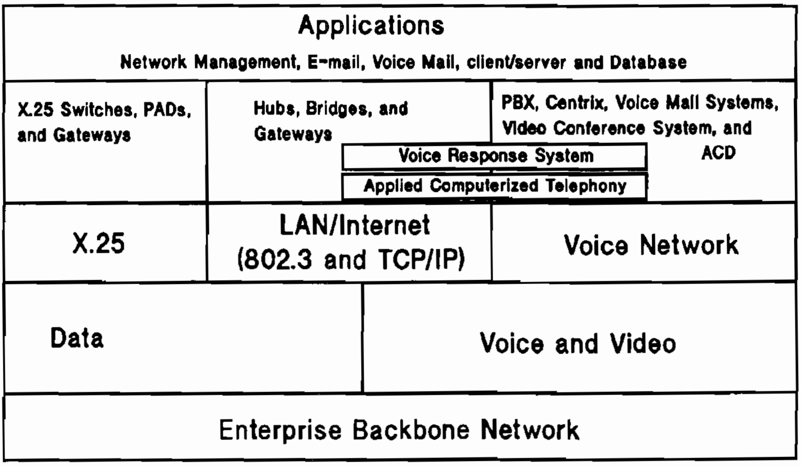
Increasingly, networks are being recognized as valuable resources that must be managed effectively for strategic performance. Network computing is presenting organizations with challenges and opportunities for process change management and competitive positioning. This paper addresses some of the planning and implementation issues involved in network applications as well as emerging communication technologies and services.

Today's workgroup computing in an organization is limited to a single local area network (LAN) in a campus environment. As the workgroup extends beyond a campus to the enterprise-wide network, the need to interconnect LANs over wider geography becomes very critical. Several wide area network (WAN) technologies have emerged to address the LAN and WAN interconnectivity. These technologies range from bridges, routers, and gateways to fiber distributed data interface (FDDI), Frame relay, switched multimegabit data service (SMDS), asynchronous transfer mode (ATM), synchronous optical network (SONET), and broadband integrated services digital network (B-ISDN).

What is Enterprise Networking ?

An enterprise network is generally a private network created by an organization for private communications of that entity. A typical scenario would include workgroup LANs connected to site LANs within a building or a campus. These site LANs are in turn connected to the corporate backbone through circuits leased from public carrier or a hybrid private and public network (Figures 1-3).

Generic Enterprise Network Architecture



Applications

Network Management, E-mail, Voice Mail, client/server and Database

X.25 Switches, PADs, and Gateways

Hubs, Bridges, and Gateways

PBX, Centrix, Voice Mail Systems, Video Conference System, and ACD

Voice Response System

Applied Computerized Telephony

X.25

LAN/Internet (802.3 and TCP/IP)

Voice Network

Data

Voice and Video

Enterprise Backbone Network

Figure 1

Enterprise Multisite Connectivity

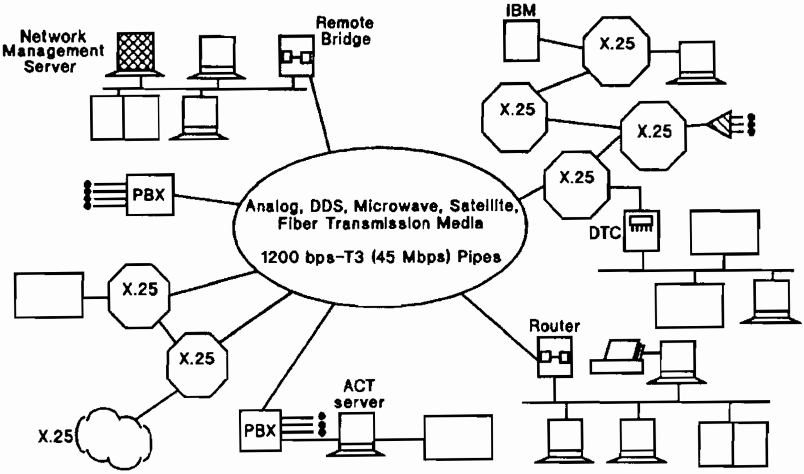

Network Management Server
Remote Bridge
IBM
X.25
X.25
X.25
X.25
PBX
Analog, DDS, Microwave, Satellite, Fiber Transmission Media
1200 bps-T3 (45 Mbps) Pipes
DTC
X.25
X.25
Router
ACT server
X.25
PBX

Figure 2

Multivendor LAN

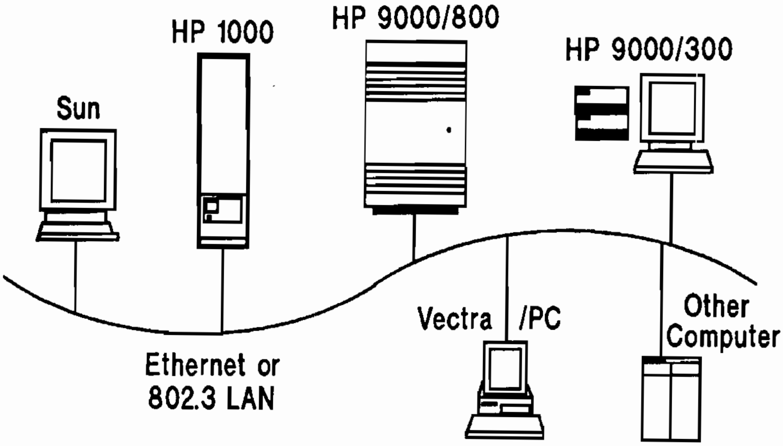


Figure 3

LAN and WAN Interconnection

Regardless of what LAN technologies are used, the network manager faces two pressing needs -- additional nodes and increased bandwidth. The Ethernet technology can be scaled, using repeaters and media access control (MAC) level bridges, to form a very large single LAN; however, this configuration has many disadvantages. For example, the broadcast replies from thousands of nodes will quickly overwhelm the network. Instead of scaling a single network, many sites have configured a topology where workgroups are connected to site backbones that are in turn connected to the corporate backbone through a series of routers. The TCP/IP-based Internet is an example of this type of widely used network.

During the last few years there has been considerable activity in the area of LAN and WAN interconnection technologies such as FDDI, SMDS, and B-ISDN. Because most of these new technologies are still in different stages of standards formation, network managers are faced with the challenge of selecting the appropriate migration path for their network to support the emerging distributed applications. LAN internetworking is a key step towards building the enterprise network platforms (Figure 4).

Table Positioning LAN/MAN/WAN Protocols

<table>
<tr><th>OSI Layer</th><th colspan="10"></th></tr>
<tr><td>Application, Session and Presentation Layers</td><td colspan="10">ARPA/BERKELEY or OSI Upper Layers</td></tr>
<tr><td>Transport Layer</td><td colspan="10">TCP or OSI TP-4</td></tr>
<tr><td>Network Layer</td><td colspan="10">ARPA IP or OSI CLNP</td></tr>
<tr><td>Link Layer</td><td>FDDI MAC</td><td>802.6 DQDB MAC</td><td>802.2
802.3
802.4
MAC</td><td>ATM</td><td colspan="3">Frame Relay (LAP-D)</td><td colspan="2">SMDS SIP</td><td>ATM</td></tr>
<tr><td>Physical Layer</td><td>FDDI PHY</td><td>T3/E3 or SONET</td><td>10base-x Physical</td><td>SONET</td><td>T1/ E1</td><td>T3/ E3</td><td>Narrow band ISDN</td><td>SONET</td><td>ATM/ SONET</td><td>SONET</td></tr>
<tr><td></td><td colspan="4">LAN/MAN PROTOCOLS</td><td colspan="6">WAN PROTOCOLS</td></tr>
</table>

Figure 4

Fiber Distributed Data Interface (FDDI)

FDDI is a second generation LAN standard developed by the American National Standards Institute (ANSI). It is based on a dual, counter rotating, token-passing fiber optic ring with 100 Mbps throughput. Using multimode fiber optics as the transmission medium, FDDI interconnects stations over a distance of up to 200 kilometers. Every active station on the ring must accurately repeat all frames it receives to ensure the integrity of the ring. While primary ring is used for data transmission, the secondary ring acts as a backup in case of link or station failure. Up to 500 nodes can be attached to an FDDI ring. The FDDI standard defines the following four layers: physical media dependent (PMD), physical layer (PHY), media access control (MAC), and station management (SMT).

The FDDI II standard represents the next generation of FDDI network technology that is being driven mainly by multimedia applications. The FDDI II standard defines how real-time or isochronous, synchronized data services like voice and video are multiplexed with the FDDI packet data services. The 100 Mbps bandwidth can be assigned to either packet data or isochronous service in 6.144 Mbps chunks. Individual isochronous channel can be allocated in any size, from 8 Kbps. Looking beyond FDDI II, standardization efforts have recently been started on the FDDI follow-on LAN (FFOL). The FFOL standards are being developed for speeds of 622 Mbps, 1.2 Gbps, and 2.5 Gbps. Instead of token passing access, FFOL will most likely use a buffer insertion mechanism to improve latency performance and interface directly with the emerging cell switched networks.

Distributed Queue Dual Bus (DQDB)

DQDB is a proposed IEEE 802.6 standard to interconnect computers and gateways over a distance of 50 kilometers. It specifies a dual-bus topology supporting unidirectional communications in opposite direction. Stations are connected to both buses and communicate by selecting the proper bus. Since each bus operates at a speed of 45 Mbps or 150 Mbps, DQDB offers an aggregate of 90 Mbps or 300 Mbps throughput. The standards committee is attempting to closely align the IEEE 802.6 physical and link layer protocol structure with that of ATM and SONET to facilitate a smooth migration to broadband ISDN services. DQDB employs a media access method based on distributed queueing arbitration scheme in which all stations have knowledge of frames queued at all other stations. It is a slotted system using fixed size slots of 53 bytes, same as ATM cell size.

X.25 and Frame Relay

The X.25 network services provide one option for network managers planning LAN and WAN interconnection. Large number of applications running on LANs can be supported by current capabilities. X.25-based gateways and routers are well suited for corporate backbone networks; however, X.25 is a fairly old technology and has served its purpose. It is a heavy weight protocol with many features and functions that tend to slow it down from delivering basic datagram service. Many vendors have started to offer frame relay service, which is a simplification of X.25 protocol. Similar to X.25, it is a virtual circuit service that offers users the ability to send packets of data across a network without tying up bandwidth during silent periods. Frame relay performs routing at the link level whereas in X.25- and TCP/IP-based internet, information routing is done by the network layer. Frame relay speed goes up to 1.544 Mbps and possibly higher in the future, but congestion control issues still remain unresolved.

Switched Multimegabit Data Service (SMDS)

The SMDS is a high-speed, connectionless, public packet switched data service defined by Bellcore. Like frame relay, SMDS performs routing at the link level of OSI stack. The SMDS interface protocol (SIP) is aligned with the IEEE 802.6 (DQDB) metropolitan area network (MAN) standard and provides functionality nearly equivalent to MAC service provided by FDDI and the IEEE 802 LANs. Although there are differences between 802.6 and SMDS, the two are aligned on the connectionless MAC service, and properly configured 802.6 devices will operate across the SMDS subscriber network interface. The SMDS will provide speeds of 1.544 Mbps-45 Mbps (T1-T3). In the future, as ATM and SONET services become available, SMDS is expected to use these facilities.

Synchronous Optical Network (SONET)

The SONET is an optical transmission interface initially proposed by Bellcore. It is a physical layer standard that specifies a hierarchy of transmission link speeds, framing, format of the frame, and the optical signal characteristics of single mode fiber required to support these speeds. The lowest level of this hierarchy is referred to as synchronous transport signal (STS-1) or optical carrier level (OC-1). The data rates of SONET range from 51.84 Mbps (STS-1/OC-1) to 2.488 Gbps (STS-48/OC-48) and possibly 9.953 Gbps (OC-192).

Asynchronous Transfer Mode (ATM)

The ATM is an evolving B-ISDN standard that allows dynamic integration and multiplexing of voice, data, and video services on SONET links operating at speeds of 150 Mbps or higher. ATM allows dynamic bandwidth allocation on demand. It uses fixed length cells and is referred to as cell relay. The 53-byte ATM cell consists of 5-byte header and 48-byte information field. ATM network routes at link level by encapsulating 48-byte information payload of fixed size cells and switching these cells through very fast packet switches. It supports data rates of 150 Mbps or 600 Mbps on SONET links.

Broadband Integrated Services Digital Network (B-ISDN)

As mentioned above, B-ISDN requires SONET links at a minimum rate of 150 Mbps utilizing the ATM cell relay technique that can dynamically multiplex voice, data, and video services with 48 bytes of information payload per 53-byte cell. B-ISDN protocol architecture has two ATM related layers -- ATM layer and ATM adaptation layer (AAL). The AAL layer supports the mapping of higher layer information into ATM cells for transport over B-ISDN and then receives information from ATM cells for delivery to higher layers. SONET OC-3 (155.52 Mbps) and OC-12 (622.08 Mbps) have been designated as customer access rates in future B-ISDN networks.

Network Management

Increasingly, powerful computing devices that run sophisticated distributed applications require reliable high-speed networks. These advances in computing and network technologies emphasize access to information across the enterprise utilizing the client/server model.

Computer networks and distributed processing are very useful and powerful tools requiring effective network management. Managing a distributed, heterogeneous environment is a highly complex undertaking. Rapid innovation has served to remove the dividing line between computing and networking. In order to provide an integrated management system, a complete solution must address both network management and systems management areas. Historically, several vendors made attempts to provide proprietary de facto standards such as AT&T's Unified Network Management Architecture (UNMA), IBM's NetView, HP's OpenView, DEC's DECmcc, and Sun's SunNetmanager systems. Unfortunately, these solutions were not quite integrated enough to manage the distributed and multivendor computing environments of networked systems.

As much as a network of diverse systems allows the users to choose the best system that meets their needs, it also makes for a costly administration function. Each hardware platform may require different management schemes. The complex nature of network management and distributed systems necessitates a unified model of network management, scalable from single system management to the enterprise network.

The Open Software Foundation's DME is one such attempt at unifying systems and network management. Various technologies integrated in the DME form a framework that provides a consistent graphical user interface (GUI), the ability to manage system resources and application resources such as software licensing, installation, and printer management.

DME includes three different management protocols -- simple network management protocol (SNMP), common management information protocol (CMIP), and management remote procedure call (management RPC). The SNMP is widely supported for network management today. CMIP may be widely utilized for networked systems management in the future. Management RPC was developed by OSF, based on distributed computing environment (DCE) remote procedure calls (RPC). SNMP and CMIP are included in DME for compatibility with non-OSF standards, and management RPC is intended to support more general functions. Both SNMP and CMIP define the structure of management information (SMI) for managed objects. Management RPC uses a version of CMIP SMI.

SNMP is a simple connectionless protocol that includes the following operations:

* **Get:** request by management system to read value of managed object.

* **Set:** request by management system to modify value of managed object.

* **Get-Next:** request by a management system to read the value of next managed object in the management information base (MIB).

* **Trap:** event notification from an agent to management system.

CMIP is an ISO standard that is not yet in place. It is aimed at network and systems management and is expected to play a major role in the management of distributed environments. CMIP is a connection-oriented protocol, that runs on top of the seven layer OSI stack. The following operations are included in CMIP:

* **M-Get:** request by management system to read the value of managed object.

* **M-Set:** request by management system to modify the value of managed object.

* **M-EventReport:** event notification from agent to management system.

* **M-Action:** request by management system that directs a managed object to perform some action.

* **M-Create:** request by management system to create a new instance of a managed object class.

- **M-Delete:** request by management system to delete an instance of a managed object class.

Management RPC is based on OSF's DCE RPC. Its operations are similar to those of SNMP and CMIP, for example, Get, Set, and so forth.

Case Study

Many organizations today are in the process of technology evaluation and implementation to streamline their business operations and achieve competitive leadership. Existing systems are being analyzed at the department level for productivity enhancement. Traditional processes are slow and human resource intensive. Advancements in network computing technologies represent an opportunity for process improvement.

A major corporation with several manufacturing sites and sales offices used to have business processes that were not automated, resulting in considerable amount of time for transactional processing. For example, the accounts payable process was very slow and took anywhere from 7 to 10 days for completion (Figure 5). The purchasing department mailed the POs to vendors that took approximately 3 to 4 days. The supplier spent 2 days for order processing to ship the merchandise and generate an invoice to be mailed for payment. The receiving department would verify the merchandise and send the receiving documents to purchasing department through inter-office mail. The purchasing department then mailed the payment.

Before Network Computing Technology

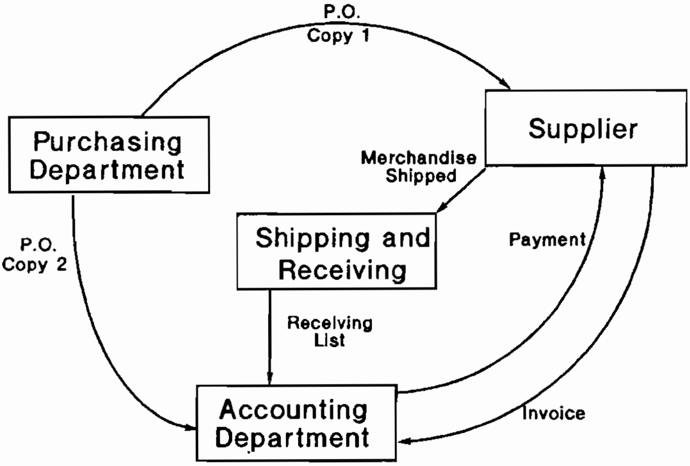


Figure 5

In order to streamline operations, the company management hired a third-party consulting firm to analyze the current processes and recommend improvements. The implementation of a network computing based solutions resulted in process reengineering that not only speeded up the accounts payable process but required fewer human resources and made it more accurate (Figure 6).

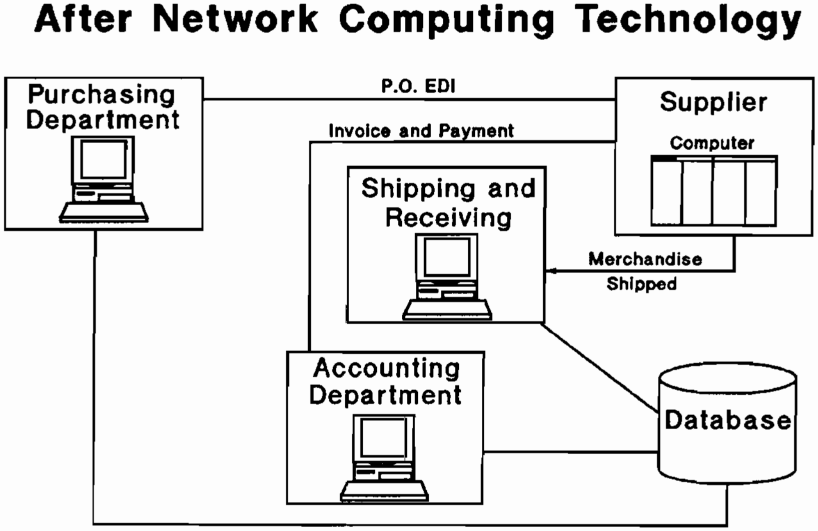


Figure 6

Concluding Remarks

In most organizations, network planning has generally remained in reactive mode. Network managers could afford to wait for bandwidth demand to build up and then install network capacity to support the demand; however, the unpredictable and complex nature of traffic patterns in distributed networked applications environment makes capacity planning function quite challenging. Network planners need to be more proactive in providing increased capacity before user applications demand it.

Several MAN and WAN technologies are emerging to provide enterprise-wide connectivity. The telephone companies plan to participate in this market segment by offering high-speed (1.5-45 Mbps) LAN interconnection, using public frame relay and SMDS services. As these technologies mature and implementation issues are resolved, the economies of scale in a public network may result in favorable pricing policy. In the long run, broadband networks using cell relay techniques will provide dramatic improvement in latency by using faster switching, reduced queuing, and faster circuits. FDDI and DQDB are promising high-speed LAN and MAN technologies. ANSI recently approved the physical layer specification for LANs based on high-performance parallel interface (HIPPI). HIPPI interfaces are widely used in super computers and support data rates of 800 Mbps and 1.6 Gbps. Frame relay, SMDS, ATM, SONET, and B-ISDN networks appear to be emerging wide area networking alternatives to networks based on router interconnection with high-speed links. These alternatives can be realized as private, public, or hybrid configurations. Due to high performance workstations and interfaces such as HIPPI, LAN based backbones will need to provide bandwidths a lot higher than 100 Mbps. Organizations with an embedded base of X.25 networks may stay with X.25 for medium-speed applications and migrate to frame relay as bandwidth and application requirements demand. There is, however costly penalty for upgrading X.25 switches. In the mean time, router-based TCP/IP internets will continue to remain as the most widely used networks.

Continued growth and complexity of computer networks require effective management. Thus far, network management systems have been focused on managing lower layers of the OSI stack. In the distributed networked applications environment, the network management and systems management disciplines will merge, and a complete management solution will need to address all seven layers. Users of the enterprise network computing environment will emphasize these criteria in their product evaluation and selection process. Vendors providing management solutions with these functionalities will lead the market place. Support organizations will benefit from these concepts in the design and implementation of structural change management process.

Acknowledgements

The authors wish to convey special thanks to Mr. Ramesh Gupta, HP Corporate Network Services, for his valuable technical assistance.

References

1. Clapp, G., "LAN Interconnection Across SMDS," IEEE Network Magazine, Sept. 1991.

2. Fiedler, J., "Enterprise Networking: The role of LAN interconnectivity," Telecommunications, Feb. 1992.

3. Hammer, M., "Reengineering Work: Don't Automate, Obliterate," Harvard Business Review, July-August 1990.

4. Kung, H.T., "Gigabit Local Area Networks," IEEE Communications, April 1992.

5. Malamud, C., STACKS Interoperability in Today's Computer Networks, Prentice Hall, 1992.

6. Malik, M., "Future Network Architectures," Thesis Chapter, University of Colorado at Boulder, 1988.

7. Open Software Foundation, Distributed Management Environment, Sept. 1991.

8. Saksena, V., Guest Editorial, IEEE Network Magazine, Sept. 1991.

9. Shah, K., "Managing Networks of the 90s," Data Communications, Dec. 1989.

10. Stallings, W., "Components of OSI: Broadband ISDN," Connexions, April 1992.

11. Usak, D., "The Future of FDDI," Lightwave, Nov. 1991.

From Downsizing to Rightsizing

PAPER #2056

A Guideline for Successful Information Technology Restructuring in the 90's

Feyzi Fatehi

General Systems Division
Hewlett-Packard Company
19055 Pruneridge Ave.
Cupertino, California 95014
(408) 447-4567

Abstract

Most of the work done on the subject of "downsizing" has been done from the *technology* perspective. This paper attempts to take a fresh look at the downsizing phenomenon from a *methodology* perspective. Its intention is to cover the subject not from a *computer* point of view, but instead from a *computing* angle. The paper does not prescribe a technology-specific solution for a successful downsizing effort; instead it presents the reader with a methodology to approach this phenomenon.

Looking at the downsizing trend as an excellent *business re-engineering* opportunity, the paper also emphasizes the inter-relationships and the mutual influence of the information technology on one hand, and the organizational structures, business goals, and management philosophy on the other.

The reader is also presented with a fresh look on the subject of *downsizing justification.* Contrary to the commonly used cost-saving approach, the author recommends a justification approach based on the expected sustainable *competitive advantage.*

Downsizing: A Dominant Trend In The 90's

A major trend of corporate information systems in the 90's will be downsizing - using smaller, more cost-effective, yet powerful processors to support corporate application requirements.

Several forces have created the demand for downsizing in medium and large corporations during the past few years. The most important of these is the availability of the required enabling technologies such as:

- Local area networks (LANs)
- Graphical user interfaces (GUIs)
- RISC-based high-performance, cost-effective servers
- Expandable symmetric multiprocessing (SMP) servers
- Gateways between LANs and wide area networks (WANs)
- Network operating systems (NOSs).

These individual technologies are synergistically brought together by the emergence of several architectural driving forces, most importantly the client/server architecture.

Today, as we move beyond most of our technological concerns, lack of clear methodologies and guidelines becomes the most significant obstacle to downsizing implementations.

In the past few years, the author has had the opportunity to work with several major corporations that have been involved in the process of downsizing their information technology solutions, and has learned many valuable lessons from this experience.

The focus of this paper is to articulate this experience and provide the corporate information officers with a set of guidelines to help enhance the effectiveness and efficiency of their downsizing effort.

Downsizing: A Historical Perspective

In broad terms, downsizing refers to the process of migrating mainframe-based applications to smaller and more cost-effective systems. Downsizing is not a new idea. It has been going on for over two decades.

For the past twenty years, departments have purchased their own minicomputers to run one or more of their applications whenever corporate MIS organizations were not responsive to their functionality/performance needs. The advent of the personal computer in the early 80s and of PC LANs in the late 80s expanded the realm of downsizing to include the desktop.

However, there are two major differences between the downsizing trend in the 90's and the one during the 70s and 80s. The first difference is an organizational one: in the past, the motivation to downsize occurred *outside* the MIS, in response to the

MIS, and while the MIS continued to support the mainframe applications.

Today, the downsizing motivation is coming from *within* the MIS. Facing the next upgrade in the regular three-to-five-year cycle of upgrades, as well as facing a large backlog of applications to be developed, the MIS is motivated to do what users have been doing for twenty years: run the work on less expensive systems. The downsizing process is the same, but the scale is orders of magnitude larger.

The second difference is a technological one: the scope of downsizing today not only covers the *box size*, but also an array of surrounding and complementary new technologies, such as local area networks, graphical user interfaces, and most importantly, the client-server architecture.

The Scope of Downsizing: From "Computers" to "Computing"

Downsizing can be defined as reducing the size of the "computer *processors*" and the cost of "computer *processing*," while enhancing user productivity, and maintaining the required degree of management, security, reliability, and availability that exists on minicomputers and mainframes.

The definition above or other emerging definitions, allude to two distinct aspects of the downsizing phenomenon: the context and the content.

The *context* refers to "computers" in a general sense, incorporating the tangible aspects of downsizing such as switching from a larger machine to a smaller one. The focus is mainly on the financial savings, resulting from the comparatively lower costs of acquisition and ownership of the smaller systems.

The *content* refers to "computing," incorporating such intangible aspects as applicaion architecture, productivity gains, ease-of-use, and the expected higher level of flexibility leading to higher degrees of responsiveness to the changing business needs and environment.

Therefore, context switching alone (that is, a mechanical migration of host-based applications to smaller systems), only satisfies a portion of what is expected of a comprehensive "downsizing" effort. To reap the full benefits of a downsizing process, implementers should also focus on the content aspect of downsizing. More specifically, if we go through a *context switch*, and not a *content overhaul*, we will still be constrained by the same unproductive, counterintuitive, not-user-friendly, mainframe-based applications, only on a cheaper system. While the MIS department will save some money, the users will not gain any productivity, and the flexibility gains will be trivial.

In contrast, doing a context switch along with a content overhaul (that is, moving from a computer-centric, centrally developed application, to a distributed, client/server application) will provide the full advantage generally expected from a downsizing effort.

In the rest of this paper, we will discuss different context and content aspects of the downsizing phenomenon as well as some other closely related topics such as "rightsizing."

Characteristics of Mainframe Application Environments

The key to downsizing large applications from mainframes is to understand the core characteristics of mainframes, and providing an environment supporting these key requirements on smaller and more cost-effective systems. Here is a list of these characteristics:

- A very large number of users
- Multi-user OLTP applications
- Very Large Databases (VLDBs)
- Extremely high I/O rate
- Support of corporate business functions
- Adequate performance for batch and on-line users
- Sufficient sets of system management tools
- RASS (Reliability, Availability, Serviceability, Security).

However, the most distinctive feature of mainframes is their "central" application processing characteristic. Each mainframe contains one set of applications and one set of corresponding data. Users make terminal connections to the mainframe to access the applications that centrally execute within the mainframe. Data is stored on disks that are locally attached to the mainframe.

In terms of application processing, traditional large minicomputer systems fall into the same category as the mainframes because both systems practice centralized, host-based, "glass-house" computing. The only differentiating factor is the number of applications they can accommodate at the same time. Minicomputers can usually accommodate one, or at most a few mainframe-class applications.

In this respect, almost all concepts about "mainframe" downsizing can be effectively applied to minicomputer applications and environments.

General Methodologies to Approach Downsizing

From the users' perspective, there are four general approaches to downsizing computer-centeric, host-based applications: Substitute, Adapt, Re-Engineer, and Off-Load. Each of these approaches is briefly explained below.

1. Substitute

In this approach, the MIS not only replaces its mainframe with smaller, more efficient systems, but also completely substitutes all antiquated mainframe-based applications with applications developed using state-of-the-art technologies and methodologies. Of course, it is easy to fall into the trap of substituting existing applications with applications developed with a mainframe mentality "crammed" into a smaller system. In this case, the benefits of downsizing such as improving flexibility and productivity will not be realized.

2. Adapt

You can use one of the following two methods to *adapt* your existing host-based applications to run on smaller, more efficient systems: manually port the applications to the new environment, or use an automated conversion tool to do the porting job for you. The main attributes of an application encompassed in the adaptation to the new environment may include, but are not limited to, the following areas: programming language, operating system, user interface, and the database. Here, inclusion of a pure "frontware" - the use of a GUI as a pretty face on top the same old business and computing logic - has the potential for giving the misleading impression of a significant step.

As mentioned earlier, this approach, in general, is regarded as a "cram in" approach, and does not fully support the *content* aspect of downsizing. This approach may take time, require high level of in-house expertise, and may prove to be costlier than the Substitute approach. However, this "make" versus "buy" decision can be justified if the available third-party applications do not offer and cannot replicate the required feature set and robustness of the existing host-based application.

3. Re-Engineer

Like the Adapt approach, re-engineering your existing applications requires the same make/buy decision-making process. However, there is one major difference: instead of simply adapting your host-based applications to smaller machines, you are re-engineering your entire application. Not only you can take advantage of the most relevant modern technology such as GUI and RDBMS, but also you have the opportunity to use the latest development methodologies and tools such as client/server model, object oriented approaches, and upper and lower CASE tools to restructure your application.

A significant benefit of Re-Engineering approach is providing the opportunity to use a new technique called Model Driven Development (MDD). Using CASE tools, MDD defines an enterprise model independent of the implementation technologies, and which contains the essential information, design logic, and business rules. Enterprise models of the application systems are then *forward-engineered*, creating technology models based on the specific technology and configuration used. Optimized source code can then be generated from these technology models. Desirable aspects of existing applications, such as business rules, can be extracted and captured into a technology model and *reverse-engineered* to form the basis for portions of an enterprise model. A major advantage of using MDD to re-design the application is the resulting high level of flexibility. When business requirements change, updates to the enterprise model drive updates to the technology models, which, in turn, drive updates to the source code. When alternative technologies become available, a new technology model can be forward-engineered from the existing enterprise model, again yielding updated source code.

The emergence of the Model Driven Development technique antiquates the terms "write" and "rewrite" in the context of application development. Developers only need to focus on *designing* their enterprise and *defining* their information models, and the lower CASE tools will "generate" the source code for them.

4. Off-Load

Off-loading in this context refers to a mainframe *coexistence* strategy. It refers to the approach of keeping some of the existing applications on the mainframe but migrating others to smaller systems. It also could refer to situations in which some of the *functionality* and *data* controlled by an existing application, such as its user interface or sets of data owned by specific departments, are moved to personal workstations or departmental servers.

This approach is intended not only to protect the tremendous investment made in mainframes and the associated mainframe-based legacy applications, but also to reap the benefits of modern technology and its resulting flexibility and user productivity.

Of course, the MIS still has to carry the high mainframe cost-of-ownership in its budget (for example, cost of hardware service, application support, power, space, and expensive staff, to name a few items), but it provides an opportunity for users to gain some productivity and for the MIS staff to gain experience with the new technology.

The benefit of coexistence is to provide new applications access to the existing data controlled by the host applications, and to provide legacy applications access to data controlled by the new applications.

The simplest and crudest method of coexistence is file transfer. It requires uploading and downloading files to and from the mainframe. No programming is required; however, it creates a considerable amount of replicated files. This method is suitable for reporting and query, but not for transactions and updates.

The next most commonly used method is terminal emulation. Using terminal emulations, users log onto the host applications from their PCs or workstations when they need to access data controlled by the host. While no programming is required, this method virtually makes the user a mainframe user with all the associated disadvantages, but in the context of a modern technology environment. The user will suffer from the typical mainframe syndrome of *"central fragmentation."* That is, despite the physical centralization of host-based application systems, each application system stands in a world of its own. Therefore, the potential productivity and efficiency gains from integration of data between several applications can only be gained through the file transfers discussed earlier.

The most elegant and resource-efficient solution to the coexistence strategy is distributed processing. Distributed processing involves real-time access of data on one system from applications executing on another. No resources are duplicated, but considerable programming is required. Using this method, most new host-based application development activities cease. However, some corporate databases remain on the mainframe until the mainframe's high cost-of-ownership and the decrease in the cost of new systems (such as RISC-based, SMP corporate servers) forces the MIS to downsize the mainframe. Until then, the mainframe can be best utilized as the enterprise "meta-server," serving the department and vertical application servers across the corporation.

Downsizing Architectures

The use of the word "architecture" has become a fixture in modern IT literature. Most of this usage reflects the widespread acceptance of the architectural approach to product design and development. It is imperative for users and developers to understand the architectural approach to application development in order to achieve successful adoption of a desired architecture for their own environment.

In the context of information technology, the word "architecture" means the formal structural specification of a computer-based solution. It requires decomposition of the total solution into modules with well defined responsibilities and interfaces, independent of the implementation technology.

Today, as we move away from centralized, host-based computing, several architectures have become dominant for downsized solutions. They include architectures based on Distributed Databases, Peer Processing, Cooperative Computing, and the Client/Server Model. Due to a high degree of looseness in downsizing terminology, there has been a great amount of confusion in the recent computing literature about the definition of each of these architectures.

One element in common among them is the fact that they are based on some form of "distributed computing" model. Distributed computing can be defined as unified systems having their *data* or their *processes*, or both, residing on two or more independent computer systems. In other words, they incorporate two types of distribution: distribution of data, and distribution of processes.

Instead of attempting to add yet another definition for each of the above mentioned architectures, I confine my comments to the prevailing one, the client/server architecture, followed by a client/server application development methodology.

A Note On Client/Server Architecture

Most definitions and implementations of client/server architecture have been based on the idea of distributing the application, not the data, between a server and a client. The client is responsible for front-end processing and the server for back-end processing. Despite all the benefits associated with distributed architectures such as added modularity, flexibility, and scalability, most current client/server implementations have a common shortcoming: they rely on a many(clients)-to-one(server) scheme. This many-to-one relationship between clients and servers is an impediment to coexistence and interoperability of centralized legacy applications with distributed new applications in an enterprise. Let's elaborate on this point.

As mentioned earlier, one of the potential benefits of the downsizing process is to overcome the mainframe syndrome of "central fragmentation" - the inability to integrate data controlled by different applications.

Integration in this context can be defined as the ability to combine and relate diverse sources of information in one physical place. Integration can be done in two different ways. The first, *"integration at the source,"* done by creating one single unified system. The second, *"integration at the destination,"* is done by combining diverse and independent sources of information at the point of delivery. While the first is driven from the center out by using a mainframe, mini, or a LAN based server, the second is driven from the outside in by using a client workstation. [1]

Most recent implementations of client/server architecture have been based on integration at the source which is less flexible in its potential for integrating heterogeneous environments consisting of incompatible legacy and "new generation" applications and databases.

However, this shortcoming is not an inherent feature of the client/server architecture. Client/server model can incorporate a one(client)-to-many(servers) relationship as well as the commonly used many-to-one relationship.

It should be noted that behind the popularity of many-to-one implementations of the client/server model, is the fact that most clients run on a single-tasked operating system (usually DOS, with or without a GUI), while "integration at the destination" requires a multi-tasked client to enable concurrent access to multiple servers.

Recently, there has been some support for a compromise. Supporters of this compromise promote an intermediate *"information/integration server"* concept. The role of this middle server is to access data from a heterogeneous set of sources (that is, legacy host-based applications), and provide *translated, manipulated, and integrated* data to a heterogeneous set of clients. This, of course, adds an additional layer of complexity to the architecture; however, it provides a higher degree of flexibility, while taking a step toward a future implementation of a true "integration at the destination" model.

Having expressed and explained these concerns, let's focus on a methodology for developing client/server applications.

A Methodology for Developing Client/Server Applications [2]

Based on the major differences that exist between client/server and centralized mainframe-based applications, client/server application development requires a different methodology than what has traditionally been used in the host-based environments.

In the *central system model*, the user is at a "terminal" being "controlled" by the application which is executed on the host. The development of this type of system is based on creating a model of what the user does. From that model, a computer application is developed to perform the task. Both structured analysis and design methodologies were intended to be used for development of this type of system.

The developer was a *software* engineer, and the focus was on developing an "efficient" computer-centric solution.

In the *client/server model*, the application is split by function. The functions related to the enterprise or data processing are assigned to the server. The functions related to the user or event processing are assigned to the client. The user - now the "client" - controls the application and is "supported" by the computer system, now the server. The client is at a "workstation" not at a terminal. The developer is an *information* engineer, and the focus is on ease-of-use, resulting in a client-centric solution.

The server represents the collective interest of the clients in an enterprise. Data flow analysis and the development of program control structure is not the paradigm that should be used to develop client/server applications. Server requirements are best understood using a data paradigm. A combination of Entity/Relationship and Attribute/Relationship modeling should be used to develop a data or information model for the server. Enterprise business rules and data-associated events can be defined as extensions to the data model.

The client workstation supports the individual worker. Client requirements are best understood using an input/output model. Inputs and outputs represent events that occur in the client domain. An event/activity or *transaction model* should be used to define the client requirements. Inputs represent the occurrence of an external event. Outputs represent application events. Different users perform different activities within the application domain. They respond to different events and perform different activities for the same events.

The best way to develop a client/server application is through an interactive process using a 4GL or graphical tool. Using the *information model*, a relational database implementation can be put up on the server. Using the event/activity model, the client implementation can be organized into windows representing the client roles and events. The activities are implemented as scripts to be executed when a window object is acted upon. During this process, developers work with people who have subject matter knowledge through several cycles of prototyping and refining the graphical user interface. This differs from the paper-based specification process inherent to building a host-based dumb-terminal application.

In summary, client/server applications are different from central system applications and need a different approach to the requirement analysis and design. The information model works well for server specification. The principal role of the server is data storage and processing the enterprise requirements. In contrast, the client is an event processor and needs a different type of analysis. A transaction or event/activity model, based on user roles should be used for client requirements analysis. For implementation, the information model is transformed into the server database. The activities become the scripts in a window interface.

Having covered some of the context and content-related subjects surrounding the downsizing trend, let's take a brief look at some of the important business related topics.

Downsizing: A Business Re-Engineering Opportunity

In the past, many businesses have restructured their business process and organization to accommodate their computer-based solutions. Departmental procedures, job descriptions, and information flows were adjusted to utilize the computer system most efficiently. From the users' perspective, one of the major expectations of downsizing is to reverse this arrangement.

To fulfill this expectation, the first step toward downsizing an information technology system, should be to go back to the drawing board, and with a fresh look, re-examine your current business, technical, political, and personnel environments, free of the constraints imposed on it by the existing, rigid, computer-centric applications.

The following list is not intended to be a comprehensive list by any means. However, it presents you with a sample set of key questions that should be answered as an outcome of the above mentioned process:

What are the key corporate strategic and tactical goals? What are your critical success factors to achieve these goals? How can information technology best be deployed to facilitate this process?

What is your relationship to competition, and what are they doing? Is the corporation growing? Shrinking? At what rates?

How, specifically, is the information technology used at your corporation? To improve staff productivity? To automate routine tasks? To enhance customer responsiveness?

How crucial is timely access to information to your organization? Does your business depend on single information sources, or do you need to integrate information from several sources and applications?

To what extent do these sources belong to the enterprise, departmental, or individual systems? How about external source/destination connectivity requirements?

In general, gaining a sustainable advantage from your downsized information technology depends on how you plan and architect it to achieve your business goals. A mere context switch to modern technology isn't enough. You may end up automating your "business inefficiencies."

A Note On Downsizing Justification

Recently, the most commonly practiced method used for justifying downsizing has been a simplistic cost-saving analysis. This method simply compares the cost of replacing or upgrading a host system, along with its high cost of ownership, with the relatively low cost of acquisition and ownership on a downsized solution. The result of this comparison, of course, points to a relatively short break-even-time in favor of a downsized solution.

Of course, this is one way of approaching this process; however, I believe it is an oversimplified way of looking at this issue. The major flaw in this approach is the fact that it simply ignores several important and relevant variables, including non-financial and hidden costs associated with downsizing, as well as all the non-cost related benefits and gains resulting from a successful downsizing. Regardless of the above argument, my favorite way of looking at this issue is from a totally different angle.

Gaining A Sustainable Advantage - I prefer to look at the downsizing justification issue mainly from a "competitive advantage" point of view. From this aspect, a corporation is either using downsizing to gain a competitive *advantage*, or simply to reduce its *disadvantage*. If you are a "first-mover" among your competitors, your justification should be to gain a sustainable competitive advantage; otherwise your justification will have to be playing "competitive catch-up".

One might argue that it is virtually impossible for users to gain advantage based on information technology alone, a claim supported by the following two points:

- All components of information technology are universally and freely available to all competitors in an industry.

- There may be differences after a custom application has been developed, but it is usually difficult to protect. Competitors can hire away key employees or use the same vendors and system integrators to take advantage of the leader's experience.

I agree with this argument; however, I would like to point out that there have been classic examples indicating that it is still possible to gain a competitive advantage from a major information technology restructuring effort, given that a "first-mover" strategy is adapted. [3]

Successful experiences have not happened due to a sustainable technology edge, but because they were able to convert a temporary technology edge into sustainable forms of conventional advantage such as scale, customer loyalty, and brand image. Here is a list of recommendations:

1. Develop a vision of how to deploy and architect your downsized solution to deliver improved value to customers or to achieve better operating capabilities.

2. Be the first to use the enhanced operating capabilities to achieve one or more "first-mover advantages," including the opportunity to grow rapidly, to penetrate distribution networks, or to build a unique reputation.

3. Exploit these advantages to achieve other sustainable advantages, such as brand recognition, scale, and customer loyalty, which will survive even after competitors duplicate your achieved capabilities.

Rightsizing

Rightsizing addresses the issues of *balance* and *optimization*. It is a response to your specific and unique needs reflected in a customized, optimized, and balanced information technology solution. In other words, rightsizing is about tuning the technology components and your information processing model to best fit and satisfy *your* unique business requirements.

It is true that downsizing is the major trend of the information technology restructuring in the 90's. However, in some instances, rightsizing could easily translate to "upsizing" - replacing your existing system with a larger, more powerful server to keep up with your expanding business and client-base needs.

In another instance, rightsizing can be interpreted as the process of creating a balance between centralized versus distributed approaches. For example, your MIS could argue in favor of a larger server to serve all the "mission-critical" applications centrally, or to centralize all the sensitive corporate information under a powerful server in the security of a glass-house environment. On the other hand, you can argue that the MIS should rely on ever increasing LAN operating system security features, and therefore decentralize both the applications and the data, requiring a number of smaller networked systems, instead of one centrally large server. A *balanced*, objective, and non-political response to these two arguments will lead to a *rightsized* solution for *your* environment.

Here is a non-exhaustive list of system aspects that can be *balanced* to provide a *rightsized* solution for each specific environment:

- Downsizing vs. Upsizing

- Centralized vs. Distributed

- Mainframe vs. Non-mainframe

- Coexistence vs. Migration

You can, of course, add to or delete from this list based on the requirements and specifications of your particular solution.

A Final Note: The Human Side of Technology

As you can see by now, a successful downsizing effort is the embodiment of many diverse concepts and subjects ranging from computers to computing, from context to content, and from technology to methodology. This paper covered some of the related and essential aspects and issues. However, there are many other associated topics surrounding the downsizing issue which are outside the immediate scope of this work.

Nevertheless, I would like to conclude this paper with emphasizing two, oftentimes ignored, skill sets that have been proven to be absolutely essential for a successful restructuring of corporate information technology systems: managing cultural and organizational change, and cross-functional project management skills.

The first skill is essential because of the wide range of cultural changes that must be implemented if an organization is to achieve maximum benefit from a major IT restructuring effort. An effective cultural change management requires a great deal of thinking, training, planning, and careful execution, to build the required degree of "organizational readiness" and effectively deal with the human side of technology.

The second one, *cross-functional project management skills*, is also often downplayed, if not ignored. That is mostly due to the fact that acquiring cross-functional skills is very challenging, especially because structures and habits work against them. People identify with their profession and usually want to get better at what they do. And most day-to-day work is function-specific.

But excellent functional expertise *alone* does not ensure that you have what it takes to implement highly inter-related information technology systems across different functions of your corporation. What ensures the synergistic integration of all the necessary *functional* expertise for a successful restructuring effort, is a set of strong *cross-functional* project management skills.

"High performing companies boast, 'We've got the best project managers in the world.' Low performers say, 'We've got the best circuit designers.' " [4]

References

[1] For a related discussion on this subject, see Dov Weizman's article titled: Integration by Re-engineering, published in the *1992 CASE Outlook.*

[2] I have not yet found a comprehensive reference on the subject of large scale client/server application development methodology, independent of the idiosyncrasies of specific technologies. However, I would like to reference a few sources that I feel will help readers develop a deeper understanding of this topic:

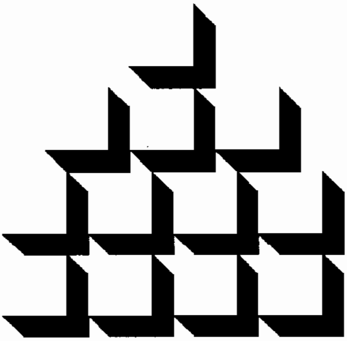
- Joan Greenbaum & Morten Kyng, Design At Work: Cooperative Design of Computer Systems, 1991.
- Stephen J. Andriole, Storyboard Prototyping: A New Approach To User Requirements Analysis, 1989.
- W. H. Inmon, Client/Server Applications In An Architectured Environment, 1991.

[3] For an in-depth review of "first-mover strategy" see John Cecil and Michael Goldstein, The IT Agenda, *The McKinsey Quarterly* - Number, 1990

[4] T. Michael Nevens, et al., What The Best Companies Do, *Harvard Business Review*, May-June 1990.

Paper #2057
Help! My Filesystem Is Full And I Can't Get Up!

By Bill Hassell
Hewlett-Packard Company
2000 South Park Place
Atlanta, Georgia 30339
(404) 850-2181



filesytem is full
filesytem is full
filesytem is full
filesytem is full

Oh, oh! It seems there are only two type of HP-UX system administrators:

Those that have seen this message,

and

those that are going to!

The effect of this message depends primarily on what filesystem is reporting the dreaded error, with the root filesystem (/) having the worst effect. When root fills up, things start failing; processes are killed or core dump, programs that depend on files on the root directories begin to have problems, and the list goes on.

The first thing to do is to gracefully shutdown the system, unless you know where the problem is hiding. There are several common reasons for a filesystem to become full without warning, while other reasons require some stealth to locate.

System Panics

One of the biggest files to suddenly appear in the root filesystem can be found in the directory **`/tmp/syscore`**. Normally, this directory does not exist so core dumps due to a system crash (panic in HP-UX parlance) will not be saved. That's good news and bad news: the good news is that a panic will not suddenly fill your filesystem with a core dump from 16 to 256 megs (or more) of data. The bad news is that there is little chance to determine the reason for the system panic without this file.

Is **/tmp/syscore** the only location for a panic core dump? No. The directory is specified in the system startup file **/etc/rc** and is set as the only parameter given to a program called savecore. When savecore runs, the two files: **hp-core.#** and **hp-ux.#** are created, along with a small file called **bounds**. The # sign is a number starting from **0** and incrementing with every panic that occurs, and **bounds** keeps track of the next number to use.

When the system reboots after a panic, the **/etc/rc** file checks to see if savecore exists and if the directory specified exists...if both are true, **savecore** checks the dump area (typically the primary swap area) for a valid HP-UX memory dump. If the right numbers are found, the **savecore** program announces the date/time that the panic occurred and creates the file **hp-core.0** (if this is the first core dump in the directory). The process continues until all of physical memory (RAM) has been written to the disk, or until (oops) the filesystem is full.

Then, **savecore** writes a copy of the current **/hp-ux** file as **hp-ux.0** to match the dump file. If the filesystem is already full, this file is created as a zero-length file. To be useful, the core dump must have a copy of **/hp-ux** at the time of the dump.

So what is the best technique to prevent a root filesystem from filling up due to a system panic? Simply pick another filesystem to store the dump, one that typically has a lot of space, or always has at least as much space as the size of RAM plus about 2-3 megs (for **hp-ux.#**). How do you locate the size of RAM? You can type the command: **dmesg** and look at the amount of real memory that is available. To pick another filesystem (**/disk7**), change the two lines in **/etc/rc** from:

```
    if [ -x /etc/savecore ] && [ -d /tmp/syscore ]
      then
        /etc/savecore /tmp/syscore
    fi
to
    if [ -x /etc/savecore ] && [ -d /disk7/syscore ]
      then
        /etc/savecore /disk7/syscore
    fi
```

If you need the space back after a panic, simply store the contents of the core dump directory on tape...the simplest command to use is:

```
cd /my_core_dump_directory
tar cvf /dev/my_favorite_tape_devicefile *
```

Don't forget to check the list that tar makes; it should show the two files **hp-ux** and **hp-core** and possibly the **bounds** file. Then, you can remove the files from the core dump directory. Now you can contact HP to have the core dump analyzed for the possible reason(s) for the panic.

Filesystem minfree

Every administrator has probably looked at the **bdf** command after mounting a brand new disk and asked: Where did some of that space go? The answer is that approximately 6% to 8% of a disk's space is occupied by inode tables and superblocks, which contain the pointers to the various blocks of files that are on the disk. In addition, the default **newfs** command will reserve 10% minfree or 10% of what's left before files are stored on the disk, to enhance the filesystem performance.

This buffer allows system administrators to fix problems with the space on a given disk (once the filesystem is marked full) and still have some room (the 10% minfree area) to work. Although the minfree area can be reduced to zero, this is not recommended for the root disk since a file system full message might not allow even the system administrator to log onto the ailing system.

Other disks might be allowed to use 0% minfree, as long as the space is monitored, or the space is essentially fixed. Note also that the HFS method os disk space management relies heavily on 10% minfree to keep the performance in allocating and deallocating filespace at a high level.

Another filesystem tuning is to increase the bytes-per-node value when initializing the filesystem with **newfs**. By changing the number of bytes of disk space managed by an inode, the overhead can be reduced by as much as 50%, at the expense of the total number of files that may be stored with larger inodes. This parameter is tricky since it may prevent easy interchange of data with other Unix systems that cannot handle a wide range of bytes-per-inode values. In general, changing this parameter from 2048 bytes to 64K bytes will only return about 3% or so of the disk space, with a corresponding reduction in the total number of files that can be stored, but this may be ideal for a small collection of large files.

Files that don't belong in /dev

Another very common problem is a file system full message just after doing a backup. How can this be? HP-UX is quite friendly in that it will allow spelling errors, but it will often do something not entirely expected. For instance, if the user were to misspell the name of the tape drive as in:

```
tar cvf /dev/rmt/om /
```

instead of

```
tar cvf /dev/rmt/0m /
```

then instead of printing an error such as: tape not found or tape devicefile does not exist, the HP-UX operating system simply creates an ordinary file with the name given in the **tar** command (or **cpio** or **fbackup**, etc) and all the data to be backed up begins filling the **/dev/rmt/om** file until the entire system has backed itself up onto the root disk (which usually fails with a file system full message).

To find these inadvertent spelling errors in **/dev**, use the following command:

```
find /dev -type f -print
```

This list will contain files that should never appear in **/dev**, that is, ordinary files. Occasionally, a core file or other unexpected files can be discovered in the **/dev** directory.

Managing /tmp and /usr/tmp

/tmp is one of those directories where everyone has access but few seem to treat it with respect. **/tmp** is defined as a temporary storage area and is not to be considered permanent. Processes like email or the **vi** editor use the **/tmp directory** for files, but normal operations will cleanup afterwards and not leave files in the **/tmp** directory. Some HP programs such as update will leave their logfile in **/tmp**, but this is considered correct practice in that the logfile should be reviewed for errors, and then removed or archived.

One way to enforce cleanup of the /tmp directory is to use a command such as:

```
find /tmp -atime +14 -exec rm {} \;
```

which will remove any files in /tmp (or files in directories below /tmp) that have not been accessed in more than 14 days. The other temporary storage area, /usr/tmp, is less often abused by users since they overlook it's existence. Again, processes will create temporary files in /usr/tmp and should (if they terminate correctly) remove their files. This command will clear up /usr/tmp of files not accessed in more than 7 days:

```
find /usr/tmp -atime +7 -exec rm {} \;
```

Another common practice is to cleanup /usr/tmp after every full backup.

Where to remove filesets

Starting with HP-UX version 8.0 and higher, the ability to remove unneeded filesets or applications has been provided through the program rmfn. This is based on the update program and indeed, looks almost identical. When rmfn is run without any parameters, rmfn displays a screen of all the filesets on the system (information on filesets is stored in /system with indexes, customize scripts and related information). rmfn also reports the size of the fileset.

Why can't I just remove the program? Well, in the old days, a simple program was just one thing and therefore easy to remove. But that was then and this is now; today, programs are stored in various pieces all over the filesystem. Things like rc files for local configs, X/Window resources in an app-defaults file, man pages for documentation, commands needed only for the administrator and other commands for general use, all part of a single application program.

To track all these items, the update & rmfn programs make use of indexes kept in the /system directory. Additionally, dependencies between files in different filesets can be tracked, which prevents misloading portions of files groupings that would not be fully functional as a whole.

Filesets that might be removed?

man pages

The documentation pages (**man** pages) can save from 6 to 20 megabytes. The **man** command will no longer find any online help files but this can save a lot of disk space.

An alternative is to remove the directories is **/usr/man** that start with the letters '**cat**'. These directories, when they exist provide a location for formatted help pages. When running the **man** program, you will see the message:

```
formatting...please wait
```

which comes from the **man** program as it turns the help pages into a readable format.

If the **/usr/man/cat*** directories exist, the finished pages are saved and thereby avoid the delay when the same page is requested in the future. A fully formatted set of **man** pages may be about 20 megabytes larger than the unformatted pages. If users are not annoyed with the delay, this can save an average of 10 megs.

Here's a tip: most users need only the section 1 or section 1m commands for day to day operations. As the system administrator, you may find that this space (approximately 3-4 megs) is well worth the time saved, as long as it doesn't grow any larger. There is a command called **catman** that can format complete sections (1 and 1m are the basic HP-UX commands for all users and system administrators, respectively) and by removing all **cat*** directories except:

```
/usr/man/cat1
/usr/man/cat1m
```

then just the pages for these commands can be formatted at one time (I would suggest doing this overnight) by using the **catman** command. Now, all the **man** pages for sections 1 and 1m pop up immediately and the disk space will not grow as **man** pages from other sections are referenced (they will still be formatted on each occurrence).

Discless

Discless is a feature found in all the 9000 platforms at revision 8.x but not everyone needs these features. **DISCLESS** is a fileset that is fairly large and can be removed if support for discless workstations is not needed. **DISCLESS** provides several kernels for various features that discless clients might need and just these 5 or 6 kernels will occupy 6 to 7 megabytes in **/etc/conf**.

NLS files

Native language support is another area that can be trimmed from systems that do not require language support other than English. Some files can be quite large for Far Eastern languages where a complex character set (ie, Kana or Korean) might be needed.

Diagnostics

These are tricky. Removing them can save a lot of space, especially on the 700 and 800 systems. On the other hand, they do provide service people with detailed information on problems that may occur on the system. You may wish to discuss the pros and cons of removing the diagnostics with your local support people.

As with all filesets, they can be reinstalled simply by running the update program and selecting the required fileset(s).

lost+found directory

During an abnormal powerfail or panic of the system, the filesystem will not be shutdown cleanly and this may require manual intervention with **fsck**, the filesystem fixit program. If **fsck** is unable to repair files or directories, rather than delete them, **fsck** will ask if you want to fix the problem. If you answer yes, then the inode (a pointer to files or directories) may be moved to the **lost+found** directory and given a name that is actually the inode number.

The space represented by these entries in **lost+found** may be temporary files that have been deleted but their space not returned to the filesystem, or just ordinary files and/or directories that have lost their names or their connection with the rest of the directories. In this case, the system administrator must look at each of the files to determine what, if any, data is to be saved.

Otherwise, these files will occupy space but serve no purpose, and might lead to creeping file system growth.

Unmounted disks

HP-UX connects separate disks into a single filesystem called root (and represented by the / symbol) by making directories do double duty. A directory can be changed into a mount point (a logical connection to another disk's filesystem) by using the **mount** command. The file **/etc/checklist** can also accomplish this indirectly in that the mount command reads **checklist** for guidance on where to mount disks.

The curious thing is that unmounting a disk returns the directory to a local status and files may be stored in that directory, which is in reality part of the root disk. These files become dormant after the **mount** command changed the use of the directory into a mount point; that is, the files on the root disk exist, but cannot be seen because of the mounted disk.

If the mounted disk is unmounted, the files in the root directory again become visible. This can lead to a common error when a mounted disk becomes unmounted:

1. Someone notices the files are missing and starts to load the files back from tape.

2. The files are reloaded (or a portion of the files are loaded) and someone notices that the root filesystem is full or close to full.

3. Someone types the **bdf** command and discovers that the second disk is not mounted...and mounts it. Now the files are back to what they were, but the root filesystem is still almost full.

What happened is that the directory was not in use as a mount point but there are no red flags showing this condition. That's why the **bdf** command is so imnportant: the mounted filesysytems are shown to the right of the listing.

Here's a tip: Bring the system down into single user mode by typing

shutdown 0 and once the shell prompt shows up (you won't have to login), find all your mount points by examining the **/etc/checklist** file. The second parameter is the mountpoint directory.

Now, check each directory to see that it is empty. If not, you will need to clean up the directory as needed, since no other disks are mounted except root. Now issue the following command for every mountpoint:

```
touch /mount_point/IamNOTmounted
```

This will create a zero-length file in the mountpoint directory to serve as a reminder that this is a mountpoint and not a general use directory. When disks are mounted, this file disappears from view; when a disk is unmounted, the file comes back.

Locations of other big files

Let's start with the files that seem to show up everywhere: **core, a.out** and ***.o** files. A **core** file is produced in the current working directory whenever a program is terminated abnormally, typically through some sort of error condition not anticipated by the program, or to a lesser degree, by receiving certain signals.

While these **core** files might be useful to a programmer that is designing or supporting a particular program, the files are generally wasted space and can be removed. **core** files can be a few thousand bytes or up to many megabytes in size. Here is a command that might be added to a **cron** entry to remove **core** files on a regular basis:

```
find / -name "core" -exec rm {} \;
```

Other files that are commonly left over from programming efforts are **a.out** and files ending with **.o** (which are compiled but unlinked files) and these are often left in various places by busy programmers. A polite way to notify users about these files is to send an email message to everyone with a list of the files:

```
find / -name ".out" -print > aout.list
find / -name ".o"  -print > start.o.list
```

This can prompt wayward programmers to clean up their files. If this effort is unsuccessful, the previous core-file-remover command can be changed from the name "core" to the names "a.out" and "*.o", although you may wish to add an aging option to the find command such that only files more than 30 days old are removed.

There is an interesting directory in /etc called /etc/newconfig and it contains some very useful files, namely, the unedited (known to work) custom files such as rc, gettydefs, inittab, passwd and so on. If one of these critical files becomes corrupt (ie, inittab), a know-to-work copy can be copied from /etc/newconfig which will allow the system to get back online. For 300/400/700 systems, always make a recovery tape (see the command mkrs) so your system can be booted and repair to your root disk can be done.

So what about users that are abusing their filespace with test files or other unnecessary data? First, where are these files? The simplest answer is to check the /users directories, and a good way to do this is with the du command. Here is a sample:

```
du /users
2          /users/root/test
32         /users/root
12         /users/rst
480         /users/wpw
2          /users/jes
3308       /users/djs/nova-files
10442      /users/djs
2          /users/mda
6          /users/jws
2          /users/gfm
2          /users/gedu
12         /users/jam
12         /users/blh
11016      /users
```

The numbers on the left are in blocks, or 512 byte units of measure. These values are not as meaningful as Kbytes or Mbytes so I simply divide the number in half and now I can see the usage in Kbytes. The du command shows the diskspace usage in directories, not individual files. This is the first step towards tracking down disk space problems.

Now, you'll notice that some directories are not really very interesting such as /users/root/test (2 blocks or 1 Kbyte), so how can we limit the list to interesting numbers, such as directories larger than 5 megabytes?

The du command uses left justified numbers, so the sort command won't produce the desired result. grep can be used in this case; here's an example:

```
du /users | grep ^.....[0-9]
```

The grep pattern says: starting at column 1 (^), skip the next 4 columns (ie, dots are don't-care positions), then match only when the 5th column contains a numeric character ([0-9]). So, the above command applied to our example above produces:

```
du /users | grep ^.....[0-9]
10442   /users/djs
11016   /users
```

which is certainly easier to read. Now it is obvious that /users/djs is the largest (5,221 Kbytes of 5 Mbytes) user in /users.

How about looking for big files? The find command has an option that will search for the size of a file in blocks or characters. For instance, to locate all files that are greater than 1 Mbyte, the following commands will work:

```
find / -size +2000 | pg
find / -size +1000000c | pg
```

where the first form specifies 2000 blocks (2000 x 512 bytes = 1 Mbyte apx) and the second form will find files that are greater than 1,000,000 bytes. In the man page for find, the use of + to mean greater-than-or-equal is documented at the beginning of the section. You may wish to change the output from a pipe to pg (or the more command) to redirect into a file:

```
find / -size +2000 > /tmp/bigfiles
```

Some files need to stay, for example, /hp-ux and /SYSBCKUP are usually larger than 1 Mbyte but don't remove them! The system will have a difficult time rebooting when these files are removed (as some new system managers or well-intentioned users have discovered).

Logfiles - information and lots of space!

Many logs are kept in HP-UX systems and the majority grow without bounds, which can mean the infamous "file system full" message starts showing up. The root filesystem is by far the most critical in that many HP-UX processes depend on having some space available, including space for logfiles. Many of these logfiles are optional and are not created in a default system, but there are several to watch for and to monitor.

/etc/netbsdsrc starts **rwhod** by default (7.0). **rwhod** makes entries in **/usr/spool/rwho** for every machine it discovers...files may be deleted and rwhod may be removed from **/etc/netbsdsrc** as a startup daemon. **rwho** also generates a lot of LAN traffic. At 8.0 and higher, **rwhod** is not started by default and you may wish to leave it disabled.

In general, most system logs are kept in **/usr/adm**, but as with all HP-UX commands, there are exceptions. The **update** and **rmfn** programs keep logs in **/tmp** (ie, **update.log** and **rmfn.log**), as well as the new JetDirect Network printer software which keeps logs in **/tmp** also, each starting with the name **hpnp***.

One of the big logfile makers is the program LaserRX/UX which logs all system activity. Depending on the settings used to quantify 'interesting' processes, the **rxlog** may grow very rapidly, especially if the CPU and disk parameters are set to zero (ie, every activity is logged regardless of importance). Recommended settings are CPU = disk = 5 which is more than adequate to track important activities.

In the following table, commands that have logfile options is listed followed by the location and name of the logfile (user-defined means that the path/filename is defined at run time), a key (K) to define conditions by which the file is created and a short description of the contents.

Command	Location	K	Contents
<acctg>	/usr/adm/pacct	P	system accounting
<acctg>	/usr/adm/acct/*	P	system accounting
backup	/etc/backuplog	A	history of /etc/backup script
cron	/usr/lib/cron/log	A	history of cron activities
dmesg	/usr/adm/messages (typ)	U	dmesg text
dmesg	/usr/adm/msgbuf (typ)	U	incremental dmesg text
eisa_config	/etc/eisa/config.err	A	eisa_config error log
eisa_config	/etc/eisa/config.log	A	eisa_config activity log
gated	user-defined	P	gateway routing, changes, etc
getx25	/usr/spool/uucp/.Log/*	A	x.25 PAD caller numbers
getx25	/usr/spool/uucp/X25LOG	A	x.25 activity log
glbd	/etc/ncs/glb_log	A	GLB diag informaton

Command	Location	K	Contents
hpnpcfg	/tmp/hpnpcfg.log	A	JetDirect configuration log
hpnpinstall	/tmp/hpnpinstall.log	A	JetDirect installation log
hpnptyd	user-defined	P	JetDirect printer pty logging
hpterm	user-defined	P	hpterm log file
kermit	./debug.log	P	kermit log file
login	/usr/utmp	R	built on every bootup
login	/usr/wtmp	R	history of logins, state change
login	/usr/btmp	E	list of bad logins
lp	/usr/spool/lp/log	P	lp activities
lockd	user-defined	P	RPC lock request errors
mountd	user-defined	P	NFS mount errors
netdistd	usr/adm/netdist.log	P	log of netdist activities
nettl	/usr/adm/nettl.LOG0...	P	network tracing/logging
notes	/usr/contrib/lib/notes/*log	P	notes activity log
ns	/usr/adm/nettrlog	P	NS network errors
opx25	/usr/spool/uucp/.Log/*	P	x.25 (HALGOL) logging
ppl	/usr/spool/ppl/log	E	x.25 ppl log
ppl	/usr/spool/ppl/bill	E	x.25 ppl log for billing
ptydaemon	/etc/ptydaemonlog	P	log of ptydaemon activities
rexd	user-defined	P	RPC activities
rlpdaemon	/usr/spool/lp/lpd.log	P	rlp activities
rmfn	/tmp/rmfn.log	A	log of rmfn activites
rstatd	user-defined	P	RPC performance statistics
rusersd	user-defined	P	RPC users error log
rwalld	user-defined	P	RPC rwall command errors
rwhod	/usr/spool/rwho	A	list of machines on the LAN
rbootd	/usr/adm/rbootd.log	P	rbootd activities
sam	/usr/sam/log/samlog..	A	log of all SAM activites
sam	/usr/sam/backup/logfile	A	log of SAM backup activities
sam	/tmp/cluster.log	A	log of SAM cluster activities
savcore	/usr/adm/shutdown.log	E	log of coredump saves
scopeux	/usr/bin/rxux/rxlog...	A	LaserRX/UX collection files
sendmail	/usr/spool/mqueue/syslog	A	sendmail history
shutdown	/usr/spool/shutdownlog	E	log of shutdown activites
spell	/usr/lib/spell/spellhist	P	history of spell activities
su	/usr/adm/sulog	A	history of su command usage
update	/tmp/update.log	A	history of update activities
updist	/tmp/update.log	A	history of updist activities
uucp..	/usr/spool/uucp/diallog	A	uucp dialing log

Command	Location	K	Contents
uucp..	/usr/spool/uucp/errlog	A	uucp error log
uulog	/usr/spool/uucp/logfile	A	uucp general log
uucp..	/usr/spool/uucp/syslog	A	uucp system log
uucp..	/usr/spool/uucp/culog	A	uucp call log
uucico	/usr/spool/uucp/.Log/...	A	uucico transactions log
uusub	/usr/spool/uucp/L_sub	A	uucp connection statistics
uusub	/usr/spool/uucp/R_sub	A	uucp traffic statistics
update	/tmp/update.log	A	history of update activities
vtdaemon	/etc/vtdaemonlog	A	log of vtdaemon activities
vtdaemon	/usr/contrib/lib/*.log	A	log of vtdaemon diagnostics
Xserver	/usr/adm/X*msgs	P	X/Windows error log file
x29server	/usr/adm/x29/x29server/*	A	x29server logging
ypbindm	user-defined	P	Yellow Pages bind log
ypserv	/usr/etc/yp/ypserv.log	E	Yellow Pages server log
ypxfr	/usr/etc/yp/ypxfr.log	E	YPages database transfers

Notes: The K column refers to special conditions for the file's existance:
A = automatically created or appended
E = not created automatically; log is kept only if it exists already
P = program creates logfile by runtime option only
R = required; zeroing or deleting the file may cause a problem
U = user sets up logging through cron or other means

As a final note, HP's new RemoteWatch now offers features to monitor log files, the **/dev** directory, and all the mounted disks automatically and will notify the root user when a problem begins to occur by using email. RemoteWatch information can be obtained through your local HP sales office or the HP Response Center.

Also, SAM is being enhanced to perform many of the big file searches and will offer other disk space management tools in the upcoming HP-UX revision 9.0 release.

Bill Hassell
HP-UX Technical Support
Hewlett-Packard Response Center
Atlanta, Georgia 30339
(404) 850-2181

Help! My Filesystem Is Full And I Can't Get Up!

2057-15

#3001

Robert Lund
34130 Parkwoods Dr. N.E.
Albany, Oregon 97321
(503) 327-3800

An Overview Of Queue Network Modeling Capacity Planning

Abstract: *Within this paper I will present some ways in which modeling techniques can be used in predicting the future performance of an HP 3000. Managers are often faced with questions like, "What will be the impact of a new application?", or "How do I know for sure how long an upgrade will last my company?" I will present some of the problems involved in selecting an upgrade, and how to know for certain when an upgrade will be necessary. In these times there are many new, and often confusing, choices as far as hardware upgrades are concerned. Modeling techniques can be used to help insure that the existing hardware is truly out of gas. These techniques can provide you with some of the information you need so that you don't either over or under buy on a new system.*

This area of Capacity Planning is particularly exciting to me. For some time now, I have had an interest in helping managers predict the future performance of their HP 3000 systems. In this paper I will briefly cover some common types of models used for performance evaluation and capacity planning, but will focus on queueing network modeling. This area is fairly immature for the HP 3000 market, though it is quite developed in the larger IBM mainframe arena. There are quite a few commercial modeling products available for "big blue" systems, but only one for the HP 3000. If you are not faint-hearted with math and queueing theory you could build a usable model yourself. So, I'll describe some of the basics of the math involved and provide some examples. Suffice to say for these introductory comments, queueing network modeling is undoubtedly the most cost effective and accurate way to predict the future performance of your systems.

Consider this comment by one capacity planning analyst:

> *"Significant savings, often millions of dollars, can result from capacity management software packages. As a result, senior management's views about capacity planning efforts are improving. Many companies are finding that the savings come from three major areas: maximization of computer resources, better planning for new computer resources in both what is needed and when it is needed, and increased productivity of both data center personnel and end users.* [1]

Definition And Importance Of Models

Let's first turn our attention to a definition of queueing network modeling.

"A Queueing Network is a particular approach to computer system modeling in which the computer system is represented as a network of queues which is evaluated analytically. A network of queues is a collection of service centers, which represent system resources, and customers, which represent users or transactions. Analytic evaluation involves using software to solve efficiently a set of equations induced by the network of queues and its parameters". [2]

If you think of queue modeling in light of the preceding definition, it really is quite a simple concept. There are many day-to-day queue networks that we are involved with. I'm thinking of things like fast food restaurants or making a phone call from the east coast to the west coast on mother's day ("I'm sorry, all circuits are busy..."). Analysts in charge of strategic planning for these types of systems often utilize queueing network modeling to solve their growth pains.

There are a few major benefits of using an analytic model to perform capacity planning. Queueing network models

- Provide an improved understanding of the physical system.
- Handle non-linear interactions of many variables.
- Have proven to be the most accurate way to predict future capacity needs.
- Relate capacity, throughput and service.

It is very difficult to evaluate the trade-offs of many different system relationships in your head. Consider the effect on user response times if you were to place an interactive workload at a lower priority while at the same time adding a new application and applying staircased business growth rates for the next two years. About this time your brain goes tilt... But this is precisely the type of real-world questions that business planners have to answer. The solution to these questions and more is queueing network models.

Such modeling can provide significant information to financial analysts.

> *"Analytic modeling packages can be used to explore further when the next upgrade will be required as well as the life-expectancy of each upgrade. Cost of the equipment, when it will be needed, and most importantly, how long it will last, are the three factors needed by the financial analysts of the organization. The more advanced notice of equipment needs that the financial side of the organization receives, the more negotiation leverage it has with the hardware vendors. More accurate information on how long equipment is expected to last gives the financial side the information that it needs to determine how the equipment should be financed and depreciated. "[3]*

Types Of Performance Models

There are four types or levels of models that people utilize to make decisions about their systems' performance. These are:

- **Implicit Models** - These are ones you probably do not realize you're using. An example might be something as simple as, "A Series 937 is faster than my 922" or "The users begin to complain about response time just after lunch time; perhaps we should move that payroll batch job to after hours". You have such simple models constructed in your brain. These models are examples of those that we create empirically and trade with one another. Implicit models represent a knowledge base that you have acquired which takes into account such subjectives as the user population, application quirks, hardware configura tion, management/political restraints, etc. These models are sometimes intuitively obvious, but nevertheless most of us utilize them.

- **Rules of Thumb** (ROTs) - ROTs typically consist of threshold metric information. An example of such a ROT might be, "A sustained page fault rate of greater than 30 per second may be an indication of memory shortage". Overall, these models are fine for simple situations, but really do not address some of the tough futuristic business planning that is required of DP management these days.

- **Simple Queueing Models** - These models utilize relatively simple math (similar to some that I will be discussing shortly) to construct a picture of your system and then allow some prediction. They are typically low cost and come with low accuracy. Also, much skill is necessary since you'll probably be programming them yourself.

- **Detailed Queueing Theory** - Good accuracy and easy to use are the traits of these models. Consequently, someone had to build in this accuracy and user-friendliness. Subsequently, these cost more since they are usually commercial (versus shareware, etc.) packages.

Modeling Functions And Steps

Some of the questions a queue network model created for your HP 3000 can answer are:

- What happens to response times if we upgrade?

- How much will our batch financials improve if we decrease the amount of CPU necessary for on-line order entry?

- Will the addition of a particular application six months from now kill our three year plan for the expected life of the current system?

- At what date will our CPU be saturated given a 20% annual growth in orders?

Basically, models help most of the questions relating to the how, what, when, where and why of performance capacity planning. The actual steps involved in forecasting performance with a queueing model are as follows:

1) Define and set up workloads. This step requires a knowledge of your applications and much patience. You might have to re-define them and re-measure them numerous times before they are right for modeling.

2) Run Data Collector on Host.

3) Perform analysis of collected data to find appropriate modeling window and obtain estimates of service demands (CPU and Disk). Identify the peak period best for modeling.

4) Enter data into modeling formulas.

5) Validate initial model results with actual observations during analysis phase.

6) If validation is within acceptable limits, proceed to forecast. If validation is not good, then you must consider the following:

- **a)** Check for dispatcher queue aberrations (third party queue management tools), changes in queue overlap, multiple C or D subqueues, modeling assumes no queue tricks.

- **b)** Miscellaneous or unknown process wait time for resources such as tape drives, Father/son waits, database locks, etc.

7) Create forecast of system and try various "what if" scenarios.

It is important to keep in mind that modeling is most accurate when considering relative changes in response time, throughput, and utilizations. That is, even though a model may speak of response times as 2 seconds now and 4 seconds 9 months from now, the actual numbers may not be exactly what the users are experiencing. It is the relative change in response that is important.

The HP 3000 As A Queueing System

The behavior of a queueing system, as applied to the HP 3000, is really quite simple. Customers (transactions) arrive at the system from terminals or batch jobs. These customers then request service at the CPU, then at other devices. They must queue if those devices are busy. When their processing is finished, customers depart the system (back to terminal or batch spool). In a picture, Figure 1.1 illustrates a bit more technical aspect of a queueing network model.

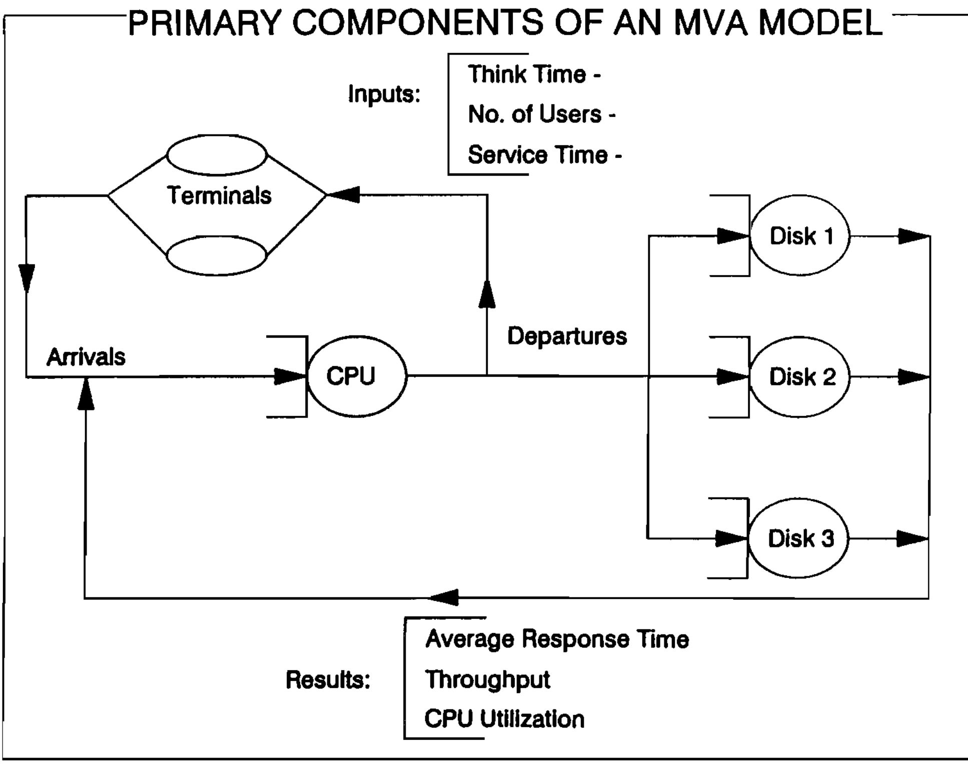


Figure 1.1 - Queueing Network Model

Basics Of Queueing Theory

If you don't like math, even basic math, you better skip the following section! However, if you can handle simple algebra, and if you are intrigued by the subject so far, your in for a treat. I will first lay out some formulas and then illustrate how simple queueing theory can be used to solve some basic problems.

Modeling Assumptions And Formulas

Single Server Model (M/M/1):

Assumptions:

A = Average Arrival Rage (per second)

S = Average Service Time (seconds)

U = Server Utilization (percent)

R = Average Response Time (seconds)

Q = Average Queue Length

Formulas:

R = S/ (1-U)

Q = U/ (1-U)

U = A * S (Utililzation Law)

For Instance:

A = 10, S = .05 seconds

U = 10 * .05 = .5 = 50%

R = .05/ (1-.5) = .1 second

Figure 1.2 - Modeling Assumptions And Formulas

Given some of the basics in Figure 1.2, we can calculate some pretty exciting answers to many otherwise unanswerable capacity planning questions. What HP 3000 IS folks care about most are things like utilizations, throughputs, and response times; predictions of these things, that is. By simply changing a few variables in such a model, you can come up with some very relevant information. Although it is very basic, the following example will help illustrate this point.

Simple Queue Model Example: XYZ Company's Order Entry Growth

The MIS director of XYZ company wants to know what will happen to their HP 3000 if 60% more data entry activity is added to their system, effectively raising the number of transactions per minute to an estimated 200. The current number of orders per minute is 125. The development team concludes that the appetite of the "average" order transaction is .27 CPU seconds.

Given this scenario:

- What is the current average CPU utilization and response time?
- What will the new average utilization and response time be?
- Will the current system be less than 95% utilized?
- Will the response times be less than their 3 second target?

Let's utilize some of the basic math in Figure 1.2 to answer some of these questions.

First we need to obtain the current CPU utilization and response time. Given the arrival rate of user requests and the service time of each transaction, we can do just that.

Transaction arrival rate:
$A = 125$ tran/min $= 2.1$ tran/sec

CPU service time per transaction:
$S = .27$ sec/tran

CPU Utilization:
$U = A * S$ $U = 2.1 * .27$ $U = 56.7\%$

So, the current CPU utilization is 56.7%. Now for the current response time:

$$R = S/(1-U) \quad R = .27/(1-.567) \quad R = .62 \text{ seconds}$$

The current average response time of each user transaction is .62 seconds.

Now we can calculate the proposed impact of the above stated growth on this system.

Projected arrival rate:
A = 200 tran/min = 3.3 tran/sec S = .27 sec tran <the appetite of each transaction hasn't changed>

Projected CPU utilization:
U = 3.3 * .27 U = 89%

The projected CPU utilization is 89%. And for the projected response time:

R = S/(1-U) R = .27/(1-.89) R = 2.45 seconds

The answers to the above questions are:

- YES, the CPU will be less than the 95% target (barely!)
- YES, the response times will be less than 3 seconds (barely!)

What happens if a new version of the order entry application is proudly introduced with all the new enhancements you asked for, but there is a slight increase in the amount of CPU needed by each transaction in amount of a mere er..uh..32% more? Now what happens to service level targets? A little math will show that the CPU will not be able to handle this new load since the utilization goes above 100%.

You can see that even with simple queueing theory math some real world answers can be attained. We obviously haven't taken into consideration a number of issues that cannot be ignored for the most accurate queue models. Some of these are:

- Multiple workloads
- Priority interaction between workloads
- The effect of other servers (disc drives, most notably)

More complex models require fairly complicated math. To illustrate the power of such models I will utilize Forecast/3000 Capacity Planner. Various "what if" situations will be presented for a hypothetical company's growth estimates.

Capacity Modeling For A Typical Company

Let's use a hypothetical company, **Progress Corporation**, to demonstrate the power of queueing models for capacity planning.

Progress has three workloads initially, each with the following characteristics:

Workload	#Users	Priority	CPU Hog
Order Entry	28	High	Medium Heavy
AR-AP	9	High	Medium
Shipping	16	Medium	Light
Payroll (fut.)	4	High	Heavy

Table 1.1 - Workload Characteristics

Progress has just upgraded from a series 925 HP 3000 to a series 947. They experienced considerable performance improvement. The users were ecstatic. However, knowing that this was only a temporary euphoria, the system manager decided to implement a capacity plan. He felt that such a plan was warranted based on some pretty aggressive business growth plans and an additional application management wanted to implement in about eight months.

Figures 1.3, 1.4, and 1.5 show the initial state of their 947. They wanted to be around 50% CPU utilization. As you can see, their response time targets for each workload (less than 2 seconds) are attained with room to spare.

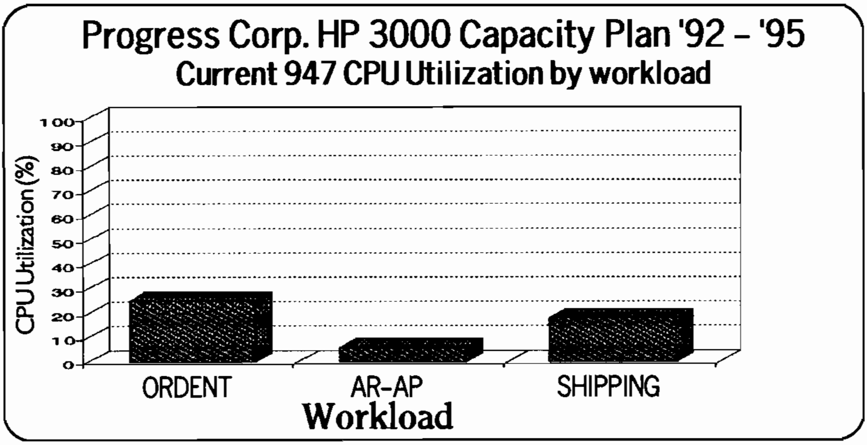


Figure 1.3 - Initial 947 Workload CPU Utilization

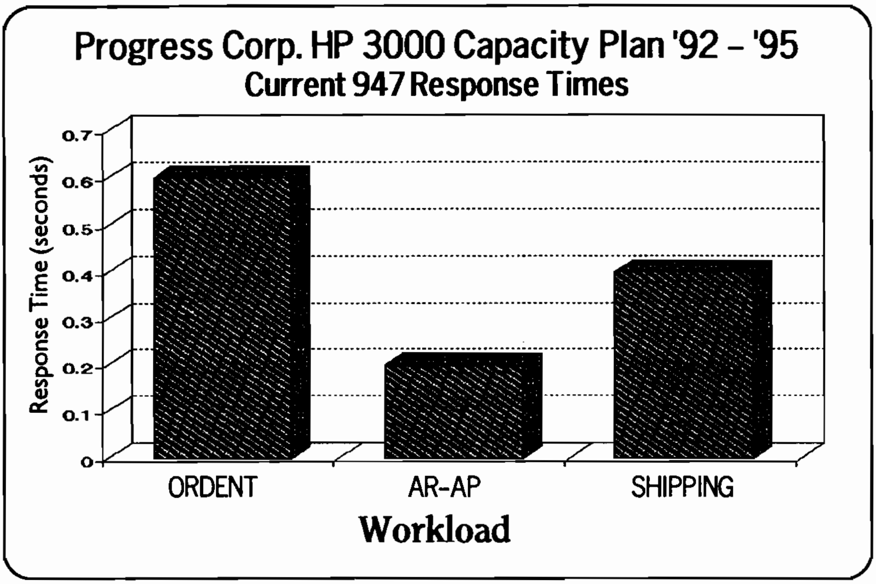


Figure 1.4- Initial 947 Workload Response Times

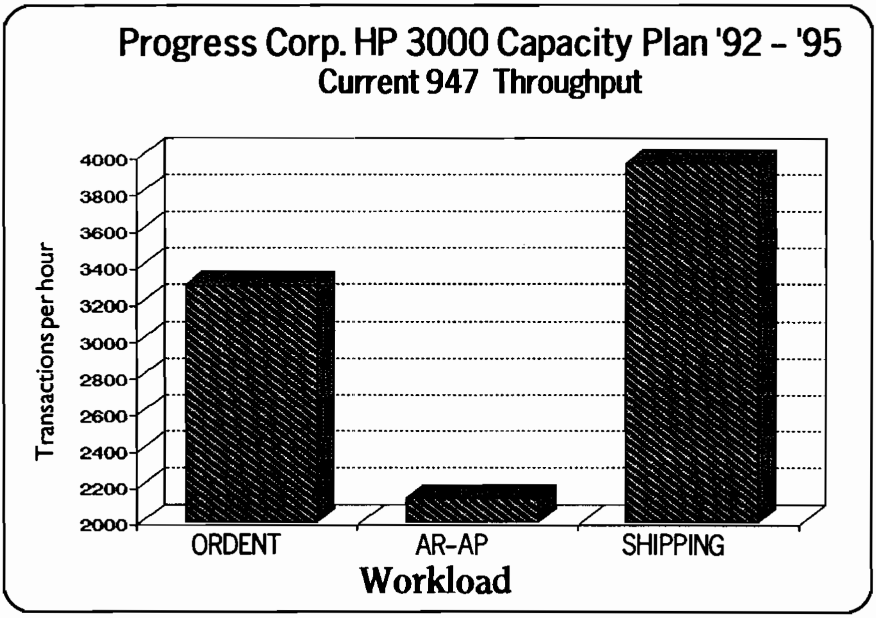


Figure 1.5 - Initial 947 Workload Throughput

The Company's Growth Plan

Progress hoped that the new system would last until the end of 1994. Their growth plans for the business were set to take off in about 6 months at a fairly steep rate; then it leveled off again. Figures 1.6 and 1.7 show their growth and resulting CPU utilization along with response times. More notes? The CPU does not pierce the 85-90% level until sometime after 1994. Notice how response times (Figure 1.7) for all workloads stay under two seconds until shortly after June 1994. The shipping workload experiences a "knee in the curve" phenomena which shows its responses piercing well above the service level target of two seconds. This is due to the fact that the shipping application is treated by MPE (on this system) as a lower priority (see table 1.1). This means that, in a pinch (CPU getting scarcer), shipping will receive less service.

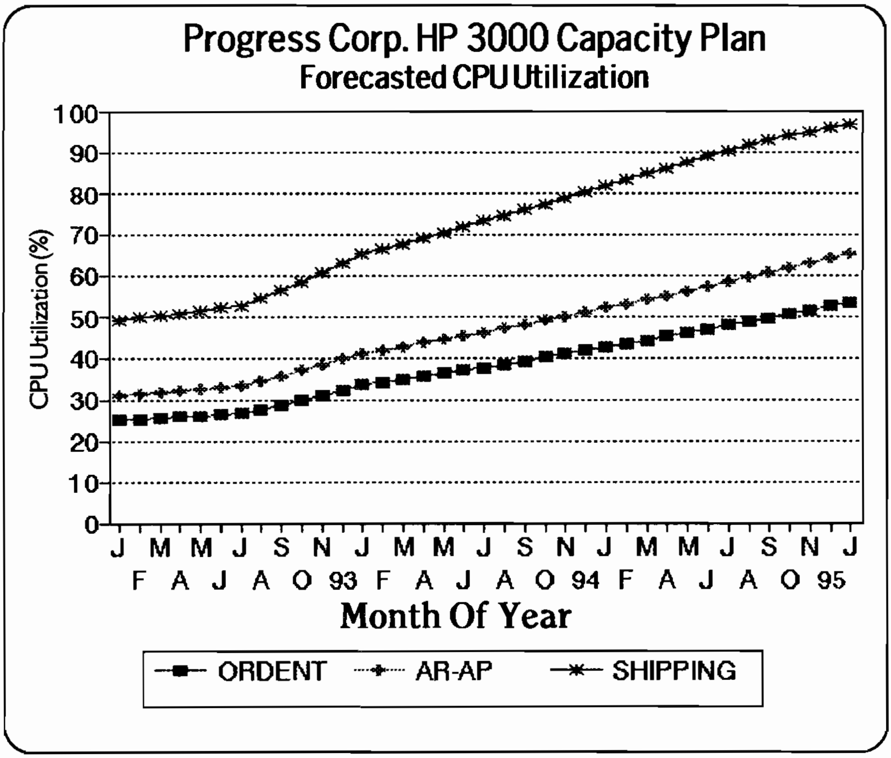


Figure 1.6 - Three Year 947 CPU Forecast

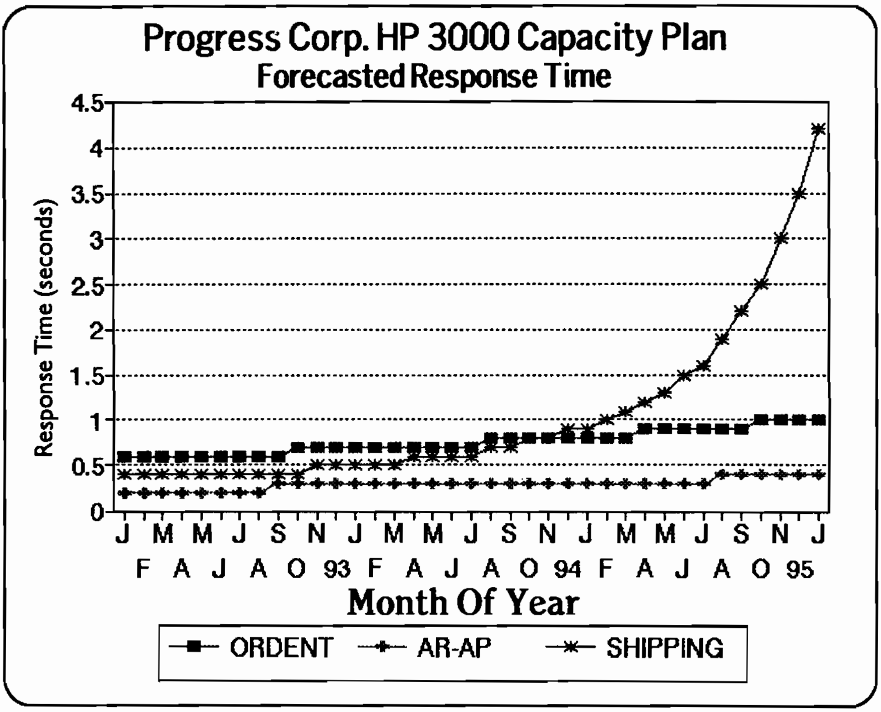


Figure 1.7 - Three Year 947 Response Time Forecast

The Plot Thickens...

But what about that payroll application management wanted to implement in eight months? As an unknown quantity, the system manager was a bit nervous about the impact of this application, and rightly so. New applications, especially if you have little or no control over the source code, can produce incredible surprises. Management said that it couldn't be that big of a deal especially since there was only going to be about four users or so. How could it impact the system that much?

When you look at Figure 1.8 you may first think of some kind of tumor that grew at the eight month mark! That tumor is the payroll application. Does it's presence affect the don't-die-till-95 plan? YES! How much an application will affect a system is purely an issue of its appetite. Queueing models portray these surprises, and more, quite nicely.

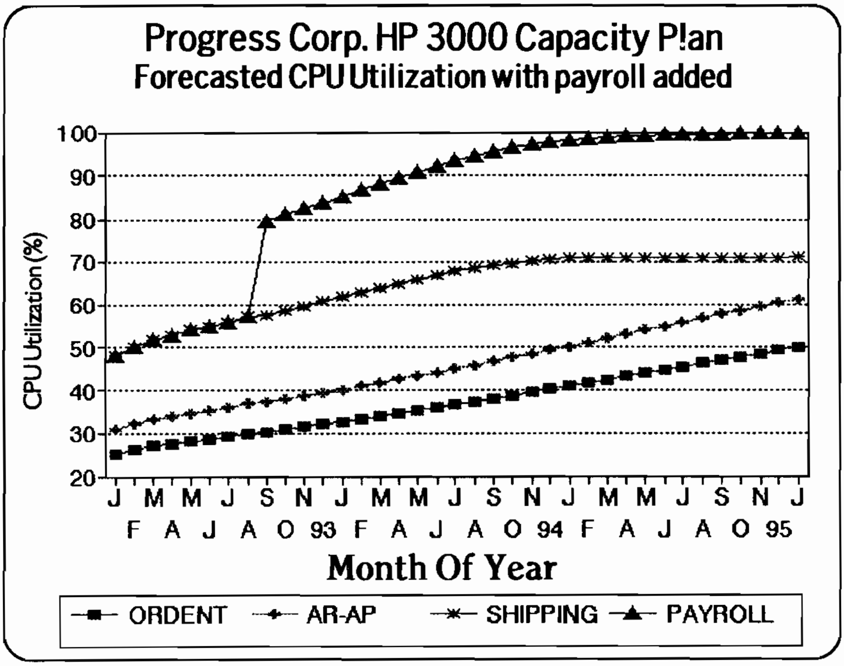


Figure 1.8 - CPU Utilization With Payroll Added

For **Progress**, the CPU approaches saturation right around the January '93 time frame.

Figure 1.9 illustrates the severely impacted response times for both shipping and payroll. Right out of the gate payroll's response time is not so great!

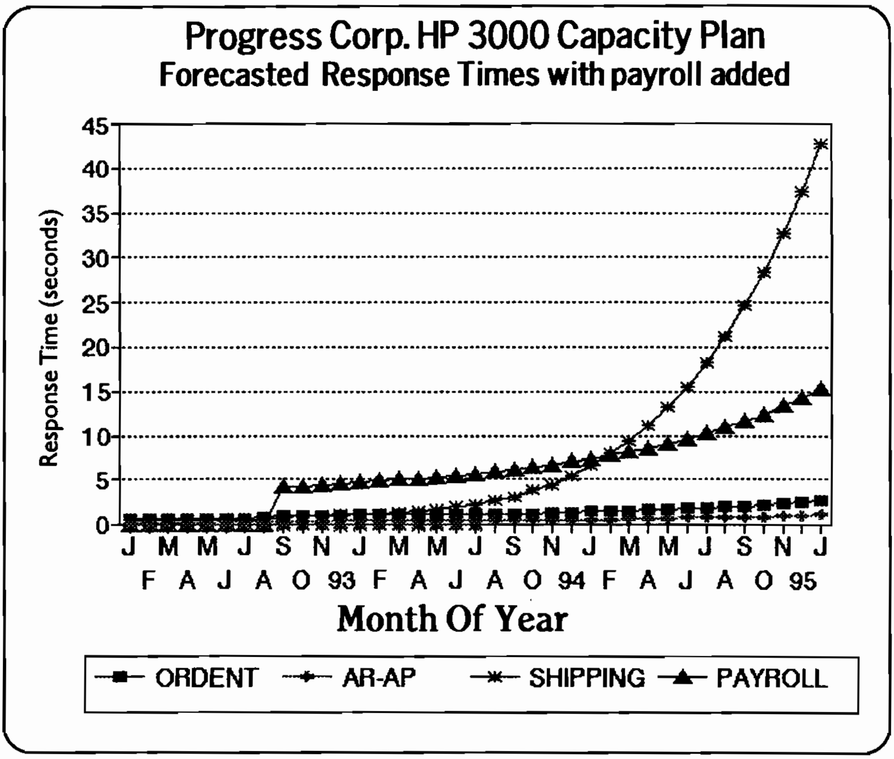


Figure 1.9- Response Times With Payroll Added

Programmers To The Rescue...

At a staff meeting the manager presented the above graphs and asked if anyone had any ideas how to last longer on the 947 even with the payroll implementation. A couple of programmers mentioned that there had been some known performance bugs with the order entry application. Spending about a month and a half would reduce the amount of CPU per order transaction by about 35%. This might help.

After some "what-if "modeling, the manager decided it would help enough to give them some more time. Figures 1.10 and 1.11 are the result of modeling 35% less CPU for each order. The net effect of this effort is that it would provide about seven more months of useful life for the system as a whole. Such modeling will allow you to determine the effect on key performance indicators before investing the actual time and effort necessary to implement them.

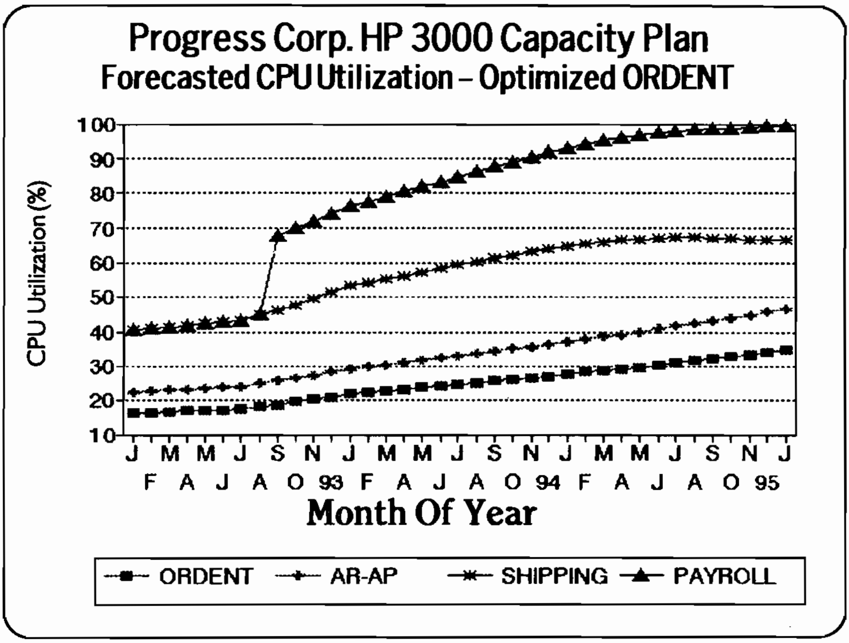


Figure 1.10 - CPU Utilization After Optimizing Order Entry

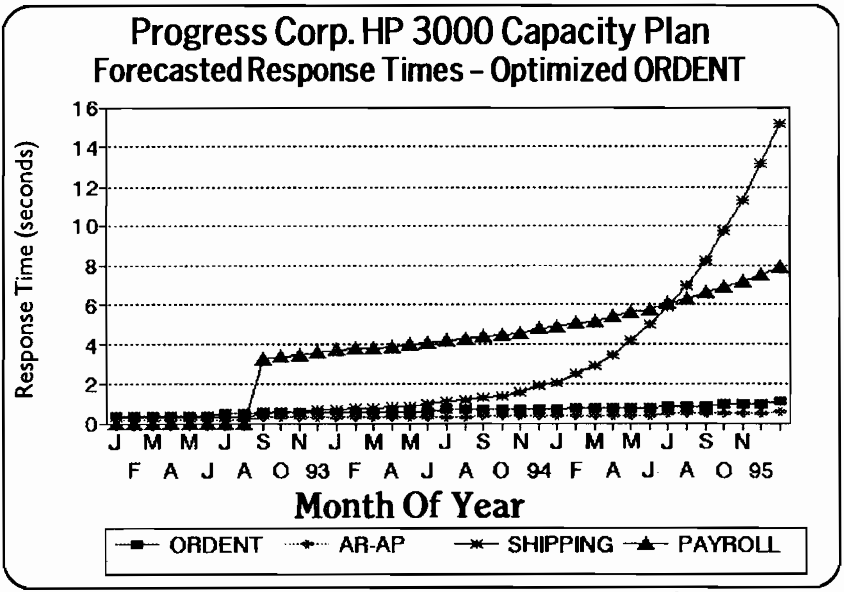


Figure 1.11 - Response Times After Optimizing Order Entry

As A Last Resort...

Progress Corp. thought they better also consider the impact of upgrading to a larger system. They chose to model the effect of a series 967 HP 3000. Figures 1.12 and 1.13 show how much improvement in CPU utilization and response times could be expected with an upgrade to the 967.

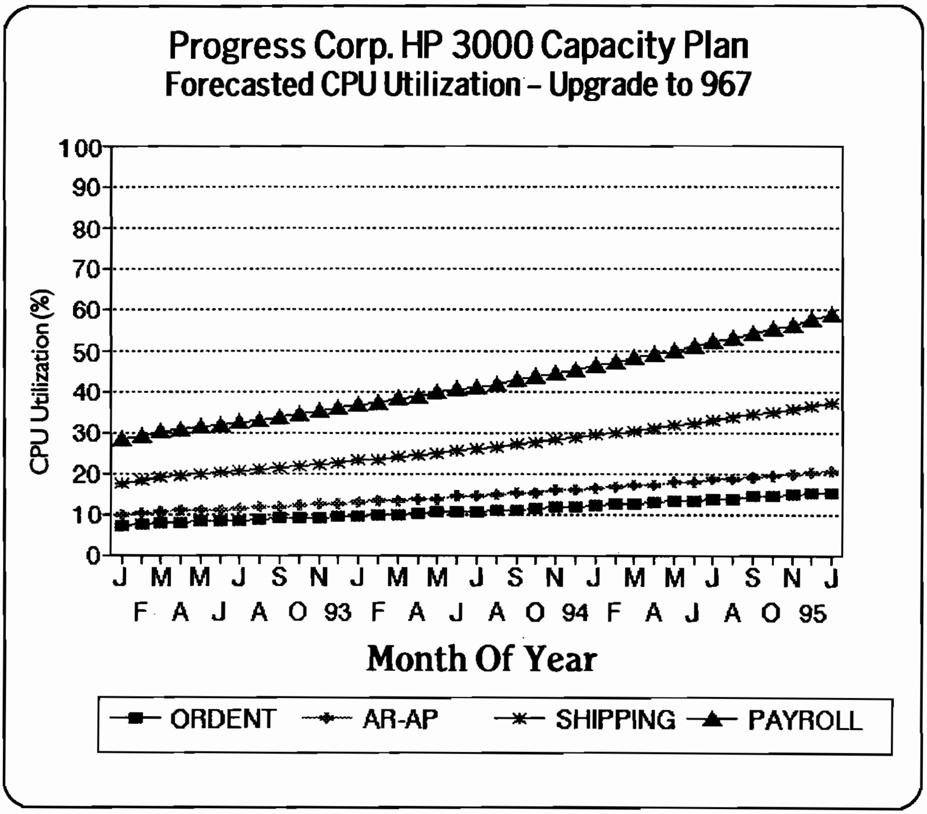


Figure 1.12 - CPU Utilization After 967 Upgrade

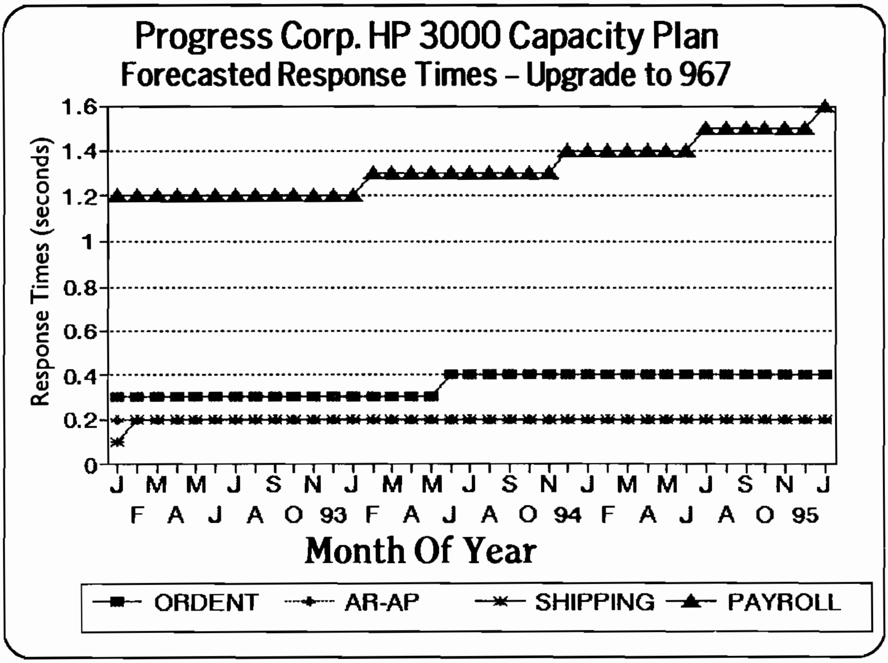


Figure 1.13 - Response Times After 967 Upgrade

In Conclusion

I hope this brief overview of queueing network models has been helpful in showing you a few things. These are:

- Models are increasingly finding a place in the HP 3000 market. This is mostly due to the vastly expanding options within the HP 3000 product line.
- Models are extremely flexible and powerful in their foretelling. A model can predict the effect of numerous changes to an application, upgrade scenarios, downsizing, etc. All without investing any time and money (apart from some time playing with a model on a PC).
- Models can be complex to "homebrew" but commercial pack ages are fairly easy to use.
- Models allow IS managers to chart a fairly predictable course which will allow knowing way ahead of time when the next performance "bump" will be encountered.

References

[1] Dave Porter, Technical Support 8-88 "Modeling: The Key to Effective Capacity Planning", page 30

[2] IS Capacity Management Handbook Series - Volume 1: Capacity Planning, page 6.23

[3] Dave Porter, Technical Support 8-88 "Modeling: The Key to Effective Capacity Planning", page 31

About The Author

Robert Lund is the president of Lund Performance Solutions, which specializes in system performance software, consulting, and training for the HP 3000. Robert is the author of Taming The HP 3000 Volumes I and II. He has designed and led development of three major software packages, SOS/3000 Performance Advisor, Q-Xcelerator Resource Manager and FORECAST/3000 Capacity Planner. He has been teaching seminars in the U.S. and Europe since 1988 and is considered an expert in the field of HP 3000 performance.

If you would like to discuss this paper or other issues with Robert, he may be reached at (503) 327-3800.

Notes...

3002

How to Win at Your Next EDP Audit!

Betsy Leight
OPERATIONS CONTROL SYSTEMS
560 San Antonio Road
Palo Alto, CA
(415) 493-4122

As data center manager, you often walk a precarious tightrope, successfully meeting corporate objectives while directing the day to day processing activities of a busy data center. And during your daily review of the data center, some concerns may go unnoticed. These are the areas most often uncovered during an EDP audit, and they can result in the imposition of unnecessarily stringent requirements on the data center.

This paper attempts to turn a tough assignment — facing an EDP audit — into a routine exercise by focusing on those concerns most often overlooked by the data center manager. By understanding what the EDP auditor looks for during a review, you will have the knowledge to prepare your data center to pass an audit with flying colors. This paper will also arm you with a checklist for improving and automating your operations, and help you to successfully anticipate even the toughest EDP audit.

PROBLEMS ADDRESSED

During a review of the data center, the auditor is chiefly concerned with the efficiency and security of DP operations in the following areas: standards and procedures, operational work flow and controls, scheduling, data security, change control, equipment utilization and efficiency, disaster planning and recovery, and environment.

If the data center manager and staff understand what the auditor is looking for and what information is needed by the auditor, the review can proceed more smoothly, and the results can be more beneficial to the organization.

STANDARDS AND PROCEDURES

Standards and procedures that should be in place and enforced include:

- Ensuring proper timing in running programs, inserting changes into programs, and using the correct data for programs
- Protecting the data and programs from accidental or intentional destruction
- Ensuring that the data processed is complete and accurate
- Specifying methods of physically moving input and output
- Scheduling work and getting work rerun in the event of an error or disaster

- Specifying procedures for controlling data, programs, and the flow of work
- Keeping records of work performed
- Determining and recording sufficient resources for the work
- Performing maintenance and general housekeeping associated with the operation of the computer center

The EDP auditor may also want to verify that formal standards exist for systems development and maintenance, program and system change control, library operations, computer operations, and documentation.

OPERATIONAL WORK FLOW AND CONTROLS

The auditor will investigate specific items in this area, including whether:

- Input data from other departments is complete and entered on time
- The data center keeps job accounting information
- Job accounting information is evaluated and used by management

Error control procedures should also be reviewed. Specific questions asked include:

- Is anyone notified in case of a production processing error?
- Are errors documented?
- Are error statistics accumulated or ignored?
- Are errors followed up on so that they do not recur?

The auditor will also confirm that downtime is reported and statistics compiled. A log of late reports and jobs should be maintained.

There should be a formal communications channel between data center operations and other departments; operational tips and other advice should be passed to all operators.

All problems encountered at the computer, as well as any action taken to prevent their recurrence, must be documented. Operators must also receive feedback on reported problems. The auditor will verify that headers and $STDLIST information is used and checked.

Next, the auditor scrutinizes output report distribution and disposal and determines whether:

- All reports have been distributed to the proper user
- Procedures have been established to control the distribution of sensitive output
- Procedures exist for disposing of confidential reports when they are no longer required

Finally, the auditor will want to ensure that jobstream run instructions are kept up to date.

SCHEDULING

Efficient and effective scheduling is extremely important in providing a high level of reliability and predictability to DP operations.

The auditor will determine whether:

- Daily processing activities are scheduled and a daily contingency schedule is maintained. A strict schedule for nightly batch runs should also be established and adhered to.
- Actual run times are recorded for batch programs.
- This data is used to calculate expected run times to ensure that runs have not been terminated abnormally.
- Unscheduled runs are supported by a work request or other written authorization. Schedule deviations should be documented and followed up on by a supervisor.
- User-submitted jobs are recorded to allow forecasting of future schedules, resource requirements and special processing considerations for online systems.
- All jobs are submitted through or controlled by data center operations. All output should be routed by operations to the appropriate destination or picked up by the user.
- Standards cover the type, quality and quantity of forms kept on hand.

DATA SECURITY

Data base information should be protected from unauthorized access or loss. Employees must be instructed about their responsibilities concerning confidential information. Management must periodically review and update controls and security provision relating to data. Live production programs should be physically separated from development programs. The staff should be prohibited from running test programs against live files, and operations personnel should be denied access to sensitive data files.

To maintain security, operators should be prohibited from renaming or transferring programs without supervisory approval. Internal labels must be used for all data and program files.

Passwords and lockwords should be used to protect accounts, users, and data files. Passwords, lockwords, dates, and constants should be introduced at run time, eliminating the need to hard-code sensitive data into jobstreams.

Access violations must be logged and reported to the security manager. An automatic log-off feature prevents unattended terminals from posing security threats. The auditor should examine the area above the suspended ceiling in the computer room to confirm that it is accessible only from that room.

The auditor investigates blank check stock and other negotiables to determine whether they are issued on a run-schedule basis, kept in a secure area when unattended, controlled by access forms, and periodically inventoried.

Every site's security needs differ according to hardware, business focus, personnel, system function, work schedules, work environment and numerous other variables. Having stated this, there remain several constants which can help you determine your security needs. Take a moment to answer the following questions:

- Are data processing employees instructed as to their responsibilities concerning confidential information?
- Are live production programs physically separated from development programs?
- Are program library changes approved and accounted for?
- Are operators prohibited from renaming or transcribing programs without prior supervisor approval?
- Are internal labels used for all data and program files. Is an operations log maintained?
- Is the area above the suspended ceiling in the computer room accessible only from that area?
- Are blank checks and other negotiables issued on a run schedule basis only?
- Do you ever have the same password for more than thirty days?
- Is sensitive data endangered by sessions that remain on unattended terminals?
- Has your auditing firm asked for stricter reporting standards?
- Is management spending too much time implementing password changes?
- When a person who has access to sensitive information leaves your organization do you globally change passwords?
- Can users log on to any terminal?
- Are your ports being tied up by people who fail to log off?
- Can users log on to terminal from remote ports?
- Can users log on from remote ports at any time of day?
- Are additional passwords needed to log on remotely?
- Can users circumvent existing procedures to run jobs during off hours?
- Would your security be enhanced if passwords were not embedded into jobstreams?
- Do users have access to sensitive jobstreams?

CHANGE CONTROL

Change control procedures for computer programs should be established and followed. The intent of these controls is to prevent unauthorized, inaccurate, and unreliable program changes from being incorporated into the live production environment. Both scheduled and emergency changes must be appropriately controlled to maintain the ongoing integrity of software.

The auditor will check to see that the following techniques are in place to ensure that proper controls are being maintained over your program changes:

- Develop and adhere to formally approved written standards for all program changes
- Define and enforce procedures detailing who can initiate and who can authorize program change requests
- Describe and track the nature and reasons for proposed changes
- Enforce testing and acceptance procedures for all program changes including emergency changes
- Test all program changes under normal operating conditions
- Involve users in preparing test data and reviewing test results
- Investigate and correct all errors before transferring code to production
- Certify that all test results demonstrate adequate protection from fraud, waste, and misuse of the program
- Document all program changes and update appropriate documentation as changes are made
- Log all completed changes as well as those changes in progress
- Utilize a formal system to report all changes to users and project managers
- Enforce a checkout-checkin procedure that prevents a file from being simultaneously modified by more than one programmer
- Develop procedures to analyze whether other systems are affected by new program modification
- Retain and secure original source code until changes have been processed, tested and updated
- Limit the frequency of program changes, except for emergency cases
- Notify both the user and EDP project manager when emergency changes are made

EQUIPMENT UTILIZATION AND EFFICIENCY

Once it has been determined that the entire DP department is following a properly implemented set of standards and procedures, the auditor may wish to review equipment utilization.

The auditor will determine how much machine time is spent on reruns, whether reruns are analyzed, and whether certain jobs are especially susceptible to reruns. The auditor will also review programs or jobs for insufficient file design or utilization. Another area to check is the full multiprogramming capability of the system for batch production. The auditor will determine whether multiple jobstreams run concurrently and whether CPU-bound and I/O-bound jobs are mixed to maximize overall throughput. The auditor then reviews whether many jobs can be restarted without rerunning the entire job.

PERSONNEL USE AND EFFICIENCY

Review of personnel practices can be a sensitive issue. Key areas of interest include:

- Do operations personnel require extensive training and experience to be effective in processing daily production work? Is extensive knowledge of each application run necessary?
- Is there a system to schedule and monitor regular daily processing? Is the system effective? Does it operate without excessive human involvement? Or do operators spend a large part of their time tracking jobs in execution, replying to program messages, and changing job priorities? Must operators modify jobstreams at run times?
- Are all necessary tapes, forms, and other resources available when needed?
- Is there excessive turnover? Does daily production depend on specific individuals?
- Is the operations department treated as less important than the rest of data processing?

DISASTER PLANNING AND RECOVERY

This catchall category includes everything from proper insurance planning to physical security procedures. The auditor will want to determine whether the emergency plan is adequate in relation to the risk. This plan should be kept current and distributed only on a need-to-know basis.

The plan for off-site storage of files and documentation should specify:

- The conditions for use of off-site processing
- Processing priority for applications
- Resource requirements
- Job scheduling
- Run documentation
- Required tapes, forms, and supplies

Formal procedures for hardware backup should also be instituted.

ENVIRONMENT

The auditor may wish to review the work space to ensure that it is adequate for the number of employees. The environment should be neat, and supplies should be easy to locate.

Auxiliary items located outside the computer room, such as bursters and decollators, should be accessible for the flow of work in the department. Tapes, disks, and other storage media should be stored in a closed, fire-protected, limited-access area.

RECOMMENDED COURSE OF ACTION

By understanding what the EDP auditor looks for during a review, you can now prepare your data center to pass its next audit. The checklists provided in this paper will help you to successfully anticipate even the toughest EDP audit. The data center manager should be aware that the following more general advice can also greatly enhance the data center review:

- Provide the auditor with as much information as possible
- Implement a software system that leaves clearly defined audit trails
- Keep accurate records
- Maintain formal written standards and procedures
- Implement an effective data security system and maintain an emergency plan
- Follow the auditor's recommendations and procedures in preparing for future audits to ensure efficient and cost-effective operations

3003

Mainframe Software Management Techniques: What Every HP 3000 User Should Know

Betsy Leight
OPERATIONS CONTROL SYSTEMS
560 San Antonio Road
Palo Alto, California 94306
(415) 493-4122

INTRODUCTION

Today's HP 3000 professionals are becoming increasingly concerned about software and data integrity. And as a result, implementing a reliable method of tightening control over the changes to systems and applications has become a prime objective of many shops.

An isolated change may initially seem harmless, but the ripple effect upon interrelated systems can often be devastating. As the volume of code used in a shop becomes greater, the effect of these change can become staggering. With an industry average cost of $50 per line, the software investment of even a small site can be valued well into the millions of dollars.

Despite the enormous value of today's software applications, much of it is vulnerable to inadvertent errors. The reason: MIS departments often use tedious manual procedures to track program changes, and these procedures are seldom if ever followed faithfully throughout the organization. Without an improved approach to monitoring how and why changes are made, a company's entire computer and software asset are at serious risk.

First, consider the case of our firm, Operations Control Systems. With over 4000 products installed throughout the world, our customer support staff receives requests for dozens of modifications every month. At the same time as these maintenance and enhancement requests are coming in, our R&D team is also under pressure to develop new products. Shipping a new release represents a major project for almost every department within the company. Without an efficient means to manage changes, maintenance work would forever delay progress on new development. Through the task of managing our own internal software maintenance and release management process, we at OCS developed much of our initial expertise in the area of change management.

Through feedback from our customers, we started to refine these techniques and make them available to the general HP user. When we first surveyed our customer base in 1986, we found that the application maintenance backlog was not unique to software developers. Some customers who relied entirely on vendor-supplied software with seemingly minor modifications ran as many as

6 months to 2 years behind schedule on their maintenance work. Thus, while many non-data-processing executives think that their MIS staff is focusing on new applications, the reality may be that MIS is bogged down in maintenance and change related activities.

This highlights a second major need for change management. Change management improves the ability of the MIS department to react to user demands for program changes.

THE NEED FOR CHANGE CONTROL

In order to understand the change control problem and how to solve it, let's explore the nature of the problem in an uncontrolled environment. The problem stems from the fact that in many HP3000 shops, programmers can access production source directly. This can have several consequences.

First, consider an environment where multiple programmers make simultaneous changes to source code files. In this situation, the last programmer to update each file overwrites the changes made by other programmers. The result is frustration and wasted time synchronizing and retesting all of the changes.

Second, a careless programmer might inadvertently destroy the source code of a critical application. If a production error occurs and the source code cannot be restored easily, production could be delayed for hours or even days. The original author may not be available. The program may even have to be rewritten from an old copy, wasting development and test time and possibly reintroducing old bugs or creating new ones.

Third, the development procedures found in many shops eventually result in discrepancies between source and object code files. That is, the master (or official) source files may not recompile into the current production object files. Since it is almost impossible to recreate source from object code, days might be spent searching for the correct version, recompiling it and validating it against the current production code. And, with all of this effort, there will be no guarantee that the source will exactly recreate the current production code.

These situations point out a fundamental rule of good change control: Programmers should never have unrestricted access to production files!

At first glance, these problems may appear obvious, but the fact is that many HP 3000 data centers continually react to the problems they cause, rather than introduce standards and controls to avoid them. After all, a data center is there to support its users. This charter forces management to focus resources on meeting daily user requests, distributing reports, managing hardware and completing batch production, rather than on implementing long-term solutions to problems that are not well understood. Although many shops can run error free for months, an innocent looking error can snowball into a major catastrophe.

If this is the orientation of your shop, you might want to consider this fact: the efficiency and accuracy of your work depends upon the integrity of your software assets. Is your data center a candidate for better change control? Try taking the following test.

What software systems do you have?
Where and why were the last changes made?
What programs comprise these systems?
Where are the appropriate backup copies?
How often do these programs change?
Which user requested the last set of changes?
Who is responsible for the changes?
What is the status of a specific change request?

If you cannot answer all these questions readily, your shop probably needs a better approach to managing changes.

Proper change control provides immediate answers to these critical questions. The result is more than just visibility and auditability. It is more than data integrity and improved maintenance efficiency. A good change control system ensures that every program change is authorized and controlled, making it possible to quickly restore systems when necessary. Change control techniques also make it possible to monitor changes made to software from the earliest stages of development through the testing phase and all the way through the move to production process.

Change control also improves the Quality Assurance process. It allows operations personnel to test new systems, moving software from development to test, or from test to production. And, all of this will be done with complete audit trails that allow the identification of each entity promoted and tested.

Another important aspect of change control is provision for management inquiry in areas such as: the status of a change, the current version of a file, the change history of a file, and the impact of a proposed change. Supplying this information can often be a tremendous benefit to managing the software life cycle.

RECOMMENDED CONTROLS FOR ALL SHOPS

In order to achieve the benefits of change control, a shop should address each of the following five major categories:

Control through stages: Keeping control over the movement of software through the development and maintenance process.

Change history: Providing a complete history of every change requested and made to the source code.

Control over source and procedures: Maintaining control over users authorized to access source code and the processes that render source into executable form.

Security: Retaining control over the personnel who are authorized to make program changes.

Inventory: Maintaining an accurate inventory of all the programs, files and other components of each system.

A more detailed checklist identifying good change control practices appears at the end of this article. With these items in mind, lets look at the costs and benefits of manual change control techniques.

MANUAL CHANGE CONTROL

HP 3000 users have developed a variety of creative procedures for controlling changes in their environments.

A common strategy requires programmers to FCOPY source code from the production location into a development location. This strategy can be implemented at three levels.

1. The development area may be nothing more than a set of individual groups where files reside. In this case, each programmer copies all of the necessary files into his own group where he makes the appropriate changes. There is no standardized testing environment.

2. Separate account structures may be maintained for production and development. Often a separate account structure is maintained for testing as well, to duplicate the production account structure. Thus, each programmer can be confident that testing is conducted in an environment that closely resembles production. Establishing separate account structures on a single computer often produces adequate results.

3. A separate development computer may be used, thus eliminating the possibility of direct access to master files located on the production computer. Although this approach is ideal, it is not possible without a separate development computer.

Unless stringent controls exist, however, this strategy does not guarantee that only one individual can check out a particular file at a time. And, since programmers are not restricted from accessing production accounts, there is no way to audit their access.

Once the development is completed, many shops allow the same programmer to test their own work and simply overlay the original production files with the new version of code. Since such a procedure seldom represents adequate control, it is widely suggested that a formal Quality Assurance (Q/A) process be used.

There are several ways to initiate such a process, depending upon the resources that are available. Smaller companies may have programmers Q/A test their colleagues' development efforts. In these cases, tests are usually performed in the development account.

Larger organizations may dedicate one or more individuals whose sole function is testing. Here, a separate Q/A test account is generally set up to reflect the production account structure. In this case it is vital that the developer is finished, all claims to the code are relinquished. After it has been moved to Q/A for testing, the one copy of the code will exist only in Q/A. If the Q/A analyst locates an error, the code will be returned to the original developer for revision. It is now Q/A's turn to relinquish all claim to the code by moving it back to the development location and purging it from Q/A. If simultaneous copies were to exist in both locations, last-minute changes in development might not be included in the Q/A version. Thus, the final production version would not be accurate. It is surprising how often this obvious safeguard is overlooked.

At the conclusion of the Q/A phase, another step will often exist. A higher level manager will perform a final approval on the development code to certify that all standards have been met and all tests have been satisfactorily completed.

Following final approval, the finished code is ready to be moved into the production location. Since the new files will be copied into production with the same names as the original files, the original files must first be backed up to tape to avoid loss of the earlier version. Next code must be recompiled to assure an exact match between source and object. Lastly, JCL and other files must be updated.

This update process is tedious, time-consuming and error prone, especially when large numbers of files are involved. Nevertheless, precise standards cannot be eliminated at this final stage or the shop will run the risk of serious inconsistencies.

Based on the previous discussion, it is possible to state several general rules for good change control:

First, establish separate accounts for production (master files), development and, if possible, test. Test accounts should be mirror copies of production. This allows files to retain their original FILE.GROUP names as they are moved from test to production and makes it possible to visualize the link between a developing program and its production/master version.

Second, when files are checked out" from the production location to development, a copy should be made. The original source should never be destroyed. However, when development is complete, files should be moved to the test location, thus eliminating synchronization problems with old versions.

Third, when Q/A has approved all changes, a project leader or manager should verify that adequate and accurate test procedures have been followed. Only at this point should code be moved into the production library. If system loads are heavy, such updates could occur in batch and should be initiated by operations or a production Librarian. The timing of the release into production should be coordinated with the user's schedules, with the release of other modules and with documentation changes, etc.

Prior to updating the library, the original production copy should be verified and stored. Without this step it is much more difficult to return to a prior version in the case of a problem. As a general rule, we recommend that the authority to release files to production be restricted.

AUTOMATED CHANGE CONTROL

Although this article has described the minimum requirements necessary to achieve reliable change control, implementing these controls places a considerable burden on everyone involved in the change process . . . unless the process is automated.

Although an automated change control system will require some initial planning, once it is operational, the burden on the MIS staff will actually decrease. The process would be as follows:

The first step in implementing an automated change control system is to define your development environment. Decide what files you want to control, what types of file movements will be performed, what approvals will be required at which stage, and who will be authorized to perform or authorize each type of file movement. This will depend on such things as:

The size of your development staff
The levels of management and job responsibilities in your development group
The size, complexity, and volatility of your applications
The amount of packaged software used by your firm
The physical hardware configuration of your computers
The degree of control which you and your auditors agree is appropriate.

The following examples show four typical development environments, presented in order of increasing complexity.

As you read through them however, keep in mind that these are only intended to be representative examples; each shop is different. A good automated change control system must be configurable to meet the specific needs of the organization.

EXAMPLE 1 - BASIC DEVELOPMENT ENVIRONMENT

The basic development environment is a small shop with a single computer and a staff consisting of two programmers, one day-shift operator, and a "shirt sleeves" manager. The development control objectives are basic, but critical:

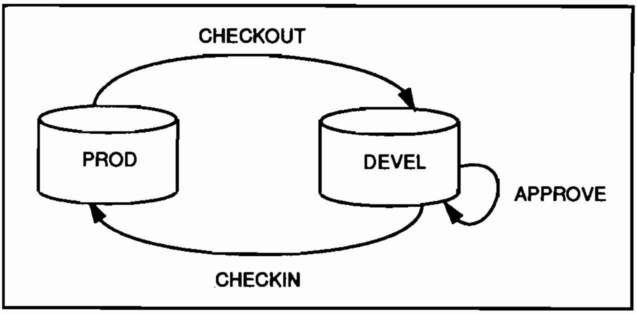
- They need to know where the current production versions of all source, object, and job files are,
- They need to be sure that all changes are made to copies of those files in a separate test location,
- They need to assure that all changes are approved by the manager before they are put into production, and
- They need to do this without further burdening their staff who is already working long hours.

The first step here, as for most installations, is to identify all of the production source, object, and job files. These files need to be grouped together and secured. Through an automated change control system, this can be accomplished by creating customized filesets which represent whole groups of files. This eliminates the need to change the operating system account/group structure or move any of the files around.

The second step is to define the file movement policy, or steps, that will be allowed in the shop. In this example, we have three basic file movement steps:

1. A CHECKOUT procedure, which copies source, object, or job files from the master location into a development location,
2. An APPROVE step, which allows the manager to stamp a changed file with their approval, and
3. A CHECKIN step, which allows the programmer to move a group of files back into the master library when they have been approved. This step also makes an automatic backup copy of the old master file. In addition, the backup copy is compressed to a small fraction of its original size to save disk space.

This process is illustrated below:



To maintain the security of the master library using MPE alone, the CHECKOUT and CHECKIN steps would have to be performed by a manager, an operator or an additional employee. The approval step would have to be documented on paper. With an automated change control system, the CHECKOUT step can be defined to make test copies of the master files even though the master library is secured. The automated system can keep track of these copies and can prevent multiple programmers from checking out copies of the same file at the same time. The APPROVE step can be defined to mark the file(s) as approved, and can be restricted so that only the manager can use it. Furthermore, the CHECKIN step can be defined to allow the programmer to push his changed, approved files from his test environment into the master library without having to log on to the master account, and automatically archive the old versions.

EXAMPLE 2 - SEPARATE Q/A FUNCTION

This example illustrates a medium-sized shop with a tightly controlled development process. The MIS organization consists of a manager, six programmers, two operators, and a full-time quality assurance staff of two. Due to the critical nature of their applications, this shop insists that their Q/A staff perform full unit and system testing on every change. They also require management approval of the testing procedure before any new or changed software can be put into production. They do, however, need to allow their programming staff to make "quick fixes" on an emergency basis, thus bypassing the delays that would normally accompany the Q/A process. It is especially critical that a complete and reliable audit trail be maintained for these quick fixes.

We can define this environment as follows:

1. A CHECKOUT step as described in Example 1. This allows the programmers that only one copy at a time is modified (other concurrent copies can be made with read-only access).

2. A SUBMIT step, to allow programmers to move their modified source, object and job streams into the Q/A area.

3. A GETCURRENT step to bring read-only copies from the master library directly into the Q/A area. This allows the Q/A staff to perform an integrated test using the current production versions of programs that are not currently being changed.

4. Since not all changes will pass Q/A, we need a way to get the files back into the development location. We define a REJECT step which will be performed by the Q/A staff when a program fails testing.

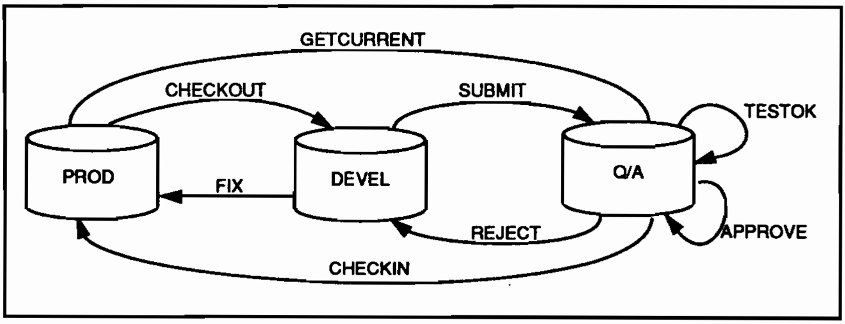
5. Q/A will signify that a change has passed its system tests with a new step called TESTOK. This will help the manager keep track of the status of work on various programs and is a positive indicator that testing is complete.

6. We still need an APPROVE step, but it is defined to operate on files in the Q/A location. The APPROVE step will be performed only after they have been TESTOK'd.

7. The CHECKIN step now moves files directly from Q/A to the master library, once they have been APPROVEd.

8. Finally, we define a FIX step, which moves files directly from the development area to the master library.

This development environment is illustrated below:



Here again, the automated system facilitates file movement, emergency fixes, and management approval without burdening their staff. The CHECKOUT, SUBMIT, REJECT, and CHECKIN steps are defined to operate on specific groups of files, so users do not need to fully qualify file references. Wild-card references may also be used to move groups of files at once. These steps can be defined to automatically purge the files from their original location after copying to eliminate any need for additional housekeeping efforts. Because an audit trail is maintained for each step, no extra effort is required to control use of the emergency FIX capability - the manager gains complete visibility by simply listing all uses of the FIX step.

EXAMPLE 3 - NETWORKING, SOFTWARE DISTRIBUTION, SOURCE/OBJECT SYNCHRONIZATION

The primary difference between this example and the two previous ones is that there are separate computers for production and development. In addition, the company's applications run on multiple remote computers. As in previous examples, the master library contains the production source, object, and job files, but in this case, the object and job files are actually executed on the various remote computers. This adds the important control objective of assuring that all of the remote sites are running the correct versions of the code. To this we will add the audit requirement of source-object synchronization: we must assure that the object files in the master library (and on the remote computers) were in fact generated from the source files in the master library.

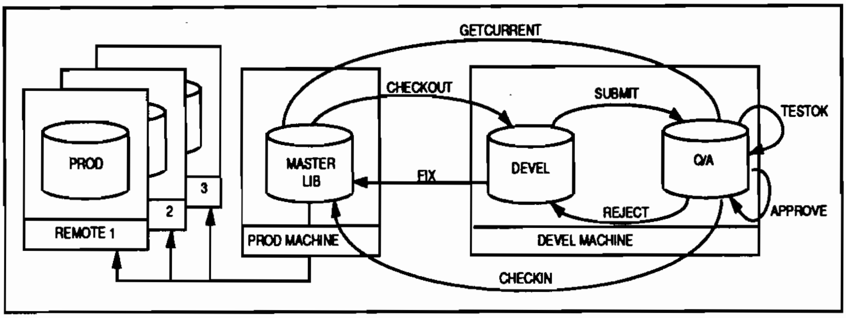
The rules and file movement steps for this environment are the same as for Example #2, with the following exceptions:

1. A new step, called RELEASE, is defined to distribute groups of object and job files to the remote computers. This step will copy the files specified to all of the remote computers, and produce an audit trail to verify that the copies were successfully transferred.

2. The CHECKIN and FIX steps will be modified to move only source and job files into the master library, and to automatically stream compile jobs to generate the object files from the new source. Since the new source has already been compiled to test the changes, this step is redundant, but it is an effective way of assuring that the source and object files always match and are synchronized.

3. While the remainder of the steps function as they did in Example #2, the automated system makes the multi-computer environment transparent to the development staff. Now CHECKOUT, GETCURRENT, CHECKIN, AND FIX all operate between the production and development computers without requiring extra steps or command.

The resulting environment is illustrated below:



EXAMPLE 4 - PACKAGED SOFTWARE, ADVANCED VERSION CONTROL

In this example, the company uses third-party application software and receives periodic releases of the software from the vendor. Their own programmers have also customized portions of the software in-house. They have developed custom reporting and other extensions to the software. Whenever this shop receives a new release from the vendor, the new software is placed into its own separate account. There it is integrated with existing software by the programming staff, tested by the Q/A group and eventually put into production in the same manner as internally developed software.

Since the software is run on a large number of remote computers, the company releases newly approved production software in stages. First, a small number of remote users conduct a beta test for a limited period. When this stage is satisfactorily completed, the software is released to the remainder of the sites. Because of this procedure, the company needs to maintain two or more versions of the software simultaneously, and, in emergency situations, make changes to a prior version without interfering with work performed on the new version.

The rules and file movement steps defined for this environment are the same as Example #3 except for the following:

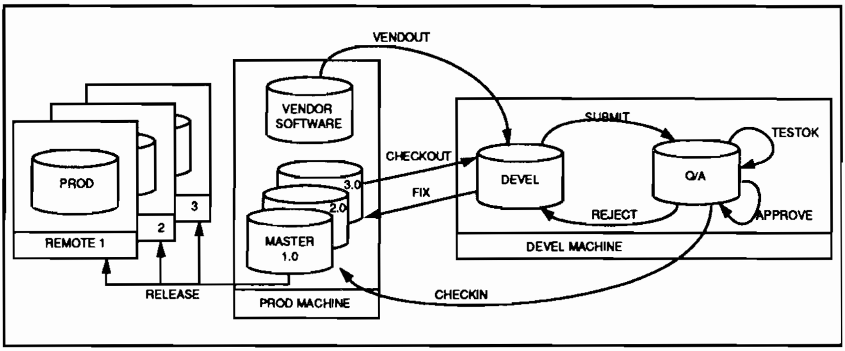
1. A new step, VENDOUT, has been defined for programmers to check out source and job files directly from the vendor software location. The step will prevent the programmer from accidentally getting an old version from a different location.

2. The CHECKOUT step is redefined to select the latest version of the software from the master library. This is accomplished by searching the "new release" location first. If the file is not found there, the "old release" location is searched. This assures that the programmers do not inadvertently use old versions of the software.

3. The CHECKIN and FIX steps are redefined to move files into the "new release" location.

4. The RELEASE step now operates on the version locations much as CHECKOUT does. When specific files are released, the automated system searches for the most recent version, and finds it even if it has not been changed for several versions. If a general release is made, only the files that have changed are distributed. This reduces tape and disc storage requirements as well as network distribution costs by eliminating unnecessary file transfers.

The resulting environment is illustrated below:



It is important to note that this rather complicated development environment can be precisely controlled with minimal effort through the careful choice of predefined file movement steps. Searching through two or more locations for the most current version can be performed automatically by the automated change control system. This enables the shop to maintain the integrity of prior versions while building a new version . . . without replicating files that have not changed. It also assures that programmers always get the most current version of any file.

SUMMARY

HP 3000 data centers have built a substantial asset in their software and data files. Yet, without effective change control, it is virtually impossible to guarantee the integrity of modifications to this valuable asset.

Furthermore, industry surveys show that most firms have a substantial software maintenance backlog. Without efficient means to manage changes, this backlog severely hampers the progress of new development efforts. As a result, it is becoming quite common for HP 3000 shops to establish formal procedures for software change control.

Although many shops initially attempt to implement change control through a manual system, these systems have met with limited success due to the heavy burden they place on the MIS staff and the inherent unreliability of the manual approach. Because automated change control systems provide an effective, reliable solution while reducing the burden on the MIS staff, they have recently become the standard. These automated systems improve development and maintenance efficiency and ensure that all program changes are properly authorized, documented and tracked.

Automated systems provide management with quick access to the status of changes, the change history of a file and the impact of a proposed change. They provide programmers with an environment that allows them to concentrate on programming rather than on searching for the correct versions of files and documenting their usage.

Safeguarding the integrity of the software asset while paving the way for more efficient productive development and maintenance, is one of the most compelling challenges facing HP 3000 professionals today. Automated change control systems can provide the tools to meet this challenge.

CHANGE CONTROL CHECKLIST

Change control procedures for computer programs should be established and followed. The intent of these controls is to prevent unauthorized, inaccurate, and unreliable program changes from being incorporated into the live production environment. Both scheduled and emergency changes must be appropriately controlled to maintain the ongoing integrity of software.

You can use the following techniques to ensure that proper controls are being maintained over your program changes:

- Develop and adhere to formally approved written standards for all program changes
- Define and enforce procedures detailing who can initiate and who can authorize program change requests
- Describe and track the nature and reasons for proposed changes Enforce testing and acceptance procedures for all program changes including emergency changes
- Test all program changes under normal operating conditions
- Involve users in preparing test data and reviewing test results
- Investigate and correct all errors before transferring code to production
- Certify that all test results demonstrate adequate protection from fraud, waste, and misuse of the program
- Document all program changes and update appropriate documentation as changes are made
- Log all completed changes as well as those changes in progress
- Utilize a formal system to report all changes to users and project managers
- Enforce a checkout-checkin procedure that prevents a file from being simultaneously modified by more than one programmer
- Develop procedures to analyze whether other systems are affected by new program modification
- Retain and secure original source code until changes have been processed, tested and updated
- Limit the frequency of program changes, except for emergency cases
- Notify both the user and EDP project manager when emergency changes are made

SUCCESSFUL PROJECT METHODOLOGY

Paper No: 3004

By: Robert S. Dobis

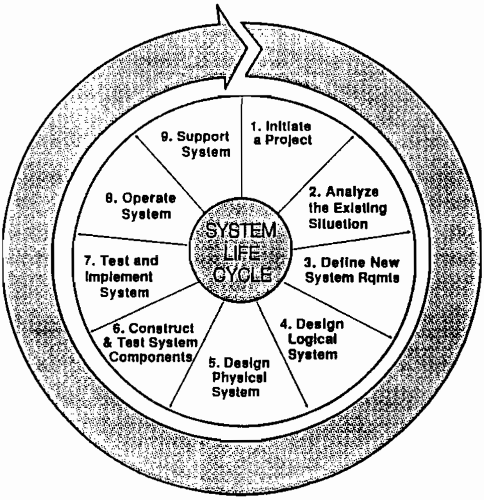
Crowe, Chizek and Company
2100 Market Tower
10 West Market Street
Indianapolis, IN 46204-2976
(317) 632-8989

Introduction

The purpose of this paper is to present a general project methodology which can be used to successfully develop any application software solution. While no two projects are exactly the same, the key to any successful project is the consistent use of a standard methodology. This consistent usage ensures that all critical aspects of the project will be covered, providing the foundation necessary for a project's success. What follows in this paper is an overview of the project methodology we use at Crowe, Chizek and Company. This methodology, called THE Guide (developed by THE Guide Associates, Inc. with permission from Myers & Associates, Tampa, Florida), provides our firm with the foundation we need to successfully complete our data processing consulting projects. During this presentation, I will discuss the purpose of each step of the methodology and attempt to provide some tips and pointers on how these steps best fit a typical project. Finally, as an aid to this entire process, a simple example project, with corresponding documentation, will be provided in the appendix to this paper.

The Project Life Cycle

The basis of our implementation of THE Guide is what we call "The Project Life Cycle". It is this cycle, presented as a wheel on the next page, which provides the general steps which must occur for any data processing project to be successful.



SYSTEM LIFE CYCLE
1. Initiate a Project
2. Analyze the Existing Situation
3. Define New System Rqmts
4. Design Logical System
5. Design Physical System
6. Construct & Test System Components
7. Test and Implement System
8. Operate System
9. Support System

As you can see from the diagram above, all data processing projects are made up of nine basic phases. These phase are: Initiate a Project, Analyze Existing Situation, Define New System Requirements, Design Logical System, Define Physical System, Construct and Test System Components, Test and Implement System, Operate System and Support System. As illustrated in the diagram, these steps form a continuum for the life of any system. At the end of the final phase, all nine steps will begin anew with the initiation of any system enhancements or improvements. It is this continuous system life cycle which is the strength of the methodology. Regardless of project size, all life cycle steps are applicable to all types of projects. Thus the general steps used to initially implement a new system are still used for all enhancements that may be made to the same system during its production life, although to a lesser degree.

Using this system life cycle as our guide, we will now examine in detail each of the nine project methodology steps.

1. Initiate a Project

The purpose of this first step is to gather sufficient information as quickly as possible in order to develop an initial project plan and strategy. While this step may seem intuitively obvious, its importance cannot be overemphasized. This is particularly true for small projects or enhancements. Many of these projects fail simply because they are never considered "projects" in the first place. While these projects typically have a high priority and require quick delivery, the users' needs are rarely clear. The user may think they need a few additional fields on an existing report to solve their problem, but without determining what the users' actual needs are, jumping right into this solution can doom a project from the start. So when initiating a project, it is important to think in terms of user needs, not user solutions.

Because we must clearly define a problem before solving it, user participation is vital to the success of a project. Therefore, the most important activity in developing an initial project plan is the assembling of a clearly defined "project team". This team should be made up of members of both the MIS and user community. Along with improving your odds of project success, this early activity will begin to help your users "buy in" on the fact that solving their problem may take more than a few minutes of technical magic by you or your staff. In addition, it will send a clear message that their participation is not only needed but required, often going a long way in setting their expectations on what must be done (along with how long its going to take).

Once you have assembled your "project team", your initial project strategy is to quickly generate and document an initial problem statement. This problem statement will serve as your guide through out the remainder of the project. When working out this statement with the project team, remember to get to the core of the users' problem, not their thoughts on any potential solution. As the project progresses, your users will have plenty of opportunities to assist with the actual solution.

2. Analyze Existing Situation

As the title of this project step suggests, the purpose of this step is to understand and document (briefly) the users' current situation. Of all the steps in our project methodology, this is probably the most important, and conversely, the most dangerous.

The reason for this conflict is quite simple. On the one hand, it is almost impossible to determine the users' system requirements without properly understanding the current environment which caused the problem in the first place. Often times, especially in smaller projects, the project team has a tendency to jump right to determining the requirements without taking time to discuss how the user community developed the problem. Likewise, large projects (where new or replacement systems are being developed) often skip this phase because the projects team "is building an entirely new system". Since replacement systems typically have 80% of the functionality of the existing system, neglecting to take the time to examine the existing situation can add greatly to the effort of defining the new system requirements.

On the other hand, many project teams will take this project step to the other extreme. In these cases, so much time is spent discussing and documenting the existing situation that valuable project momentum is lost and the project team begins to loose sight of the problem. In addition, the team can become so enamored with the current system that they loose their ability to look beyond their existing boundaries when determining requirements and eventually designing solutions.

The correct balance, therefore, is to learn enough about the current situation to feel comfortable with how the problem relates to its environment. This will allow the team to further elaborate on the problem and, in turn, understand the environment that any solution must operate in With this proper balance in mind, spend the minimum amount of time necessary documenting this current understanding. Many formats can be used for documenting the existing situation, including data flow diagrams, process models and simple written paragraphs. Regardless of the format used, however, only document what is necessary to solve the problem at hand. Remember, your goal is to design and implement a new system, not re-document your existing system.

3. Define New System Requirements

With the project teams' understanding of the problem to be solved and the current operating environment insured, the team can now begin to develop the requirement for the new system. In this step, the project team will develop a written requirements document which clearly defines what functionality the user wants in a new system or enhancement. This document should describe these requirements in full sentences and in a very non-technical language. The intent of this document is to determine the scope of the system as it relates to solving the problem.

The easiest way to begin this phase is to simply take notes on all of the requirements the project team feels are necessary. The team should then break this list down by the functionality required. In the case of simple enhancements, there may only be one function necessary to solve the problem. Once all of the functional areas have been defined, break down all of the requirements into one of the newly created functions. It is important that these requirements describe what the system/enhancement needs to do, not how its going to be accomplished. Remember, this document must be non-technical. Stay away from any data processing terms your user community may not understand. It is very important that this document be at a level that all users of the proposed system can comprehend. In addition to defining the functional requirements, any specific performance requirements should also be documented. Finally, any general constraints that must be worked around should be included.

After the initial requirements document has been drafted, the final activity to be accomplished during this step is the written acceptance of the document by the project team. Since the requirements document serves as the definition of the scope of the project, obtaining approval will successfully define the system you are going to create. Getting approval at this point should also prevent a project's most dangerous threat, "scope creep". As most of us know from experience, nothing can kill a project quicker than the constant addition of new requirements. Prevent this from happening by gaining approval now. In this way, when new requirements are brought up (and they will be), it will be evident to everyone on the project team that these requirement should be part of a new project or, if necessary, will be included in the existing project, but only after taking into account the cost of the additional time, money and effort necessary to add it.

4. Design Logical System

With an approved requirements document in hand, the project team can now begin the task of designing the new system or enhancement. The first step in this process is the creation of the Logical Design. This design, a constraint free, client-acceptable conceptual design for the new system, will provide us the first sketch of our new system.

The intent of this part of the design is to describe and illustrate how the new system or enhancement will meet the requirements generated during the previous phase. As with the requirements document, try to describe the system in non-technical terms. It is important in this phase not to actually write the system (as in program specifications) but to simply describe how the system will work. In construction terms, this phase is roughly equivalent to an architect's sketch, providing a glimpse at what the building will look like, both inside and out. It is not, however, a substitute for the detailed blueprints, which will be generated after approval of the sketch.

Keeping this analogy in mind, the project team should start the design by describing and illustrating the shell of the system. Sample screens and reports necessary to fulfill the functional requirements should be generated. Detailed descriptions of the system processes related to both these samples and the desired functionality should be written. The detail in this document should always address how the system will operate, not how it will be assembled. When describing each functional area, take the time to describe how the user will interact with the system. Again, this level of detail will allow your users to fully understand the new system they are about to receive. Finally, when developing this logical design, always refer back to the requirements document. Your Logical Design, in fact, should contain the exact functional areas described in the requirements document. This direct link to the requirements document will insure that all requirements are addressed and will allow users to more easily follow the Logical Design.

As the final step in this phase, the team should again obtain written acceptance of the Logical Design by the project team. Like the requirements phase, this approval further limits the scope of the project, properly setting users' expectations for the system. This Logical Design approval is also important because of the nature of the remaining phases. Each of the next two phases become more technical and MIS oriented, and it will become more difficult to describe the system to users using outputs from these phases. This directly infers that the Logical Design will be the best document to clearly state to your users how the new system will look and feel.

5. Design Physical System

Based on the outputs of the Logical Design, the project team now has a conceptual design for the new system or enhancement. It is the purpose of this phase, the Physical Design, to take this design and get it to fit into the "Real World". Using our construction analogy, the team must now create the detailed blueprints which match our architect's drawing, taking into account all of the practical constraints of the building (cost, materials available, etc.). This is the point of the project were activities start to become more technical and are akin to traditional system designs steps. MIS participation will begin to increase while user input will begin to decrease. This occurs because, for the most part, the intent of this phase is to describe how the design will be implemented, not how the system will meet requirements (this should have been done in the Logical Design).

With this level of MIS participation in mind, every MIS shop has its own set of standards when it comes to detailed systems design. This is the phase of the project where all of these standards will be used. The format of the physical design is not as important as the components which must be included. While the components of a good Physical Design are listed below, feel free to use the standards your organization already utilizes when creating this document.

The first step in the Physical Design is to confirm the hardware and software foundation to be used in the new system or enhancement. In many cases, this foundation has already been determined and is now simply a constraint which must be taken into account during the design. If the project is for a new system with no established foundation, however, the project team must determine the platform before proceeding further with the design.

Once the hardware and software foundation is in place, the remainder of the detailed design must be completed. This design should include new or modified database or file structures, final screen and report layouts (based on the Logical Design and incorporating any constraints from the hardware/software foundations), detailed application and conversion program specifications (process flows, pseudo code, etc.), and any new or changed job streams or batch processes. As stated above, use all of the standards your organization has already developed when creating these design components.

With the development of detailed design moving along, one last but very important component of the physical design must now begin; the system test plan. The intent of the system test plan is to fully document the set of procedures which will test that the functional, performance and constraining requirements developed in the requirements phase will be successfully met by the system or enhancement. This is the one activity during physical design where the project team should concentrate on getting user participation. The more your users help create the test plan, the more likely the system will meet the stated requirements the first time around.

To begin the system test plan, the project team should first state a test of each requirement documented during the third (requirements) phase of the project. With each basic test stated, develop the detail of each test using the Logical Design as the basis. Create the screen input, for example, based on the screens in the design. Since the Logical Design describes how these screens will function, the test plan must simply show example data and a description of what the system will do with the data (screen edits on the sample data, activity which will occur with data, etc.) This test data should then expand beyond the screens to outputs such as reports and forms. Ideally, the examples used to test screens can now be used to test all of the outputs and additional processing detailed in the Logical Design. This entire system test plan process can be long and very time consuming. For large new systems, this step can almost be considered a separate project. But without a good test plan and the corresponding level of necessary user participation, the odds of a successful implementation decrease significantly.

As with the last two project phases, project team acceptance of the Physical Design should be documented at the completion of this phase. The level of user sign-off will probably depend on the level of system sophistication your user community has. This experience level can be used as a guide for who should approve such items as program specifications. Regardless of user expertise, however, the system test plan must be approved by the users in your project team. Only with an approved test plan will you have the criteria to confidently consider a system or enhancement complete and working properly.

6. Construct and Test System Components

After many meetings, much user participation and endless document changes, the project team is now ready to begin the phase everyone in MIS was waiting for, system construction and testing. It is during this phase that computer programs finally get written, databases built, and the physical design translated into a working test system. In addition to the creation of the automated system or enhancement, it is important that any new manual procedures which have been identified during the design phase be documented.

As with Physical Design, all programming and development standards which your organization has established should be used in this phase for the construction of the system. Along with all corresponding programs, remember to develop the one deliverable all MIS professionals hate to develop most; user and operations documentation. In addition, make sure all programs are thoroughly tested as individual units. While this will be no substitute for the final system test in the next phase, it should help insure that all issues not related to the interrelationship of different units have been corrected. Also remember that construction incorporates setting up the appropriate test system, files and conversion programs necessary for the successful execution of the system test plan in the next phase. Finally, with the completion of this phase, your final products are executable, fully tested application programs and written, fully tested human procedures.

7. Test and Implement System

This is the phase of the project where the activities performed in the earlier phases will really pay off. Where historically this phase has taken as much time as the other preceding phases combined, it should now be a relatively smooth integration of all of the previously tested units. Since the test plan has already been developed and approved, it will simply need to be applied to the tested programs. When performing the test, encourage as much user participation as possible. This will add to the confidence of your users and serve as a good training introduction on the system. If you find something wrong in the test, allow the test to continue to completion and then fix the problem. This will help prevent inadvertently fixing the wrong problem or creating a new one. Also, while testing the software itself, make sure the user documentation and procedures are tested too. It is far easier to correct these now than after implementation.

Once you have successfully completed the system test and corrected the software, documentation and procedures, it is time to have the system formally accepted by the project team. To accomplish this, the system (and the results of the system test) are compared to the original requirements generated in the third (requirements) phase. If the system meets all of the requirements originally agreed upon in the document, the project team and your users have no choice but to accept the system. Remember, if any new requirements occur at this point, they should be automatically deferred until the next release of the software (and the creation of a new project).

With the system successfully accepted, the remaining steps are user training and live system implementation (including any data conversion). With agreement from all parties that the system is acceptable, this should be a relatively easy, straight forward process. After these activities have been successfully completed, the project you started with the initial problem statement is now ended. Take advantage of this time to formally end the project with a final project team meeting. Use this meeting to evaluate how the project progressed, whether it met original estimates for time and cost, and to review individual performances during the project. Finally, enjoy yourself! You have successfully completed this project and you should take pride and enjoyment in your work. And maybe more importantly, you better enjoy it now because the next new project is probably just around the corner.

8./9. Operate the System/Support the System

While the final two phases of the project life cycle are obviously important, they do not relate directly to the project methodology necessary to successfully develop an application software solution. It is included in this paper to complete the System Life Cycle and to point out two important MIS facts:

1. The value of a system is really only measured after it is in use and operating for an extended period of time and

2. No matter how good the system is, on-going support activities are going to be needed.

But as we discussed earlier in this paper, the cyclical nature of the project life cycle is in fact its biggest strength. What these last two phases demonstrate is what all MIS professionals already know, that the next project or enhancement almost always comes out of the operation and support of an existing application.

Conclusion

As we stated in the beginning, the purpose of this paper was to present a general project methodology which can be used to successfully develop any application software solution. As you can see from the methodology's steps and probably already know from experience, project management is much more of an art form than a science. As such, the methodology is intended to be flexible enough to adapt to each organizations own unique environment. Hopefully, with this project methodology as a guide and your experience as the driving force, you will be in a position to improve and refine the process of implementing successful solutions in your organization.

Appendix: Sample Project Document

Project Document File

ABC Gas Processing

ABC Maintenance Inventory to MSA Interface

Define Phase

Problem Statement

ABC corporate in Houston has required the Indianapolis plant to provide an electronic interface of inventory valuation information and detail adjustments which affect the inventory valuation. No automated mechanism currently exists to accomplish this.

Current Environment

Hewlett Packard's Maintenance Management software is utilized at the Indianapolis plant. New data elements have been customized into the Maintenance Management system to store most of the data which Houston requires. Screen edits are in place to validate the critical information.

Scope and Objectives

Crowe Chizek's involvement in this project will be limited to ensuring that the data required by Houston properly flows through the Maintenance Management application to the stock activity level. From this point, ABC will develop the mechanism to retrieve, format, and transmit the information to Houston. ABC will also develop the mechanism to value inventory and transmit the valuation data to Houston on a monthly basis.

In addition, Crowe Chizek will also time stamp the stock activity records and accurately store the quantity on hand after the transactions affecting stock activity are completed.

Requirements for Acceptable Solution

The solution designed and developed to solve the above problem must adhere to the following criteria:

- Require no additional data entry on transaction screens,
- Minimize impact on system performance,
- Make stock activity data reliable,
- All issues must use the work order task number, including unplanned issues,
- Allow selection of valuation activity to be sent by date range,
- Provide an audit trail of the valuation activity transmitted,
- The solution mnst be delivered and acceptance testing completed by October 31, 1991,
- Summarize the valuation activity transmitted. This should include the number of transactions, net change in dollars, dollars received, dollars issued for non-capital work orders, dollars issued for capital work orders, dollars returned from non-capital work orders, and dollars returned from capital work orders,
- When performing inventory valuation, do not include capital parts.

Closed Issues

Determine whether to include time stamp on the stock activity data.
Include the time stamp on stock activity.

Should all Issues go against order tasks and not at the work order level.
Yes, but this is not forced by the system.

Does RETURN use the same Class, Project, and Work Order Number as the issue. This could cause a balancing problem at Houston. Should Order Class not be allowed to be changed on the orders and tasks?
A new field will be added to the Add WO screen that will require a Yes/No reply indicating if the work order is a capital project. This value will roll down to the task level when tasks are added. ABC will add this field to the proper screens, reports, and datasets and perform conversion of existing data.

How will Houston let us change inventory value if we cannot use negative dollar figures? At the time we Change Average Cost, we will not have the cost center, company, property unit, etc. Houston needs to define the transaction to be sent to them in this case. Do they want quantity times average cost or just total adjusted dollars. We need to be allowed to used a signed value.
Houston does not need the missing values. Houston will default these values based upon transaction type. The 02 transaction will be the only one sent. The per unit delta will be sent in the average cost field on the Change Average Cost transaction. The quantity entered field on the transaction being sent to Houston can be populated with the quantity on hand, but do not force the value into quantity entered of stock activity.

The Count transaction has the same problem as the Change Average Cost transaction. Do they want quantity times average cost or just total adjusted dollars? Don't have other field values (listed above).
Same as the Change Average Cost transaction above.

Is the TXN-AVAIL field of the stock activity dataset equal to quantity on hand or quantity available after the transaction is completed?
This is equal to quantity on hand after the transaction is completed.

Is the TXN-AVAIL field expected to be used by Houston?
We will not send this field to Houston. The quantity entered field will be used for all transactions sent.

Is everything ordered in Maintenance Management is an inventory item, does it gets valued, and does the transaction get sent to Houston?
Yes to all three questions.

Review differences between 91 inventory adjustment work order and the Count transactions.
ABC needs to develop consistency with the use of this work order.

All work order tasks must have valid equipment IDs.

Does average cost get properly moved when moving parts from stock to repair and from repair to stock?
Average cost gets recalculated during the move transaction.

Stand-by inventory (which is classified as capital) is sometimes consumed by the work order and sometimes not. The average cost can not be used when a capital part is consumed by a work order.
We will prevent capital inventory from being sent to Houston. ABC will add a new flag to the inventory master to uniquely identify capital parts, perform conversion for this new field, and create a report of issues, returns, and work orders.

What value should be used in the field PT-COMPANY of the transaction to be sent to Houston.
The field PT-COMPANY will contain the same value as COMPANY.

Design Phase

Logical Design

Two alternative designs have been created to address the defined requirements. The first alternative (see document labeled Alternative 1) involves ensuring the required data flows down to the stock activity level. The fields which are not currently flowing down to the stock activity with accuracy include cost center, material account, property unit, project number, Houston's work order number, company, capital project indicator, capital part indicator, purchase order price, quantity on hand after the transaction, and time of transaction.

Once the data is available at the stock activity level, a strip process will be executed which will extract stock activity within a given date range. The strip process will create an ASCII file which can then be used to report from, log what was transmitted to Houston, and convert into the actual EBCDIC file to be transmitted to Houston.

The second alternative (see document labeled Alternative 2) also involves ensuring the required data flows down to the stock activity level, but instead of using the standard Maintenance Management data files to store the data required by Houston, a new data file would be created and used to store the required information. This would allow the standard stock activity file to remain in its "vanilla" form. The remainder of the second alternative would stay the same as first alternative.

The first alternative has been selected as the design to address the defined requirements.

Physical Design

Crowe Chizek will develop the required user exits to populate the following stock activity fields from the indicated source:

STOCK-ACTIVITY Field	Source of Value
COST-CENTER	TASK-MASTER
MATL-ACCT-CODE	ITEM-DATA
PROPERTY-UNIT	TASK-MASTER
PROJECT-NUMBER	TASK-MASTER
WRK-ORDER	TASK-MASTER
COMPANY	TASK-MASTER
CAPITAL-PART-FLG	ITEM-DATA
CAPITAL-PROJECT	TASK-MASTER
PRICE-PO	PURCH-ORDER-DTL
TXN-AVAIL	INV-MASTER
TIME-INV-ACT	Time when transaction occurred

The transactions which user exits will be added to populate the above data are:

ADJUST'ONHAND
MOVE
AUTOBOFILL
ISSUE'EU
ISSUE'ORDER
RECEIVE'INSP
RECEIVE'PO'PART
COUNT
RETURN
FILL'BO
ISSUE'EXCEPT
UNPLND'ISSUE
RECEIVE'PO'ITEM
CHANGE'AVG'COST

Customization

The following customization will be performed by ABC. An updated version of the Maintenance Management application which includes these customizations will be sent to Crowe Chizek upon completion.

___ Add a capital part yes/no flag to Item-Data, Stock-Activity, and Add/Change Part. The field name will be CAPITAL-PART-FLG. This field will be X(2) and will allow only "Y" and "N" as values.

___ Add a capital project yes/no flag to the work order and tasks screens and datasets, history, and stock activity. The field name will be CAPITAL-PROJECT. This field will be X(2) and will allow only "Y" and "N" as values.

___ Add a time stamp to Stock-Activity dataset. The field name will be TIME-INV-ACT. This field will be X(6) and will allow only numeric values in HHMMSS order.

___ Add a key path to the Stock-Activity dataset by DATE-INV-ACT.

ABC will be developing the mechanism to strip the stock activity data, create the required files for transmission, logging, and reporting.

Also, ABC will be developing the mechanism to provide Houston with a transmission of the value of inventory at the beginning of each month.

User Exit/Field Cross-reference

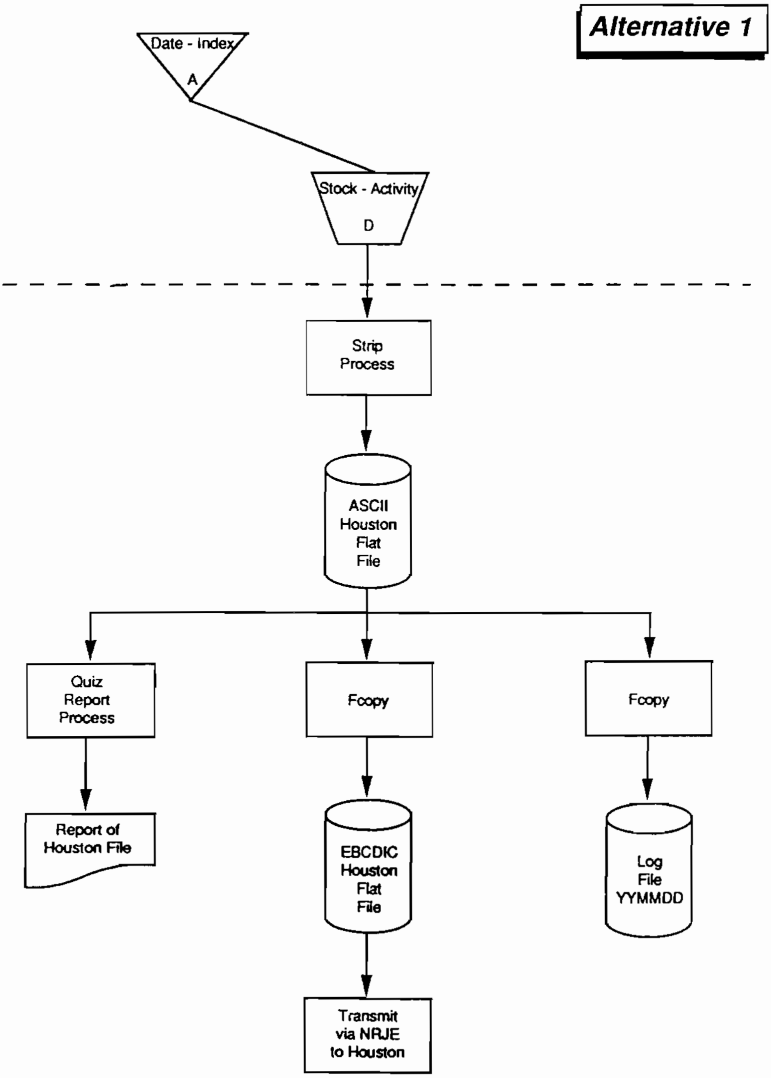
The following cross-reference indicates the fields being populated in the specified user exit and the source of the fields value.

ADJONHD2:	ITEM-DATA.MATL-ACCT-CODE	-> STOCK-ACTIVITY.MATL-ACCT-CODE
	ITEM-DATA.CAPITAL-PART-FLG	-> STOCK-ACTIVITY.CAPITAL-PART-FLG
	INV-MASTER.QTY-ON-HAND	-> STOCK-ACTIVITY.TXN-AVAILABLE
	TIME-OF-DAY	-> STOCK-ACTIVITY.TIME-INV-ACT

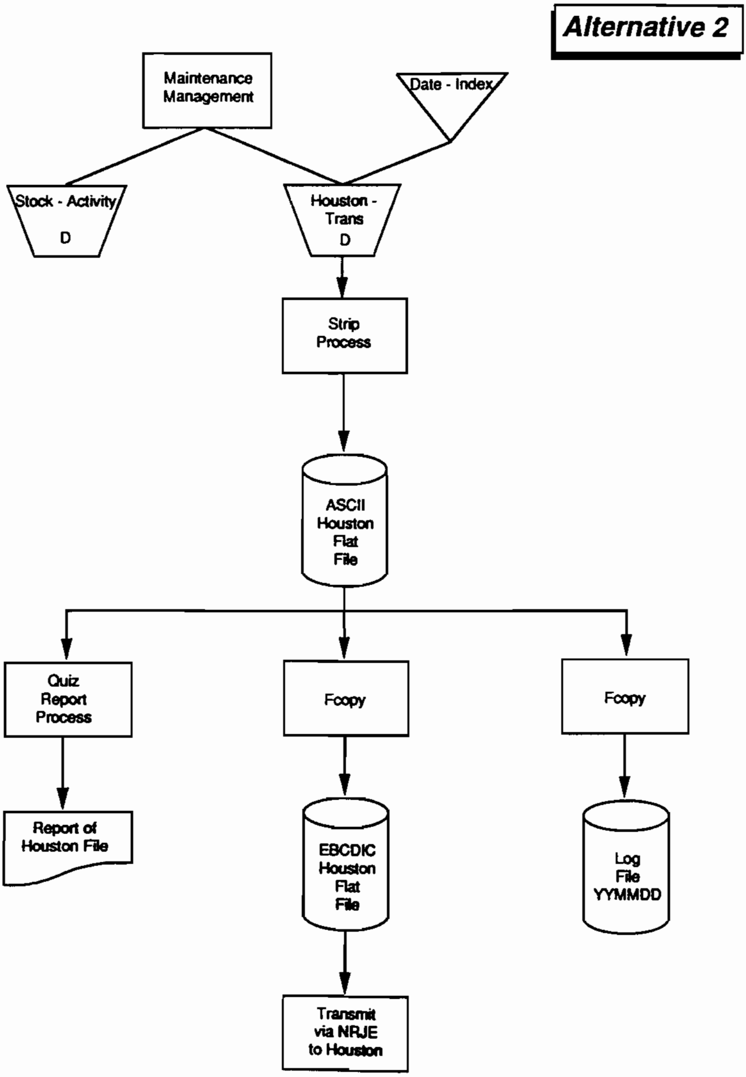
AUBOFIL2:	ITEM-DATA.MATL-ACCT-CODE	-> STOCK-ACTIVITY.MATL-ACCT-CODE
	ITEM-DATA.CAPITAL-PART-FLG	-> STOCK-ACTIVITY.CAPITAL-PART-FLG
	INV-MASTER.QTY-ON-HAND	-> STOCK-ACTIVITY.TXN-AVAILABLE
	TIME-OF-DAY	-> STOCK-ACTIVITY.TIME-INV-ACT
	TASK-MASTER.COST-CENTER	-> STOCK-ACTIVITY.COST-CENTER
	TASK-MASTER.PROPERTY-UNIT	-> STOCK-ACTIVITY.PROPERTY-UNIT
	TASK-MASTER.PROJECT-NUMBER	-> STOCK-ACTIVITY.PROJECT-NUMBER
	TASK-MASTER.WRK-ORDER	-> STOCK-ACTIVITY.WRK-ORDER
	TASK-MASTER.COMPANY	-> STOCK-ACTIVITY.COMPANY
	TASK-MASTER.CAPITAL-PROJECT	-> STOCK-ACTIVITY.CAPITAL-PROJECT
CHGAVGC2:	ITEM-DATA.MATL-ACCT-CODE	-> STOCK-ACTIVITY.MATL-ACCT-CODE
	ITEM-DATA.CAPITAL-PART-FLG	-> STOCK-ACTIVITY.CAPITAL-PART-FLG
	INV-MASTER.QTY-ON-HAND	-> STOCK-ACTIVITY.TXN-AVAILABLE
	TIME-OF-DAY	-> STOCK-ACTIVITY.TIME-INV-ACT
		(2 STOCK ACTIVITIES USED)
COUNT2:	ITEM-DATA.MATL-ACCT-CODE	-> STOCK-ACTIVITY.MATL-ACCT-CODE
	ITEM-DATA.CAPITAL-PART-FLG	-> STOCK-ACTIVITY.CAPITAL-PART-FLG
	INV-MASTER.QTY-ON-HAND	-> STOCK-ACTIVITY.TXN-AVAILABLE
	TIME-OF-DAY	-> STOCK-ACTIVITY.TIME-INV-ACT
		(2 STOCK ACTIVITIES USED)
FILLBO2:	ITEM-DATA.MATL-ACCT-CODE	-> STOCK-ACTIVITY.MATL-ACCT-CODE
	ITEM-DATA.CAPITAL-PART-FLG	-> STOCK-ACTIVITY.CAPITAL-PART-FLG
	INV-MASTER.QTY-ON-HAND	-> STOCK-ACTIVITY.TXN-AVAILABLE
	TIME-OF-DAY	-> STOCK-ACTIVITY.TIME-INV-ACT
	TASK-MASTER.COST-CENTER	-> STOCK-ACTIVITY.COST-CENTER
	TASK-MASTER.PROPERTY-UNIT	-> STOCK-ACTIVITY.PROPERTY-UNIT
	TASK-MASTER.PROJECT-NUMBER	-> STOCK-ACTIVITY.PROJECT-NUMBER
	TASK-MASTER.WRK-ORDER	-> STOCK-ACTIVITY.WRK-ORDER
	TASK-MASTER.COMPANY	-> STOCK-ACTIVITY.COMPANY
	TASK-MASTER.CAPITAL-PROJECT	-> STOCK-ACTIVITY.CAPITAL-PROJECT
ISSUEEU2:	ITEM-DATA.MATL-ACCT-CODE	-> STOCK-ACTIVITY.MATL-ACCT-CODE
	ITEM-DATA.CAPITAL-PART-FLG	-> STOCK-ACTIVITY.CAPITAL-PART-FLG
	INV-MASTER.QTY-ON-HAND	-> STOCK-ACTIVITY.TXN-AVAILABLE
	TIME-OF-DAY	-> STOCK-ACTIVITY.TIME-INV-ACT
ISSUEXC2:	ITEM-DATA.MATL-ACCT-CODE	-> STOCK-ACTIVITY.MATL-ACCT-CODE
	ITEM-DATA.CAPITAL-PART-FLG	-> STOCK-ACTIVITY.CAPITAL-PART-FLG
	INV-MASTER.QTY-ON-HAND	-> STOCK-ACTIVITY.TXN-AVAILABLE
	TIME-OF-DAY	-> STOCK-ACTIVITY.TIME-INV-ACT
	TASK-MASTER.COST-CENTER	-> STOCK-ACTIVITY.COST-CENTER
	TASK-MASTER.PROPERTY-UNIT	-> STOCK-ACTIVITY.PROPERTY-UNIT
	TASK-MASTER.PROJECT-NUMBER	-> STOCK-ACTIVITY.PROJECT-NUMBER
	TASK-MASTER.WRK-ORDER	-> STOCK-ACTIVITY.WRK-ORDER
	TASK-MASTER.COMPANY	-> STOCK-ACTIVITY.COMPANY
	TASK-MASTER.CAPITAL-PROJECT	-> STOCK-ACTIVITY.CAPITAL-PROJECT

ISSUORD2:	ITEM-DATA.MATL-ACCT-CODE	-> STOCK-ACTIVITY.MATL-ACCT-CODE
	ITEM-DATA.CAPITAL-PART-FLG	-> STOCK-ACTIVITY.CAPITAL-PART-FLG
	INV-MASTER.QTY-ON-HAND	-> STOCK-ACTIVITY.TXN-AVAILABLE
	TIME-OF-DAY	-> STOCK-ACTIVITY.TIME-INV-ACT
	TASK-MASTER.COST-CENTER	-> STOCK-ACTIVITY.COST-CENTER
	TASK-MASTER.PROPERTY-UNIT	-> STOCK-ACTIVITY.PROPERTY-UNIT
	TASK-MASTER.PROJECT-NUMBER	-> STOCK-ACTIVITY.PROJECT-NUMBER
	TASK-MASTER.WRK-ORDER	-> STOCK-ACTIVITY.WRK-ORDER
	TASK-MASTER.COMPANY	-> STOCK-ACTIVITY.COMPANY
	TASK-MASTER.CAPITAL-PROJECT	-> STOCK-ACTIVITY.CAPITAL-PROJECT
MOVE2:	ITEM-DATA.MATL-ACCT-CODE	-> STOCK-ACTIVITY.MATL-ACCT-CODE
	ITEM-DATA.CAPITAL-PART-FLG	-> STOCK-ACTIVITY.CAPITAL-PART-FLG
	INV-MASTER.QTY-ON-HAND	-> STOCK-ACTIVITY.TXN-AVAILABLE
	TIME-OF-DAY	-> STOCK-ACTIVITY.TIME-INV-ACT
		(2 STOCK ACTIVITIES USED)
RECINSP2:	ITEM-DATA.MATL-ACCT-CODE	-> STOCK-ACTIVITY.MATL-ACCT-CODE
	ITEM-DATA.CAPITAL-PART-FLG	-> STOCK-ACTIVITY.CAPITAL-PART-FLG
	INV-MASTER.QTY-ON-HAND	-> STOCK-ACTIVITY.TXN-AVAILABLE
	TIME-OF-DAY	-> STOCK-ACTIVITY.TIME-INV-ACT
	PURCH-ORDER-DTL.PRICE-PO	-> STOCK-ACTIVITY.PRICE-PO
		(4 STOCK ACTIVITIES USED)
RECPOIT2:	ITEM-DATA.MATL-ACCT-CODE	-> STOCK-ACTIVITY.MATL-ACCT-CODE
	ITEM-DATA.CAPITAL-PART-FLG	-> STOCK-ACTIVITY.CAPITAL-PART-FLG
	INV-MASTER.QTY-ON-HAND	-> STOCK-ACTIVITY.TXN-AVAILABLE
	TIME-OF-DAY	-> STOCK-ACTIVITY.TIME-INV-ACT
	PURCH-ORDER-DTL.PRICE-PO	-> STOCK-ACTIVITY.PRICE-PO
RECPOPT2:	ITEM-DATA.MATL-ACCT-CODE	-> STOCK-ACTIVITY.MATL-ACCT-CODE
	ITEM-DATA.CAPITAL-PART-FLG	-> STOCK-ACTIVITY.CAPITAL-PART-FLG
	INV-MASTER.QTY-ON-HAND	-> STOCK-ACTIVITY.TXN-AVAILABLE
	TIME-OF-DAY	-> STOCK-ACTIVITY.TIME-INV-ACT
	PURCH-ORDER-DTL.PRICE-PO	-> STOCK-ACTIVITY.PRICE-PO
RETURN2:	ITEM-DATA.MATL-ACCT-CODE	-> STOCK-ACTIVITY.MATL-ACCT-CODE
	ITEM-DATA.CAPITAL-PART-FLG	-> STOCK-ACTIVITY.CAPITAL-PART-FLG
	INV-MASTER.QTY-ON-HAND	-> STOCK-ACTIVITY.TXN-AVAILABLE
	TIME-OF-DAY	-> STOCK-ACTIVITY.TIME-INV-ACT
	TASK-MASTER.COST-CENTER	-> STOCK-ACTIVITY.COST-CENTER
	TASK-MASTER.PROPERTY-UNIT	-> STOCK-ACTIVITY.PROPERTY-UNIT
	TASK-MASTER.PROJECT-NUMBER	-> STOCK-ACTIVITY.PROJECT-NUMBER
	TASK-MASTER.WRK-ORDER	-> STOCK-ACTIVITY.WRK-ORDER
	TASK-MASTER.COMPANY	-> STOCK-ACTIVITY.COMPANY
	TASK-MASTER.CAPITAL-PROJECT	-> STOCK-ACTIVITY.CAPITAL-PROJECT

```
UNPLDIS2:   ITEM-DATA.MATL-ACCT-CODE       -> STOCK-ACTIVITY.MATL-ACCT-CODE
            ITEM-DATA.CAPITAL-PART-FLG     -> STOCK-ACTIVITY.CAPITAL-PART-FLG
            INV-MASTER.QTY-ON-HAND         -> STOCK-ACTIVITY.TXN-AVAILABLE
            TIME-OF-DAY                    -> STOCK-ACTIVITY.TIME-INV-ACT
            TASK-MASTER.COST-CENTER        -> STOCK-ACTIVITY.COST-CENTER
            TASK-MASTER.PROPERTY-UNIT      -> STOCK-ACTIVITY.PROPERTY-UNIT
            TASK-MASTER.PROJECT-NUMBER     -> STOCK-ACTIVITY.PROJECT-NUMBER
            TASK-MASTER.WRK-ORDER          -> STOCK-ACTIVITY.WRK-ORDER
            TASK-MASTER.COMPANY            -> STOCK-ACTIVITY.COMPANY
            TASK-MASTER.CAPITAL-PROJECT    -> STOCK-ACTIVITY.CAPITAL-PROJECT
```



SUCCESSFUL PROJECT METHODOLOGY


Alternative 2
Maintenance Management
Date - Index
Stock - Activity
D
Houston - Trans
D
Strip Process
ASCII Houston Flat File
Quiz Report Process
Fcopy
Fcopy
Report of Houston File
EBCDIC Houston Flat File
Log File YYMMDD
Transmit via NRJE to Houston

Deliver Phase

Files Created

TOOLLIB.COPYLIB - This file contains standard working storage and procedural code used in Crowe Chizek development.

ADJONHD2.EXITS - This COBOL program contains the user exit, ADJONHANDUE2, for the ADJUST ON HAND screen.

AUBOFIL2.EXITS - This COBOL program contains the user exit, AUTOBOFILLUE2, for the AUTOMATIC BACKORDER FILL screen.

CHGAVGC2.EXITS - This COBOL program contains the user exit, CHGAVGCOSTUE2, for the CHANGE AVERAGE COST screen.

COUNT2.EXITS - This COBOL program contains the user exit , COUNTUE2, for the COUNT screen.

ERRMSG2.EXITS - This COBOL program contains the standard error message retrieval routine.

FILLBO2.EXITS - This COBOL program contains the user exit, FILLBOUE2, for the FILL BACKORDER screen.

ISSUEEU2.EXITS - This COBOL program contains the user exit, ISSUEEUUE2, for the ISSUE EXTRA USAGE screen.

ISSUEXC2.EXITS - This COBOL program contains the user exit, ISSUEXCEPTUE2, for the ISSUE EXCEPTION screen.

ISSUORD3.EXITS - This COBOL program contains the user exit, ISSUEORDERUE3, for the ISSUE ORDER screen.

MOVE2.EXITS - This COBOL program contains the user exit, MOVEUE2, for the MOVE screen.

RECINSP2.EXITS - This COBOL program contains the user exit, RECINSPUE2, for the RECEIVE INSPECTION screen.

RECPOIT2.EXITS - This COBOL program contains the user exit, RECPOITEMUE2, for the RECEIVE PO ITEM screen.

RECPOPT3.EXITS - This COBOL program contains the user exit, RECPOPARTUE3, for the RECEIVE PO PART screen.

RETURN2.EXITS - The COBOL program contains the user exit, RETURNUE2, for the RETURN screen.

UNPLDIS2.EXITS - This COBOL program contains the user exit, UNPLNDISSUE2, for the UNPLANNED ISSUE screen.

ERRMSGJ.JCL - This job stream compiles the standard VPLUS error message display routine.

EXITSJ.JCL - This job stream will compile all the user exits in the maintenance account.

MSGCAT.MESSAGE - This file is a compiled catalog of error messages used in the above exits.

MSGCATJ.MESSAGE - The job stream compiles the message catalog source into object.

MSGSRC.MESSAGE - The file contains the message catalog source.

TOOLUSL.USL - This file is contains the compiled user exits. These are then segmented into the SL file in PUB.

Installation Procedures

1) Copy the following files from the test account into the live.

    @LIB@.COPYLIB
    @.EXITS
    @.JCL
    @.MESSAGES
    @.USL

2) Turn on the enable exit flags through CUSTOMIZER.

3) Enable the following exits:

TRANSACTION	USER EXIT
ADJUST ON HAND	ADJONHANDUE2
AUTOBOFILL	AUTOBOFILLUE2
CHANGE AVG COST	CHGAVGCOSTUE2
COUNT	COUNTUE2
FILL BO	FILLBOUE2
ISSUE EU	ISSUEEUUE2
ISSUE ORDER	ISSUORDERUE3
ISSUE EXCEPT	ISSUEXCEPTUE2
MOVE	MOVEUE2
RECEIVE INSP	RECINSPUE2
RECEIVE PO ITEM	RECPOITEMUE2
RECEIVE PO PART	RECPOPARTUE3
RETURN	RETURNUE2
UNPLND ISSUE	UNPLNDISSUE2

4) Prep the dictionary and check the results after completion. Make certain a valid COPYLIB has been generated.

5) Stream EXITSJ.JCL and check the results after completion.

HP Mainframe Management

or Masochism Made Easy

by Guy Smith

Truth in advertising legislation requires that I provide the following disclaimer:

I Don't Know What I'm Talking About!

But then again, neither does any HP3000 systems manager. I should clarify this acerbic statement before my colleagues provide me a blunt instrument lobotomy.

My first site consisted of two systems, a model 42 and a model 48. At this point in modern history (and boy, this might date me) the model 68 was the big kid on the block, and the model 70 was just a gleam in the eye of a Cupertino engineer. Each system had less than two gigabytes of disc space. Weekly backup could be done without second or third shift operators (the fact was we had no operators, so unattended backups were a requirement). The 6250 BPI tape drive was my latest 'toy'. If either of these systems went down, then only a dozen individuals were idled, and they could always find some alternate corporate endeavor to occupy themselves. I suspect that most of us began our respective careers with similar setups.

But now . . . now I face a pair of 980s, a 100 and a 200. A pilot application on one of the systems has over 13 gigabytes devoted to it. This figure will quadruple when the project expands corporate wide. All told we support over 3,000 on-line users. With third shift operators having trouble meeting the five hour batch window, we are investigating banks of tape drives for parallel dumping. And the batch window may get tighter yet. If my main system dies, as many as 1,100 folks may have nothing to do, for they are dependant on the application.

Thus my crack about no large installation HP3000 system manager knowing what he or she is talking about. The sad fact is we cut our eye teeth on minicomputer metal. No facet of small systems administration prepared us for controlling the wiles of mainframe power. Anything I know of the subject was conceived as a side effect of my mistakes, or garnered from those who had a head start with large systems. I have adopted some tactics from my wife who, despite therapy, is a devoted 3090-DOS aficionado, and thus is more than familiar with high capacity hardware.

Your mission, should you be crazy enough to accept it . . .

We have only two true missions. The first, which I'll postpone for a bit, is that of educating programmers on how not to abuse system resources. Our other task is to maintain system throughput. In my demented manner of thought, throughput includes assuring up-time. In short, we are duty bound to keep our end users productive, by keeping our systems up and fast. I'll cover the conventional wisdom of such tactics, as well as document some ways to minimize down time caused by HP system software shortcomings.

Maintaining throughput can be broken down into several basic tasks which include (but are likely not limited to) managing down time (unplanned, but more importantly planned), streamlining operations, assuring application optimization, and controlling relative performance. Each site will rank these categories differently in terms of how they affect throughput. For convenience (mine, not yours) I'll list them in the order they currently plague me:

- Planned downtime
  - Backups
    - Frequency
    - Tools
    - Peripherals
  - Patches and operating systems
    - Patch testing
    - OS/Patch installation routines
    - Required disc space allocations
  - Application patches and modifications
    - User down-time
    - Menus/file equations and versioning

- Streamlining operations
  - Automation of problem capture, alerting and reporting
  - Automation of job scheduling

- Assuring application optimization
  - Problem isolation
  - Tactical analysis
  - Proactive analysis

- Unplanned downtime
  - Crash recovery rules
  - Volume management and recovery

- Controlling relative performance
  - Priorities and what is truly essential
  - Using up extra daytime cycles

Backups

Frequency: I'm easily confounded. My CIO was pondering backups one day and asked if we really *needed* to backup databases every night. My eyes glazed over, my jaw went slack and I might have even mumbled incoherently. It was a simple question with profound consequences.

A look at our batch schedule indicated that a full three hours was devoted to backing up production databases and associated files. Over time we had staggered these backups so batch for one applications could proceed while backup for another started. With a five hour window approaching recovering these three lost hours would be a boon. Our Sunday batch cycle was more lenient (seven hours). Since all databases were being logged, and since other data files were small and could be backed up quickly (or even to another node), we moved to a weekly backup of production databases. The justification was almost too simple: The time spent in rolling forward all production databases, time the number of full recoveries we committed in the past three years was far less than the unproductive time we spent in batch with nightly backups. Open, shut, and done.

The point of this saga is that some basic questions concerning your operations must be asked. Why do you backup up every night? Must you back up everything in your list (we discovered that some program files that had not changed in two years were being backed up nightly)? Can backups be staggered to facilitate batch flow? And as a lead in to my next topic, what are the alternatives to your current software/hardware backup tools.

Backup tools: I recall, with a respectful degree of dread, my second shop. We used Three Kays to edit and assemble a massive proposal for our ever thrifty government (this was long before the days of Desk Top Publishing and other modern miracles). Though management managed to pull together a hefty CPU (again for that particular era) and enough disc space to hold the acquired wisdom of a number of NASA engineers, they couldn't scare up a 6250 tape drive. I instead managed to wrangle several 1600 reelers and managed to backup to all of them at once (it wasn't pretty, but it got the job done).

Today we can have a single STORE command direct it's output to multiple tape drive, compress the data as it goes and even write it to media aside from open reels . . . all while the users are still accessing the data. I am often surprised at shops that have invested millions in CPU power but skimp on tools that improve their operations. Thus the primary challenge of this section is to evaluate you backup needs, and select the proper tool to do the job.

To my alleged mind, the amount of batch window and the criticality of the data dictates what tools you need. There is no substitute for time, and no way to compress it, though I do manage to get in occasional 25 hour days. Thus when the window is tight, you must optimize the tightest bottleneck, which is typically the tape drive itself. I was initially excited about optical technology, with it's massive storage potential. However, the I/O speeds for these devices are abysmal and unsuited for a high volume site. DAT is likewise, but now that there is hardware compression in the drive they are competitive with 6250 open reels. Sadly, none of these options supersedes the 7980XC open reel compressing drive.

Thus, if you are unwilling to pay the hefty charges for the top-of-the-line tape drives, then you must backup to multiple drives and must invest in more exotic backup software. Several vendors offer utilities that reduce backup time by allowing files to be stored while users access them. The tradeoff for this capability is some consumption of CPU and temporary disc space. These tools are most viable to shops that must backup a great deal of data, but have little other batch processing to do.

Peripherals: Aside from open reels, optical disc and DAT tape drive seem appropriate for large scale shops. Each of these are slower than the top-of-the-line open reel drives, but they can be ganged up for parallel backups. For each the media cost is much less than open reel tape, and access time for an individual file is faster during restore operations.

There is a point of diminishing returns when parallel backups occur. Since this is a relatively new procedure, little information exists. However, I am told by those who have experimented with such systems, that six storage devices is about the limit a system can serve simultaneously. If this is a given, then you have a mathematical basis for determining if multiple storage devices will significantly outperform high capacity open reels, or even multiple open reels (a rare form of Nirvana).

The points to my aimless rambling are these: First, if you haven't investigated alternative backup devices, then do so before reading any further. Opportunities abound, and some might just suit you. The second point is that the type of software/hardware backup scheme you employ depends on your batch window. If your shop is like mine (seven days a week, three to five hour window) then the strategy is fast backups. If you run hard Monday through Friday, but have a good twelve hour window on the other two days, then perhaps slower, but denser devices are best (especially if your third shift could be eliminated by stacking up DAT drives and scheduling the STORE job).

Patches and Updates

I've made the public assertion that MPE/iX is no easier to manage than MPE/V. There are certain operations which are simplified, but others are more complicated. Nowhere is this subject more prickly than patches. Given the HP tools for installing patches, their lack of a versioning system in libraries and other phenomenon which I will describe forthwith, patching can be viewed as a major source of downtime.

The first convention that must be observed is to have a separate development system, a *crash and burn* node if you will. Many shops, witnessing the ever expanding power of XL decided, wrongly, to keep programmers on, or merge programmers onto a production system. Due to the design of HP patch utilities, this is not practical since you must drop the system to apply a patch, back out of a bad patch, or even drop a subsystem (NS, SNA) to simply run the patch utility. Every change you make directly impacts production, and thus is undesirable. A separate node, no matter how small, allows you to install patches off line, let them bake and be tested via normal daily activity before committing them to production. Additionally, the SLT created during a patch can often be copied to a production node thus eliminating an entire AUTOPAT/PATINSTL session.

The same theories apply to OS updates. I recall the year the 980 became available. As a matter of procedure, I keep my systems at least one version of MPE/iX behind, graciously allowing other Three Kay shops to debug newer releases. However, the 980s were our only alternative to buying another 960s and my bosses wrongly decided we should plow ahead (you can't teach these IBM types anything). The 980s required version 2.2 of MPE/XL and thus we were the second shop on the face of God's gray earth to receive this release.

Needless to say we faced two months of solid problems and production outages. Thus we have three lessons rolled into one:

✓ **Stay a revision behind in MPE/iX**

✓ **If you must install a new version, have a crash-and-burn node to test on**

✓ **Have a crash-and-burn node anyway**

With MPE/iX we have one unique problem in the installation of patches. UPDATE requires that 60,000 sectors of *contiguous* disc space be present on LDEV one. Often a system will have enough disc space, but free space will be scattered. There is one solution once you have the 60,000 sectors squared away. Simply edit your SYSSTART.PUB.SYS file's UPDATE section to submit a job which builds a file to reserve this space. The 60,000 sectors is always freed up after an UPDATE. Thus if you build a file to hog that space after every update, you will never have to rustle up the balance.

Operations Streamlining

I managed to scare a squad of programmers into action one day, and by telling the truth. We have a third party tool which watches for and reacts to aborted batch jobs. This gaggle of programmers were not employing the tool as was intended and thus were causing some degree of confusion with our operators. During a weekly staff meeting, I told them a truth: that this tool could be trained to automatically dial the beeper of the designated support person when a job aborted, and that I was considering activating this feature. Needless to say these programmers began to employ this package with greater care.

This story serves to illustrate two points: First, there are tools which can aid in streamlining the way any phase of the operation cycle. Second, it takes involving the self(ish) interest of the rest of your organization to get streamlining in effect.

My experience, albeit limited in scope, shows that the two primary areas of operations streamlining are (1) automation of problem identification and capture and (2) eliminating *hands-on* manipulation, or operator intervention. The former is important because undetected problems typically compound themselves and thus take more manpower to resolve than a problem caught in its early phases. The latter is vital since all human dabbling (especially from the under-educated and poorly-motivated) is a source of problems. Anyone buying drinks can get a load of good war stories from me on both of these subjects.

In theory, jobs can be created such that they will trap all possible errors and report them in such a way as to alert programmers or late shift operators. However, I prefer third party tools over programmer diligence any day. The Three Kay market place has a number of tools which watch for aborted jobs and perform all manner of tricks. The one we employ can be configured to logically decide on a course of action based on how or at what point the job aborted. One critical job which we perform weekly will cause an automatic submission of a recovery routine if the process aborts.

The points I am driving at are (1) it is essential that aborted job be caught the moment they occur and that (2) you not depend on JCL to handle all aborts.

Another aspect to insuring up-time is the automation of the job cycle. There was a time when the third shift operators had to manually launch each job, monitor its success or failure, and proceed with the next one. I hear tell that IBM shops still do. This method is still in vogue for a few HP shops who value the assumed attentiveness of operators over the quality of their own code. I have taken a different tact. We attempt at all instances to eliminate operator intervention. In those instances where we must have operator support, we combine tasks together so a lack of attention does not impeded the flow of nightly batch.

A point which I have not dwelled upon is the timing and consistency of operator interfaces. Since operators must be involved at some point, it is wise to maintain a consistent interface (i.e. messages, warnings errors and replies) and to force interaction at logical times. Every shop has experienced an operator who managed to ignore pending console requests, left a STORE job running without changing tapes, or other such disaster. Though human intelligence (or lack thereof) is the primary culprit in these dramas, the operator interface may contribute. It is important to attempt to cluster operator interactions chronologically. Some shops will design batch runs so that all operator input is performed at the start of the cycle. Others prefer to have several cycle check points. However, the object is the same: to have operator interaction with the system occur in buckets.

Unplanned Down Time

It sounds strange to plan for unplanned downtime, but the concept is valid. Any form of downtime can be controlled to some extent, including those elements which you have no direct control over.

The key word in that last sentence is *direct*. Unplanned downtime (i.e. crashes, runaway applications, etc.) are side effects of other events. A system crash is typically caused by a bug in the operating system, or unauthorized experimentation by your system programmers. Runaway applications are either bugs or misuse of an application. In each case, downtime could be controlled both proactively and to a lesser degree, reactively.

From the systems standpoint, unplanned downtime management consists of preventing the system from crashing, and minimizing the time it takes to recover. Having supervised the migration of an exotic site to the MPE/iX architecture, I have become well versed in the art of crash recovery, and more importantly what tactics are available in minimizing recovery time.

The best first step in minimizing crash related downtime is to minimize crashing. This sounds simplistic, but I know of shops that still risk life, limb and up-time by rushing headlong into the newest operating systems, experimental hardware and barely tested system code.

For brevity sake, I'll list some of the steps you can take to minimize system failures:

> **Stay behind:** There is danger in every new release of the operating system. As a rule, we try to stay one revision behind (updating to a revision just before the next revision is to be released). This allows other shops to debug the OS for us.
>
> **Avoid dump tapes:** We maintain an ASCII file on all nodes which list what system aborts have occurred, and on what system. When a system crashes we consult this list and make a dump tape *only if the same crash has occurred on the same system!* Our experience shows that many crashes occur once, never repeating. With systems capable of holding gigabytes of transient disc space (and thus dumping all this data to tape)

dump times have increased dramatically.

Limit dumping: When a dump tape is called for, most system manager dump the entire system including core and transient data. Often you need only the core and perhaps the transient data from LDEV one to ascertain what caused the system abort. Review the DUMP command and see if backing up all that transient data is really worth your while.

Test patches: This should go without saying, but the production system is not the place to install a patch. Patches have side effects and often these effects can be caught on a development system first (we installed a patch once which caused a system abort during boot - and never suffered production downtime thanks to our paranoia).

System applications: I let system programmers wax their surfboards in their cubes. I don't let them dump code onto production systems. These folks by their nature and occupation program right down to the solder joints. Anything they develop has the potential to crash a system. Thus you need to have in place controls which prevent untested or poorly tested code from entering the production environment.

Runaway applications (those dead looping in either application or system code) are more difficult to predict and prevent. I have encountered cases where a son process was tight looping in system code and could not be killed. In these situations, the best action is to change the priority of the process (ALTPROC) to the lowest priority (255) and wait for a convenient time to reboot the system.

No discussion of MPE/iX and downtime would be complete without bringing up the art and science of user volume set management. For the uninitiated, all pools of disc drives on iX systems are user volume sets. Your systems comes with one default user volume set, namely the MPEXL_SYSTEM_VOLUME_SET. You may add as many other volume sets as you like.

The important point to note about user volume sets is that they are independent and operate almost exclusively from one another (except for the MPEXL_SYSTEM_VOLUME_SET which everything else depends on). Thus if a drive on one volume set dies, then the other volume sets remain operative. If applications are segregated in isolated volume sets, then entire applications can remain functional while discs for other applications are under repair.

There are other benefits to aggressive volume management, like higher throughput and reduced likelihood of transaction manager choking. Leave it to say that if you have an MPE/iX system with more than four discs, and are not employing volume management techniques, then you are missing your one best opportunity for up-time.

Relative Performance

I never know if I should cringe or chuckle when a user complains of slow response time. Each user employs the host differently and thus has a different perception of what quick response time is.

The point of this section is that performance is relative. Your objective should be to maximize response time for expensive activities, usually people. If a user must wait ten seconds between prompts, and if you have 200 users, and each earns $30,000 a year and performs 720 transactions a day (1/4 the working day at ten seconds a transaction) than the slow ten second response time cost your firm about *two and a half million dollars a year*. Now if these users are delayed because some report job is running in the wrong queue. . . well, you get the point. Maximize the utilization of your expensive resources.

There are two facets to this maximization of processing. The first entails establishing processing priorities that make sense. The other entails using extra resources wisely.

Processing priorities: Tuning any MPE-V, XL, iX systems is a blend of art and science. Where many artist go astray is when they attempt to mimic other artist. So it is with TUNEing and priority queue assignments. Every shop and every node is different. What works for your peer won't work for you. What works for your production systems won't work for you development node. Thus you must view and control each system as individually as you do people.

Many shops have few controls or conventions over queues. Most hosts do not employ the EQ at all. This is where most misuse of potential performance occurs. Granted, my shop is different due to our application, but here is the way I view processing competition:

CQ: Time sensitive and mission critical sessions and jobs.
DQ: All end users and jobs that must complete on schedule.
EQ: The pit where all other batch and hoggish users (programmers) go.

The point is that you must insure adequate processing priority to the right elements. If your 800 users are competing in the same queue as the system manager while he/she/it is attempting a problem resolution . . . the problem will take longer to solve. Should the monster month end job (which doesn't need to be ready today) run in the same que as the critical afternoon flash reports, then the flash reports won't seem so flashy. Lay out a plan that make sense in your environment and don't be afraid to enforce the rules.

Extra resources: People look at me as if I were insane when I tell them there is nothing wrong with running month end processing during the day. Some folks have become so paranoid about effecting on-line performance that they refuse to attempt any batch processing during weekday hours. T'is silly.

Examine your Trend or LaserRX charts (if you haven't gotten to the point of doing your own performance trending, then you have some work ahead of you before this tomb will be of use). Viewing daytime processing you will likely see the CPU line peak and fall. At times it will approach 100% CPU utilization, while at other times (lunch per chance) it falls to near nothing. These valleys can be valuable batch processing pools if your evening batch window is small.

The only lesson in daytime batch processing is the management of the queues. Ideally there should be an FQ, which is used for this purpose. The FQ should process as the bottom part of the EQ and use CPU only when excess is available. Since no such queue exists, you may have to let these jobs compete freely with regular batch, or ALTPROC;TREE them to a fixed priority of 255.

As a side note, let me advise that you don't bother with queue enhancement tools on large hosts. Products which allow you to define additional pseudo queues are fine . . . on small systems with a small number of processes. However, on large systems with 1,000+ active processes, the CPU used by these tools will outstrip CPU saved and the throughput gained without said tools.

Application Optimization

I don't have the largest installation in the world. As a scope for comparison, here are some basic statistics about our production IMAGE databases;

516	Datasets (more actually since some are duplicated across nodes)
5,700,000	Entries
1,400,000	Largest entry count in one dataset

The point I am driving at is that database optimization may become the most important aspect of application optimization. A miscalculation of optimal capacity for hashing in an automatic master for a large dataset may result in MIPS being translated into Minutes In-between Prompts.

In this day of XL architecture, the old rules and tools for database maintenance must be reevaluated and possibly replaced. In my tenure as the system manager for several large scale systems, I've replaced my database management tool, refined certain assumptions about how to estimate capacities, and have automated certain performance enhancements.

It's time to start over again. If you've entered the 98x or 99x series of processor, you must prepare for a new way of doing business. Here are some highlights about Image in the 90s;

> **Capacities and downtime:** This year my batch window shrinks to five hours. Naturally any database capacity change must occur quickly. But more importantly, they must occur less frequently. Track your entry counts over time and project them into the future. We attempt to perform no more than two capacity changes for a dataset in a year (one to get it right and one for application change induced needs). Several tools are available which monitor and graph dataset usage.
>
> **Capacities and performance:** A prime number is not enough! We religiously used prime numbers when setting capacities for masters, and over sized masters when expanding datasets. However, our efficiency reports still showed migrating secondaries and inefficient pointers (if you don't use Robelle's HowMessy or some such tool, then you are shooting

in the dark). Tools are available that can examine a range of capacities and report which one will be most effective in terms of ratios of secondaries and their distribution.

Dataset repacking: We don't. Or I should say we haven't until recently. Our datasets were of sufficient size that detail dataset repacking could not be performed in our old seven hour batch window with traditional tools. Only of late have vendors provided Native Mode tools that took advantage of the peculiarities of the XL operating system (mapped files, XL sort, etc). However, this task should be done, and if possible, automated. The questions you must answer up front are (1) how much time per batch cycle can I afford for packing, (2) is it worth automating or just monitoring and (3) are the tools I own sufficient for the process.

As you can see by this laundry list, management by exception is not acceptable. Management by proactive evaluation is. This statement leads me to my next point concerning application optimization and that is your best defence against bad code is a good offence.

Programmers are allegedly humans, and thus have human frailties. Some are smarter, some more educated and other more experienced. Of the three variables listed (smarts, experience and education), you can only control the last. Thus as system managers it behooves you to extend education on all matters to programmers.

A case in point: The sort routines on XL require a substantial amount of transient disc space (more than twice the size of the data being sorted). One of our systems began dying whenever a certain job was streamed, which was several times a day. We use Super Tool and Super Tool used HPSORT. Super Tool had to assume that the number of records it needed to extract and sort were equal to the number of entries in the dataset. Thus when the job ran, Super Tool and HPSORT conspired to make two copies of a 800,000 record dataset. The results were that disc space went critical and certain system functions failed.

Once discovered, the problem was reported to all programmers. However, this emergency education was conducted at our regular weekly meeting where we presented our "hint-of-the-week". The point is that education is ongoing and as a system manager for a large shop you must make it part of their daily routine. Again, it is your first and best proactive application optimization technique available.

Problem isolation: A system with high up-time requires speedy isolation and correction of problems. Corrections are often up to HP and we have little control over their progress. However, we can insolate the problem.

By isolate I infer two things: first we can identify the problem rapidly and second, we can put the problem in a quarantined condition. Run away programs can have their priorities reduced until a shutdown is possible. Users can be locked out of problem code. Programmer stunts can be controlled on production systems. Subsystems can be disabled in a pinch. Strive for containment when proactive measures fail.

Closing Thoughts

We live in interesting times (and anyone versed in ancient Chinese curses will recognize what trouble were in). We have the mantle of greatness thrust upon us by necessity. Long gone are the days where we could walk by each user's desk and ask them personally to log off while the system was *quickly* rebooted. Downtime cost are geometrically proportional to the number of users logged on . . and with MPE/Ix systems now capable of supporting several thousand users, and with recovery times soaring, the costs are mounting.

The bad news is that we are on our own. We were taught to manage mice, not elephants. Our instincts are to use a club where a broom once worked. Thus we need to change our mind sets. Keep these goals in mind:

1) Prevent the problems through testing, education and isolation

2) Production is the rule and development is the exception

3) Rethink everything . . . the old rules don't apply

4) Expect the worst and pray for the best

PAPER NO.: 3006

TITLE: HP Quotes & Contracts 101

AUTHOR: Michael Hornsby
Beechglen Development
5576 Glenway Ave.
Cincinnati, OH 45238
(513) 922-0509

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

Paper No: 3007

DESIGNING AN EDI AUDIT AND CONTROL SYSTEM

Kurt Nuehring
Outdoor Technologies Group
One Berkley Drive
Spirit Lake, IA 51360
712-336-1520

INTRODUCTION

Electronic Data Interchange (EDI) is fast becoming a key component in today's business environment. It is one of the critical elements in a Quick Response strategy that aims at getting the right product to the right customer at the right time for the right price. EDI allows businesses, referred to in the EDI world as trading partners, to exchange business information with one another in a fast and reliable way via computer hardware and software, communications firmware, and Value Added Networks (VANS). This information often comes in the form of business documents such as purchase orders, invoices, advance shipping notices, and purchase order change requests to name only a few transactions. Because these documents are critical to the success of a business, it is important that a business be cognitive of any failures in the EDI life-cycle and be able to recover from such failures and go on. To ensure your business this capability, reliable controls must be established to provide proper audit trails and exception reporting in the event of a failure. In this paper we will explore what is involved in designing an EDI audit and control system. More specifically, we will examine:

I. The EDI controls available for our use within the ANSI X12 standard.

II. How to successfully implement and integrate these controls in your EDI business system.

Included in this discussion will be some of the legal issues involved with EDI. A basic understanding of EDI and related terminology is assumed.

I. EDI CONTROLS AVAILABLE FOR OUR USE

The first area we will examine is the EDI controls available for our use. As I stated in the introduction, this paper is based on the ANSI X12 standard. The controls we discuss will be those controls available within the ANSI X12 standard.

I have divided these controls into three areas:

A. The ANSI X12 Document Structure
B. The Functional Acknowledgement
C. Value Added Network Controls

A. THE ANSI X12 DOCUMENT STRUCTURE

The ANSI X12 standards provide a structure to separate individual documents from groups of documents. The purpose of these groupings is to convey control information and provide logical grouping of the data segments. The three groupings we will examine are the Transaction Set level (ST/SE), the Group Segment level (GS/GE), and the Interchange level (ISA/IEA).

Transaction Set Level

The EDI ANSI X12 standards book defines a transaction set as "the smallest meaningful set of information exchanged between trading partners. The transaction set shall consist of a transaction set header segment, one or more data segments in a specified order, and a transaction set trailer segment".

The control records used at the transaction set level are the transaction set header (ST) and Transaction Set Trailer (SE). For each transaction there will be a matching transaction set header and trailer. There are three key control points at the transaction set level.

i. The type of data in the transaction set envelope. This may a purchase order, invoice, etc.

ii. The matching ST and SE control numbers. If the control numbers do not match in the ST and SE segments, then data is missing. You do not have a complete transmission. Your translator should catch this error.

iii. The segment count in the transaction set trailer. Each segment in the transaction set will count one including both the header and trailer segments. If the actual segment count does not match the segment count in the SE, data is missing within the transaction set envelope. Your translator may or may not catch this. If your translator does not check the segment count, you may wish to design this check in your audit and control system.

TRANSACTION SET ENVELOPE

ST Transaction Set Header
BIG
DTM
N1
PO1
CTT
SE Transaction Set Trailer

Functional Group Level

The next hierarchical level in the X12 document structure is the Functional Group. It is similar to the transaction set envelope in purpose and key control points. The functional group envelope is made up of at least one transaction set and may include several transaction set envelopes.

What is the purpose of the functional group envelope? Suppose you are ACME Manufacturing Company. Within ACME are several divisions. Division X makes automobile breaks and division Y manufactures auto seats. If your trading partner is GM, they may send a purchase order to division X and one to division Y. The functional group would direct each purchase order to the correct division.

The key control points in the functional group envelope include:

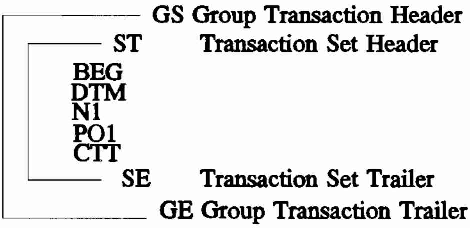
i. The sender and intended receiver of the data. These addresses may be denoted by the respective duns number or telephone number.

ii. The EDI standard, version, and release. In our discussion the standard is ANSI X12. The version/ release will depend upon what your trading partner and you have agreed upon. An example of the version/ release would be 2002, 2040, 3010, or some other supported version.

iii. Matching group control numbers in the transaction group header (GS), and the transaction set trailer (GE). Like the transaction set envelope, if the group control numbers do not match your data is not complete. The group control number may be useful in catching duplicate data from entering your business system. If there is a gap between your last GS control number and your current control number with one of your trading partners, this may indicate you have not received all of the transmissions intended for you. Of course it is your trading partner's responsibility to monitor the functional acknowledgements returned by you to make certain you have received all of their intended transmissions. From experience I can tell you this is not always the case.

iv. The transaction set count in the functional group trailer. If the actual number of transaction sets do not match the transaction set count in GE, you are missing transaction sets and your transmission is incomplete.

ACME Manufacturing Company



GS Group Transaction Header
ST Transaction Set Header
BEG
DTM
N1
PO1
CTT
SE Transaction Set Trailer
GE Group Transaction Trailer

Interchange Envelope

The interchange envelope is the outer most envelope that encompasses one complete transmission from your trading partner. The interchange envelope consists of two separate data segments, The Interchange Start ANSI (ISA) and the Interchange End ANSI (IEA). The purpose of the ISA segment is to "start and identify an interchange of one or more functional groups and interchange - related control segments". With each ISA there must be a matching IEA.

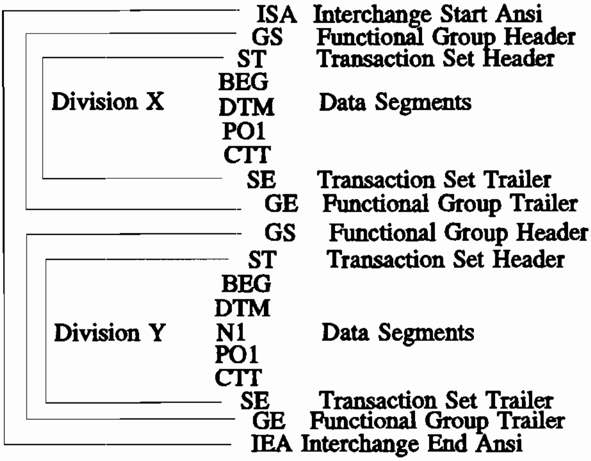
The key control points in the Interchange Envelope are:

i. The sender/ receiver of the interchange. In the case of ACME, the ISA would be directed to ACME.

ii. Matching ISA/ IEA control numbers.

iii. The functional group count in the IEA segment.

ACME MANUFACTURING COMPANY



ISA Interchange Start Ansi
GS Functional Group Header
ST Transaction Set Header
Division X
BEG
DTM Data Segments
PO1
CTT
SE Transaction Set Trailer
GE Functional Group Trailer
GS Functional Group Header
ST Transaction Set Header
Division Y
BEG
DTM
N1 Data Segments
PO1
CTT
SE Transaction Set Trailer
GE Functional Group Trailer
IEA Interchange End Ansi

B. THE FUNCTIONAL ACKNOWLEDGEMENT

We have just finished looking at the ANSI X12 document structure and the controls available. Our next focus will be on the Functional Acknowledgement document (FA) provided for in the ANSI X12 standard.

The purpose of the Functional Acknowledgement is to acknowledge to your trading partner that you have received the EDI document he has sent you and if you were able to translate that document, that is if it met the ANSI X12 standard and version. The Functional Acknowledgement is considered a legal confirmation of having received the document. It is not a legal acknowledgement stating you agree to the terms.

There are three different states the FA can acknowledge. It can state you have accepted your trading partner's document with no errors, accepted the document with errors, or rejected it completely.

We will now look at the structure of the Functional Acknowledgement and the purpose of the different data segments within that structure.

FUNCTIONAL ACKNOWLEDGEMENT ENVELOPE

ISA	Interchange Control Header
GS	Function Group Header
ST	Transaction Set Header
AK1	Functional Group Response Header
AK2	Transaction Set Response Header
AK5	Transaction Set Response Trailer
AK9	Functional Group Response Trailer
SE	Transaction Set Trailer
GE	Functional Group Trailer
IEA	Interchange Control Trailer

The Interchange Envelope and Functional Group Envelope serve the same functions discussed in the ANSI X12 Document Structure section. When designing an EDI audit and control system, you will need to be concerned with the AK segments found within the Functional Acknowledgement.

The AK1 segment relates to the functional group level of the document you are acknowledging. There is one AK1 for each functional group acknowledged. The AK1 will tell you the type of document you are acknowledging and the group control number you will need to match with the original document.

The AK2 segment corresponds to the transaction set level. The AK2 will verify the ANSI transaction set you are acknowledging (i.e. 850, 810, 856, etc.) and the transaction set control number from the original document.

The AK5 segment is grouped with the AK2 segment and there is one of each for each transaction set found in the document received. The AK5 will tell both your trading partner and you if the document was accepted unconditionally, accepted with errors, or rejected. There are several different errors that can occur. Examples of the errors flagged might be when the transaction control numbers in the ST and SE segments do not match with the original document received, or the SE segment was missing in the original transaction set.

The AK9 segment corresponds to the functional group level of the original document received. Key control points include the functional group status, accepted unconditionally, accepted with errors, or rejected, and the total transaction set segment count. More specifically, the total number of segments transmitted by your trading partner, the total number received, and the total number of ST segments accepted by your translator.

C. VALUE ADDED NETWORK CONTROLS

The last part of the controls available for our use are the controls provided by your value added network (VAN). These controls may or may not be available from your VAN for your use. We will examine two critical controls hopefully your VAN provides. If you are not sure your VAN provides these controls, you may wish to contact them and ask them if they offer these controls.

The value added network controls should answer these two questions:

1. Did my VAN receive all the data I sent them and post it to the correct trading partner's mailbox, or successfully send it to another VAN?

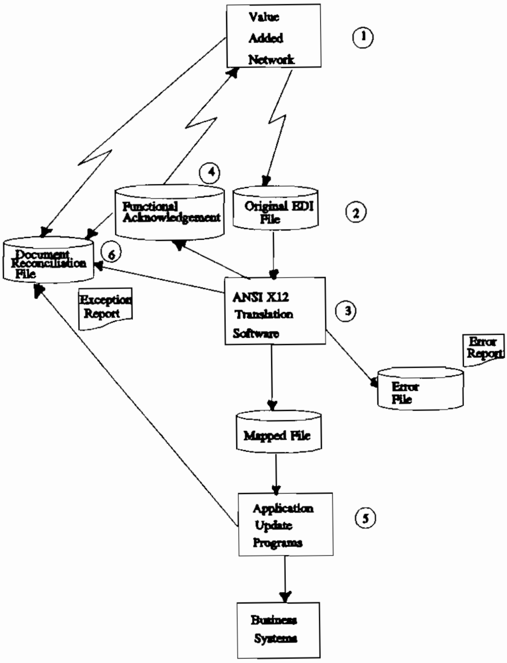
2. Did my trading partner pick up everything I sent him?

The answer to these two questions should come in the form of a formatted report or a data file that you will be able to integrate into your EDI control system. If you are embarking on a Quick Response business strategy, these controls can be invaluable in helping top management grade your trading partner's responsiveness. One question you will need to answer is, after the data you sent was posted in your trading partners mailbox, how long did he wait to pick it up? To effectively implement Quick Response, you need to reduce the number of trading partners from many to a few and then develop close relationships with them. If several days lapse before they pick up their mail, how serious do they take your business? On the other hand, your trading partner may be grading you. Perhaps management would like to keep a handle on how responsive your company is!

II. DESIGNING AN EDI AND AUDIT CONTROL SYSTEM

Now that we have looked at the controls that are available for our use, we now turn our attention to how we can apply those controls to the design of our EDI system. In this section we will look at the life-cycle of both inbound and outbound EDI documents. Upon examining each phase of the EDI life-cycle, we will note the failure points, how to control those failures, and when a failure does occur, how to recover. Please refer to corresponding diagrams for system flow. Each control point is assigned a corresponding reference number.

INBOUND EDI AUDIT AND CONTROL SYSTEM


Value Added Network
1
4
Functional Acknowledgement
Original EDI File
2
Document Reconciliation File
6
Exception Report
ANSI X12 Translation Software
3
Error Report
Error File
Mapped File
Application Update Programs
5
Business Systems

A. AUDIT AND CONTROL SYSTEM FOR INBOUND EDI DOCUMENTS

1. Value Added Networks

When your value added network receives an EDI document from your trading partner, they should make a backup of the data file, process it, and then post it to your mailbox. Hopefully, they will keep a record of each time your trading partner delivers mail to you and you pick it up. If your VAN does keep a record of these transactions, they should be able to provide this information to you in the form of a data file or formatted report. This is one of the key control points your VAN can offer you. From this information you should be able to answer the question, "Did I receive everything that was placed in my mailbox"? If you can receive this information in the form of a data file, you may wish to automate this process. This will be discussed more fully under the topic of document reconciliation.

Another failure point you will need to address, if a transmission failure occurs, what procedures do you and your VAN have in place to enable you to retransmit the lost data? And how long will that process take?

2. Original Data File

Once you have received the complete EDI data file, you are now responsible for it. This is your legal document. You should make a backup of this file. Some auditors state you should keep a copy of the original file for seven years. Others say as long as it takes to complete the full EDI life-cycle. You will want to check with your own auditors for their recommendation. There is one final point I would like to make on this subject. The EDI data file as a legal document has never been tested in a court of law to my knowledge. If you would like to learn more about the legal issues involved with EDI, the American Bar Association has prepared a report entitled, "The Commercial Use Of Electronic Data Interchange", which includes a model trading partner agreement.

3. Translation

You must now translate the original EDI file and map that data into a fixed-length file your application programs can update your business systems with. The translator should take advantage of the EDI controls we outlined in the ANSI X12 document structure. First it will check to make certain the incoming data document(s) meet all ANSI X12 standards. It will also verify the incoming document(s) are from a legitimate trading partner.

Your translator may also be able to warn you of duplicate data by keeping record of controls numbers already processed. If your translator does not provide this function, you can design this into your application update programs.

Your translator may also be able to warn you of duplicate data by keeping record of controls numbers already processed. If your translator does not provide this function, you can design this into your application update programs.

If your translator does find any syntax errors or duplicate control numbers, it should create a file containing the bad data and a report stating the errors that occurred. Several questions need to be addressed. If we have received bad data what procedures must be taken to receive good data? Will your VAN retransmit data at the functional group level or the interchange level? How about your trading partner?

4. **Functional Acknowledgement**
Once the original EDI file has been translated, your translator should also create a functional acknowledgement. The FA should be checked to see if the translated file meets all the ANSI X12 standards. If it was accepted with errors, you may wish to know what those errors are and if it would be prudent to continue with the processing.

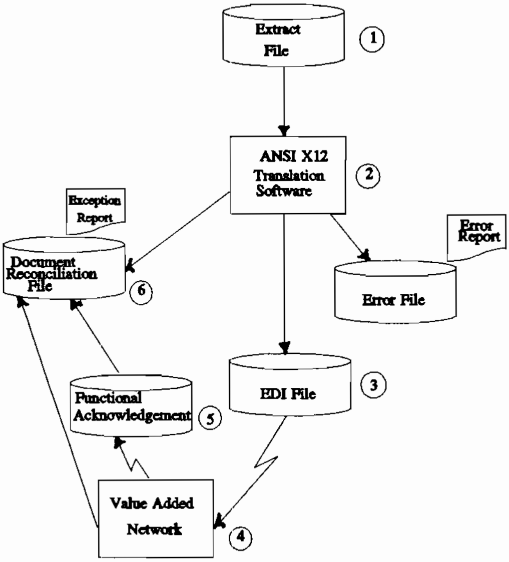
5. **Application Update Programs**
You can design this to be as simple or complex as you like. Your first concern should be the integrity of the data updating your system. Does it pass all of the edit checks used on data entered through conventional methods. If an edit failure occurs here, how do your application programs handle this? Can you easily trace where the failure occurred? Can you easily restart from the failure point? Do you have audit trails installed? What if your trading partner sends you a duplicate PO or some other duplicate document with a unique control number, will your application program catch the error and not allow it to be entered in the system?

6. **Inbound Document Reconciliation**
The purpose of document reconciliation is to verify that all phases in the EDI life-cycle have been completed successfully. If there remain any outstanding steps, then an exception report should be generated and appropriate action taken. I will briefly outline the phases of document reconciliation in the EDI life-cycle.

Phase 1: When translating the inbound EDI file, post the ST control numbers processed for each trading partner.

Phase 2: Using data from the functional acknowledgement, flag the status of each ST segment in your reconciliation file.

Phase 3: Flag the status at the ST level of each transaction that entered your business system.


Extract File
1
ANSI X12 Translation Software
2
Exception Report
Document Reconciliation File
6
Error Report
Error File
EDI File
3
Functional Acknowledgement
5
Value Added Network
4

B. AUDIT AND CONTROL SYSTEM FOR OUTBOUND DOCUMENTS

Now that we have examined the control points for inbound EDI documents, we turn out attention now to the audit and control system for outbound documents.

1. Extract File

The first thing is to back up your extract file. If there is ever a problem in the EDI life-cycle of this document, you will at least have the original file to start again.

2. Translation

The translator will read a mapped fixed-length file and reformat the data according to ANSI X12 standards into a file containing variable-length records. The translator will not check to assure your data is correct. Your translator will generate the ISA, GS, and ST control numbers with sequential numbering. You should use the ST control numbers to post to your outbound reconciliation file.

3. **Translated File**

This is your legal document. Like the inbound EDI file, this file should be stored off-site for seven years. You may wish to discuss the length of time with your auditor.

4. **Value Added Network**

The same control points mentioned in the Inbound section apply here only this time you are the trading partner. Questions that you need to have answered by the network are, did they receive your transmission and were they able to process it? Will they let you know or will you need to wait until your trading partner returns the functional acknowledgement? What if the response time is several days, will you be able to recover? If your VAN does keep a record of all EDI activity you produce, you will want to monitor the activity yourself. If they offer that information in the form of a data file, you can update your reconciliation file with this data.

5. **Functional Acknowledgement**

This is one of the critical control points. Did I receive a functional acknowledgement from my trading partner, and if I did was the data accepted, accepted with errors, or rejected. It is your responsibility to monitor this. If you have not received one from your trading partner, you will want to initiate the contact to find out what happened to the data you sent him.

6. **Outbound Document Reconciliation**

Like inbound document reconciliation, the purpose of outbound reconciliation is to verify that all phases of the EDI life-cycle have been completed successfully. The reconciliation steps are:

Phase 1: Post the ST control numbers processed for each trading partner to the reconciliation file.

Phase 2: Update the reconciliation file with the audit file from your VAN.

Phase 3: Flag the status of each ST transaction from the functional acknowledgement your trading partner has returned to you.

Phase 4: Report any exceptions.

CONCLUSION

In this paper I have attempted to show how to design an EDI audit and control system. We examined the EDI contiols available for our use and how to apply those controls within the EDI life-cycle. If we take careful thought in designing our EDI systems, then we should be confident when disaster strikes - and it will happen, it is only a matter of time - that we will have the necessary procedures in place to recover with a minimum of trouble.

REFERENCES

Electronic Data Interchange X12 Standards, draft version 3 release 1,
Data Interchange Standards Association, Inc., 1990.

Stelzer, John, "Reliable EDI Controls: A Production Imperative",
Sterling Software Exchange '91, May, 1991.

Stelzer, John, "Designing a Reliable EDI Audit and Control System",
Sterling Software Seminar, October, 1991.

PAPER NO.: 3008

TITLE: Computer Management Solutions

AUTHOR
Rod E. Keiser
Computer Management Systems
7208 South Tucson Way
Suite 175
Englewood, CO 80112
(303) 790-0123

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

ARE YOU MY PRIORITY?

JEFF ODOM

BAHLSEN INC.
ONE QUALITY LANE
CARY, NC 27513
(919)677-3227

It is most probable that all of us have participated in a minimum of one time management course by now. It is also highly likely that we are using more than one of those modern "time saving" devices (computers, faxs, cellular phones, etc.). There is also a good chance we actually use our "time saving" devices to do more as opposed to saving time. We could actually say we probably are better educated in time use, do more in the time we have and ultimately, we remain a bundle of frayed nerves. Instead of having the four day work week promised a decade ago, we now are stressed out by the demands of intense multi-tasking.

Often we become unsettled if we do not have dozens of balls in the air simultaneously. We are stressed if we do, stressed if we don't. Where do we turn? Of course, we turn to our training! We change our "bad" behavior into "good" behavior. To do this we remember that they told us true change will not take place before three weeks, we will not change if we do not start within 24 hours of learning how, we must remember this is good for us (no pain, no gain?), know your priorities, everyday study the list of time wasters, absolutely use the miracle tools the trainer so graciously let us in on, and on and on and..........

In reviewing and participating in so called time management courses, I have found everything from outright product pitches to positive attitude promotions. One course I attended covered the following subjects all in a six hour ordeal:

- Personality profile
- Motivating others
- Concentration and alertness levels

- Task juggling
- Time management assessment
- Time management tools
- Delegation
- Procrastination
- Team building
- Positive attitude and words
- Lifestyle
- Job stress
- Assertiveness evaluation and training
- Communication
- Time waster tips

It seems the developer of the course needed some time management training to realize you can not adequately cover and integrate all of the above concepts in so short a time. Granted we need the whole of these concepts if not a few more, to effectively manage our time; but let us please have digestible portions! Further, when it is all said and done, how will I get done what I need done?

What is missing is the engine that drives the other time management techniques and tools. We need help setting priorities. What's important right now! I have even attended courses whose title included words such as managing or setting priorities, yet spent virtually no time on the subject. How could they with all the above listed topics to cover? What was their priority? Maybe it was as simple as duping us into one more time management course.

Actually, we are not putting down time management courses. Implementing the tools and techniques generally taught in them will clearly improve your plight. It was during a particular session whose title connotated help with priority setting but whose subject actually was a little time management tools and a lot of people management that my mind began to wander. How can we characterize and manage priorities? I began an animated daydream that paralleled P.D. Eastman's wonderful children's book Are You My Mother?. If you recall, the story is of a little bird who hatched while the mother was off looking for food. The hatchling wonders where it's mother is and begins searching, meeting other animals and some machinery in the search inquiring "are you my mother?" Ultimately the "Snort" (steam shovel) places the young bird back in the nest just in time for the mother's arrival with dinner.

If we could, let us take a few moments to share this adventure in recognizing the forces setting our priorities. But before we begin, take a moment to write on a piece of paper in priority order the 6 things you would do with your remaining time (including revenue generation) if you found out today that you would live in 100% health but die overnight exactly 6 months from now. What are the things you would want to do? Now set the paper aside and let us ask of the following tasks or personality traits found in others and yourself, are you my priority?

☺ The Bear - Threatening and powerful, successful at intimidating us.

☺ The Snake - Sneaky and cool, winding into our time.

☺ The Otter - Full of life and mischief, never a care for now, we will get to it later.

☺ The Lion - Loyal and powerful, seeing the world in the light of loyalty. Dignity and grace further adds to their influence.

☺ The Rhino - A very territorial beast who defends its space with charging and snorting. Not like a bully this one will defend to the death it's turf.

☺ The Kitten - So sweet and cute to "purrrrfection" ease you away from your sense of direction.

☺ The Rabbit - This one is cute but so jittery to boot. You begin to be attracted out of compassion for a creature such as this.

ARE YOU MY PRIORITY?

☺ The Elephant - The most commanding of all of our animals yet regal and unassuming, content with steering the course.

☺ The Steam Roller - Compacting and crushing all that falls in front of its path.

☺ The Fire Truck - Everything is a crisis, for that's it's job. Sometimes we wonder who's setting those fires?

Are these our priorities? Well yes in part, but the list you made a few moments ago are your true life priorities and all of the other things in your life must first be matched up to this list then set as a priority based on your own time management technique (A, B, C's or I, II, III's, etc.). Use this list as your priority targets. From your target you can establish goals which will help identify your day-do-day and moment-to-moment priorities. Granted, the many influences of people and work tend to draw you this way and that, but keep it all in perspective of your larger whole life priorities and you will stay on track with your goals.

Use the tools that work best for you in maintaining calendars, to do's, delegation and so forth. Remember the difference between important and urgent. Through people skills and assertiveness training, learn to manage the influence on your day-to-day priorities. Evaluate your priorities throughout the day, work in bite size pieces and avoid procrastination. These are all important to managing our time. The key though, is to thoughtfully evaluate what is important in the way you spend your time. Stop and evaluate "Are You My Priority?" based on a sound and healthy knowledge of what your true priorities in life are.

HOW TO TELECOMMUTE AND RETAIN YOUR SANITY

PAUL EDWARDS
PAUL EDWARDS & ASSOCIATES
1506 ESTATES WAY
CARROLLTON TX 75006
(214) 242-6660

HOW TO TELECOMMUTE AND RETAIN YOUR SANITY

ABSTRACT

Telecommuting is becoming a very popular form of employment. There are many factors to be considered to make this new working environment successful. If the telecommuter is properly prepared, then this can be a very efficient situation for the employee and employer. The telecommuter could work part-time or full-time. Extensive requirements definition are a must.

Proper equipment, software, office environment, work attitude, and comprehensive employer policies and procedures are vital for success.

An in depth discussion of these areas will be provided and practical examples of personal experiences will be presented.

Telecommuting is a fast growing form of employment. It can be used effectively by full-time employees of companies or by part-time entrepreneurs. We will discuss the particulars of this work environment in depth. The amount of success enjoyed by the employee is directly related to good planning with realistic goals in mind.

This presentation covers the definition of telecommuting, justification, equipment, software, office environment, work attitude, and employer policies. Most of the material presented is based on many years of experience in this field.

DEFINITION: The word telecommuter is made up of two words, TELE and COMMUTER. Webster defines TELE as far off, distant, remote/at, over, from, or to a distance. And COMMUTER as a person who travels regularly between two points at some distance.

The corporate employee may be telecommuting on a part-time or full-time basis. The determination of type is based on the justification needed.

JUSTIFICATION: Sometimes the talent required for specific tasks is not available locally. This talent could be made available to a company by a telecommuter. A personal situation of an employee, such as pregnancy leave, illness in the family, or handicaps or other special cases of valuable employees can provide the justification needed. The transportation costs or restrictions of certain areas of the country may make telecommuting necessary.

EQUIPMENT: Office machines, computer systems, and furniture must be purchased or provided to properly set up the office environment.

Basic office machine inventory should consist of a telephone, fax machine, and copier. The neighborhood printer or office supply store might provide the fax and copier services. This should only be a temporary solution. You will waste too much time, otherwise. A two line phone switch can be installed to answer the home and business lines from one instrument. A headset is very valuable if much time is spent on the telephone. The basic fax machine should have an automatic paper cutter, 10-page automatic feed, speed dialing, and Group 3 protocol.

The portable or cellular phone adds flexibility and movement to leave the home office without missing calls.
Voice mail or an answering machine can assist with the answering of the telephone and taking messages to prevent an important message from being missed.

A minimum configuration desktop PC should consist of a 2mb main memory, 386SX based CPU with 40mb hard disk storage and 1.44mb floppy disk. VGA monitor prices are reasonable. A laptop PC is a necessity for traveling use. Compatibility with the desktop PC is a requirement. For any remote access, a 2400 baud modem on either PC is a minimum. The price on 9600 baud v.32 modems is falling rapidly.

A minicomputer is very affordable if it is needed. A small HP3000/37 can be purchased for about the price of a PC. If you do your own support, it is a very inexpensive system.

The printer requirements will determine the type and number of printers. They range from inexpensive portable inkjet to dot matrix to laser page printers. A laser printer is the preferred choice for quality, speed, and low noise level.

Furniture can be obtained from family sources, used office furniture stores, computer stores, and large furniture outlets. The basic office should contain a desk with a good back supporting chair. The standard desk may be too high for comfortable access to a PC keyboard. A computer table should be used instead. A thorough analysis of work flow will determine the placement of equipment. If other individuals visit the office, sufficient seating should be available for their use.

A sufficient supply of office materials can be supplied from the supply room of the company. You will need pens, pencils, paper, disks, and any other items used in the office.

A complete set of manuals are needed for any hardware and software system used. Some software packages offer the option of printing manuals from the programs themselves.

Flexible storage must be abundant and easily available.
Shelves can be wall mounted or located in closets. A four drawer file cabinet is needed, also.

SOFTWARE: Basic software considerations are operating systems, word processing, spreadsheet, graphics, data base, communications, language compilers, and electronic mail.
Ensure that any software purchased is widely used and supported by all equipment you have.

The usual operating system for the PC is MSDOS but UNIX is gaining in popularity. If you purchase an HP3000, current FOS and subsystems tapes should be provided. Windows for the PC provides a graphic interface to isolate the user from the operating system. A mouse should be used with Windows to derive the most benefit from the interface.

Word processing software needs are driven by the business focus. The needs will vary from simple letters to programming language text to document processing. The popular products are WordPerfect, MS Word, Wordstar, and PageMaker.

Many people use the Lotus 123 spreadsheet product. There are several compatible products available. Compatibility with other software and hardware is important.

Graphics capability are included in some of the spreadsheet and word processing software. Harvard Graphics provided a very simple way to make the slides for this presentation. It will also make a slide show with your slides for a moving presentation if you use an overhead projector display or a PC projector. High quality graphics are essential to a professional presentation.

There are many data base products available. The best known is DBase IV. Most are relational with menu-driven interfaces. If a fourth generation language is used with the HP 3000, such as Speedware/4GL or Powerhouse, there is the capability of using them with a PC as well as the HP 3000.

Communications products must be chosen based on the systems to be accessed. If access to an HP3000 is required, then Reflection and Business Session are the most popular.

Requirements for programming will determine the language compilers needed. Microsoft has Cobol, Basic, C. Micro Focus has a main frame Cobol. Borland has Pascal and C.

Electronic mail systems vary from simple bulletin boards to extensive products like HP DESK/HP MAIL.

OFFICE ENVIRONMENT: The room used as the office is very important to the telecommuter. Location, power, air conditioning, lighting, and communications are major parts of the successful installation.

A room shared with other home activities will function temporarily but should not be considered permanent. The ideal is a spare room, such as a bedroom or basement is preferred. To limit noise and other outside interference, a room with doors to shut out the rest of the house activity should be used. A room could be added on the house if the funds are available and the existing structure will support it.

Power for all system devices should be wired directly to dedicated outlets with 20 amp breakers. Air conditioning, if required, should be provided to maintain a proper temperature level year round. Lighting should be sufficient to ease eye strain. Communications lines should be available for voice and data communications. Voice/fax/modem switches are available to share lines to cut cost.

WORK ATTITUDE: A successful telecommuter must act as if he is at a formal office. Mood is very important. You need to set work hours that are realistic. You must be a self starter because no one is there to manage your time and work habits. Also, you must resist the temptation to be a workaholic.

You should dress based on the interaction with others and at least have some basic formality, not a bathrobe. You will tend to wear out tennis shoes and blue jeans instead of suits and dress shirts.

You will find a need for social interaction. You don't have the office water fountain to congregate around. This could cause large phone bills to satisfy the urge to socialize.

The support that is given by the family is most important to your success. They have to agree to the fact that you are at the office for specific times during the day even though you have some flexibility. Stay involved with your local users group.

EMPLOYER POLICIES: The employer can save costs by having you telecommute. You need to have a written agreement for expense reimbursements, attendance at meetings, time and progress reporting, capital equipment purchases, and insurance. This agreement provides the formal ties to the employer. It also delineates responsibilities. This is important to the corporate employee who works part-time or full-time.

A corporate employee should have a business plan for their telecommuting operation. This helps set priorities and provides economic justification, if needed.

PAPER NO.: 3011

TITLE: Managing a PowerHouse Shop

AUTHOR: David G. Robinson
PowerSpec International
403 Cross Lake Drive
Fuquay-Varina, NC 27526
(919) 552-8049

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

Paper Number 3012

Automating Computer Operations

Kirk E. Bedwell
Marsh Company
707 East B Street
Belleville, IL 62221
(618) 234-1122

Overview

The main goal of any Management Information Systems (MIS) group has always been to "serve the user." Service has included providing users with the best solution, both in hardware and software, for any given problem. Users now enjoy state of the art order entry, payroll, manufacturing, engineering, general ledger and a host of applications. A payroll run which use to take a few days to process can now be done in a couple of hours. Why the MIS departments even provide bursting and decollating services for their users.

But what about the MIS departments? What goes on behind the closed doors of the computer room? Are there persons in the computer room all hours of the night to ensure that all of the batch jobs are run and that back-ups are being completed? If the answer to this question is yes, then it is time to take a long hard look at the way the computer room is being administered.

This paper will describe a relatively simple way to automate computer operations in a one month time frame at a minimal cost. Highlights of this paper will include both benefits of such automation plus some of the traps which may arise from automating.

What Should Be Automated

In a word, everything relating to the operation of an HP 3000 computer system should be automated. This includes system back-ups, job submission, error detection, report distribution and system performance and availability monitoring. System back-ups need to be automated so that they are performed at the most optimal time. Of course this would be when the least amount of users need access to the computer system. There are several products which allow back-ups to be run to disk and then to alternate media. These same products also allow multiple tape back-ups which means that a second or third tape (reel to reel, DAT, etc.) can be used without having to change media.

Job submission is the area where MIS managers will reap their biggest savings. Many shops currently use some type of user defined command (UDC) to stream jobs on a daily basis. This is fine provided that the system never fails or no job dependencies are needed. For example, if UDC called TUESDAY is created and it has specific times that jobs are to be run, what happens to those jobs if the system was to fail? They would probably be resubmitted one job at a time. And what if the job need to start another job? Yes one job can start another but think about all of the coding that would be needed to start another job., especially if the dependent jobs are different for each day. For example, a manufacturing utility is run every day to load order demand. Upon completion of the utility, the user want REPORT1 run on Monday, REPORT2 on Tuesday, REPORT3 on Thursday and all three reports on Friday. Indeed it is possible to program the job control to accomplish this task but imagine the complexity of the stream file. There would be IF THEN logic throughout. With an automated schedule, an entry for each day is made to stream the appropriate report job.

Error detection is also another way to provide MIS customers with outstanding service. With an automated operations area, errors in job streams are detected immediately and can be relayed to operations personnel via HP Deskmanager, system console, modem or nearly any other way. Another advantage of error detection is that the MIS manager can program the schedule for what types of errors to look for. For example, if there is no !EOJ line found in the job stream, a serious error (REMAINDER OF JOB FLUSHED) has occurred. At that point the system manager could have the schedule stop and send a simple message via modem to a beeper or an individuals home. And think about the time savings. The night operator no longer has to scan a STDLIST to verify that the job ran correctly. The flexibility, power and time savings is reason enough to automate computer operations.

Report distribution. Why do I want to automate report distribution? In a small firm of less than twenty people there probably is no reason to automate report distribution. Everyone knows everyone else and it is relatively easy to figure out which report goes where. But in a larger firm, report distribution can become quite confusing. For example, a firm distributes twenty copies of a sales report to various managers throughout the facility. Unless a single contact point is created to distribute the reports, chaos could arise. With report distribution automation, the names (titles, office numbers, departments, etc) are printed on each copy of the report. Computer operations can simple check off which users get which reports.

System and performance monitoring is critical to a firm. Not only is system performance critical but assigning priorities to various types of jobs is also critical. With automation, MIS managers can determine which jobs execute when and in what system queue (CS,DS or ES). Take the case of Joe Payroll who is constantly running large payroll reports during the day. Joe has been asked to submit the reports as a batch job so that other users are not affected. But Joe will not listen. He wants his report and he wants it now. With automation, the MIS manager can force Joe's job into the D or E que even though it is running interactively. By doing this, other users are not affected by Joe. Conversely, the telephone order entry program needs to have the 3000's full attention any time it is run. With automation, the MIS manager can adjust the priority of the program to ensure that the program remains active a tall times.

System monitoring watches the system for system failures, room temperature and console replies that are pending. Take Mary MIS who comes to work on Monday morning to find the system has failed. Not only does she have to scramble to recover but she also has to explain to the users and her managers why the system is not available for use. When system monitoring is automated, MIS managers receive notification that there is some type of system problem. This way, Mary MIS could have had the problem resolved and possible an available system for users on Monday morning.

How To Get Started

The following is an outline for automating computer operations.

I. Determine Automation Goals

In this step, the MIS manager in cooperation with the operators and programmers, should outline a plan for automating the computer operations. The plan should include target dates and assigned personnel. No one person should be charged with the complete automation process.

II. Begin Implementation Plan

Start the implementation and involve every member of the MIS department. During this step, all hardware and software required to complete the project. Every department member should be aware of what is happening and how this function affects their individual job. Define the roles of each department member as it pertains to that individual.

III. Test The Proposed Solution

After the hardware and software has been installed, test and retest the systems. If automating an operations schedule, run dummy jobs which simply logon and logoff to check job dependencies. Then check the various error conditions but forcing those error into a job.

IV. Go Live

Once testing has been completed, immediately go live. The longer the official start-up date, the less likely the system will be implemented.

V. Monitor The System

Unlike most other systems, an automated environment needs to be tweaked and tuned every so often. Jobs get added, changed or even deleted from the daily schedules. Report distribution are modified, job priorities change and a number of other changes occur.

What Can Go Wrong

When automating performance monitoring, several things can go wrong. It is quite possible to slow down some users that they get virtually no response at all. If this happens, the users will normally call and complain.

On scheduling automation, there may be an error condition that does not get tested. For example, the manufacturing application which posts sales demand may encounter a broken chain or other file system error. If the condition does not flag an error, it may go undetected.

PAPER NO.: 3013

TITLE: Obtaining the Competitive Edge Through Automated Data Collection

AUTHOR: Ray Agrusti
Eagle Consulting & Development Corp.
170 Kinnelon Road
Suite 3
Kinnelon, NJ 07405
(201) 838-5006

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

Paper # 3014
Applying Mainframe Security Standards In the Minicomputer Environment

Lee Courtney
Monterey Software Group
10420 S. De Anza Blvd - Suite C9
Cupertino California
408-253-1778

"The minicomputer is dead! Long live the minicomputer!" is a cry often heard today. With personal computers offering superior price performance and ease-of-use, and mainframes firmly entrenched in many MIS shops, the minicomputer sometimes looks like it is headed in the same direction as the punched card. But despite predictions of the imminent demise of minicomputers, many vendors are continuing to see respectable growth for both proprietary and open systems. For example, last year Hewlett-Packard reported over ten percent growth for its HP 3000 product line, with the installed base surpassing 50,000 systems.

Minicomputers exhibit features found in both PC and mainframe environments. From the PC environment minis share ease-of-use features and approach the PC level of price-performance. At the high end minis share mainframe traits such as the ability to function as an organization's information resource processing and storage utility. A good example of the minicomputer environment today is the HP 3000 and its MPE operating system. While technical specifics may not be applicable, many of the general concepts concerning the HP 3000 are germane to other platforms, be they a DEC VAX or an IBM AS/400. Thus in this paper it is assumed that the HP 3000 can effectively represent the minicomputer spectrum and it will be used to illustrate general security characteristics of minicomputers.

While minicomputers approach mainframes in terms of system throughput, the number of users supported, and storage capacities, they do not commonly possess the same sophisticated administrative and security facilities. What are the security and audit problems facing minicomputer sites? To answer this question it is helpful to look at the evolution of the HP 3000 product line, and examine the security capabilities of the MPE operating system.

The HP 3000 was introduced in the early 1970s as the first interactive multi-user business minicomputer offering COBOL, a bundled database management system called IMAGE, and a easy to use terminal forms package called V/PLUS. At that time most HP 3000s were stand-alone systems running a single application for a homogenous user community. The security strategy was one of controlling access based on the concept of an "insider" verses an "outsider". An insider was a person within the organization with permission to use the system, while an outsider did not have permission. The only means of enforcing this control was logon passwords.

The next phase of HP 3000 maturation began in the early 1980s with the introduction of the first system capable of supporting over 200 on-line active users. By this time networking was becoming standard and included both peer-to-peer as well as HP 3000-to-mainframe connections. Users now had the CPU power and features to greatly expand where and how the HP 3000 was used in their organization. It was not unusual to find Fortune 500 companies using the HP 3000 in a distributed processing environment with extensive networking, while medium and small businesses used the HP 3000 as their central information utility. Applications running on the HP 3000 began to include electronic mail, accounting systems, database management applications, document preparation, and program development. Thus this period was characterized by (1) both networked and stand-alone HP 3000s, and (2) a variety of usually centralized applications.

Today the HP 3000 product line has evolved from that first stand-alone minicomputer to a family of systems capable of supporting up to 1000 users. In the late 1980s HP introduced the first commercial system based on a Reduced Instruction Set Computing (RISC) architecture. Almost all HP 3000 systems are integrated into some type of network (including networks of PCs as well as peers and mainframes) and run a diverse set of applications. For many businesses they perform the same role as mainframes may have performed previously, but with improved price-performance and ease-of-use.

Concurrent increases in the power of the HP 3000 and in the integration of computer processing into daily business operations has made computer processed data the most valuable asset of an organization. However, the security functions offered by the MPE operating system have not grown commensurately with the value of the information stored on those systems. In 1992 the primary control used to secure HP 3000 systems is still a password at logon time.

How can security, access control, and audit concerns be addressed in a manner that will preserve HP 3000 ease of use, compatibility, and price-performance? Mainframe systems are characterized by a history of well developed commercial security standards and awareness, along with IBM and third-party security products. Thus one can examine solutions successfully used in the IBM MVS and VM environments, and apply those concepts which would be beneficial.

Which standards in the mainframe environment might be utilized on the HP 3000? In general these include individual accountability, auditability, and separation of duties. Individual accountability provides the capability to tie an event on the system, such as a logon or file open, to a specific individual. Closely tied to individual accountability is the concept of auditability. Auditability insures that the system collects the information necessary to accurately determine who is using the system and what they are doing.

For example, an auditable system would record who unsuccessfully attempted to open the payroll database. Separation of duties can be used to implement checks and balances on the system and thus acts as a deterrent to abuse.

In addition to the above general concepts, some implementation specific characteristics are also essential to insuring a mainframe level of security on the HP 3000. First is the concept that security and audit controls should be integrated at the operating system level rather than at the application level. Second, security and audit controls must apply to all classes of users on the system. Increased security should not require changes to applications and should be compatible with operating system and network interfaces. In other words increased security should not be disruptive to users, applications, the network, or production work.

How can desirable mainframe standards be applied on the HP 3000 today, and what additional capabilities are required to secure an HP 3000? Mainframe security falls into three areas: system access control, file access control, and audit. System access control is concerned with the identification and authentication (I&A) of users when they first establish contact with the system. This includes both interactive access through terminals and batch access through jobs, along with access through network protocols. File and database access controls are concerned with controlling access to both programs and data files by authorized users. Detection of unauthorized access attempts, and verification of authorized access is controlled by the audit function.

"Of Course We're Secure, All My Users Have Passwords!" - System Access Controls

The first security mechanism users encounter regulate access to the system through interactive sessions or batch jobs. Like mainframes, the primary system access control on the HP 3000 is the password. However, while MPE provides passwords at the user, account, and group levels, it does not provide any password composition or administration controls.

To extend the capabilities of the rudimentary password protection provided by MPE, some security implementations use a Logon User Defined Command (Logon-UDC) to implement I&A controls. While this method may have been adequate in the past, its use today is a severe control deficiency. The problem with Logon-UDCs is that authentication takes place after a user has logged on the system, not before. This would not be a problem except that the Logon-UDC mechanism may be disabled by accidental or malicious action. If this happens, there are no system level access controls in place at all! Logon-UDCs are also incompatible with network file transfers since they prompt for passwords differently than MPE. While not security related another concern is the impact Logon-UDCs have on logon performance. On systems where Logon-UDCs are utilized the I&A sequence can take up to 20 minutes to complete.

In the mainframe environment one finds the same identification and authentication checks, but more comprehensive and efficient password and administrative mechanisms. These include extensive password controls to prevent unauthorized access as well as features to control access by legitimate users. Password controls in mainframe environments greatly increase the usefulness of passwords. These controls integrate I&A, password aging, reuse and composition controls at the operating system level and enable user password administration.

Integration of password checks at the operating system level is one of the most important aspects of mainframe environments. Users encounter the password verification process before gaining access to the system. In addition, when implemented at the system level, rather than at the application level through Logon-UDC schemes, the performance and resource impact of the I&A process is greatly reduced.

In order to increase the effectiveness of a password system, many mainframes implement extensive password controls. Password aging and reuse controls force users to periodically make use of 'fresh' passwords to reduce password exposure. Along with password aging, mainframe security products also allow a security administrator to setup standards for password composition. The ability to setup such standards should be an integral part of any HP 3000 security program.

While password controls prevent access by unauthorized users, in many mainframe environments we find mechanisms which are also intended to further enhance access controls for authorized users. These controls include both user and device logon restrictions. User controls define the how and when of access to the system. Device controls are used to specify where a user is allowed access to the system. Typically these controls are realized using a general access rule language. Such controls can be an effective deterrent against insider abuse by restricting a user to only permissible times, days, and terminals.

System access can also be gained through batch jobs. While batch environments are different between the HP 3000 and mainframes, the use of some general principals can increase job security. The first is to use facilities that eliminate the need for embedded passwords in batch jobs. Embedded passwords greatly increase the probability of password exposure and make secure job sharing difficult, if not impossible. Second, when batch jobs are shared between users, users should make use of facilities which completely eliminate password sharing and prompting. Use of access rules not only provides a secure job submission and sharing facilities, but also increases the productivity and flexibility of batch job streams.

The purpose of system level access controls on mainframe systems is to prevent unauthorized access and to control access by authorized individuals. The goal for HP 3000 system managers and security administrators should be the same. System management should make sure that they are taking advantage of technology which provides baseline password composition and administration controls, and is tightly integrated and compatible with the operating system and networking protocols.

"My Users Don't Know Enough To Do Any Harm" - Program and Data File Access Controls

One of greatest security risks on the HP 3000 today is a deficiency of effective and easy to use file access controls. These access controls are the mechanisms governing which files and databases authorized users may access once admitted to the system. Given that the greatest potential for computer abuse is from users having permission to access a system, file and database access controls are a critical requirement. Studies consistently show that the vast majority of computer crime and security related events are perpetrated by users who are authorized to access the system. Given this fact, implementation and use of controls which enforce individual accountability should be mandatory.

Until recently the absence of effective file level access control facilities, especially at the operating system level, has been a major security deficiency on the HP 3000. In the mainframe environment we find several products which define baseline standards in file access control. The two most widely used being RACF(from IBM Corporation) and ACF2(from Computer Associates). These products provide a mechanism called access rules which can be used to clearly define file access and sharing in multi-user environments. Access rules define the how, when, where, and who of file and database access.

On the HP 3000 if file access controls are implemented at all, it is common that they are embedded within an application. However, when implemented at the application level, controls are limited to those programs that have been explicitly modified. Moreover, putting application programmers in the security implementation business violates well accepted standards concerning separation of duties. And even with these difficult implementations, the resulting application controls give only the appearance, rather than the substance, of security. Using system level file access controls removes these problems by providing a comprehensive set of controls which apply to all users, programs, and data without the need for application changes. The controls even apply to access by the operating system itself, thereby controlling all access to sensitive data and applications.

At the system level MPE provides several mechanisms to control file access, but these mechanisms are difficult to use and maintain, contradict one another (there are 3 separate ways to share a file under MPE), or cannot be used to enforce individual accountability. The most widely used sharing mechanisms are the RELEASE command and the access matrix. But neither of these mechanisms is capable of enforcing individual accountability since they cannot allow a file to be shared based on a logon-id. The RELEASE command allows all users unrestricted access to the file until the owner of the file explicitly invokes the SECURE command to retract such access. The access matrix allows sharing at the group, account, or system level, but not by individual logon-id. MPE Access Control Descriptors(ACDs) allow flat files and devices to be shared based on an individual user's logon-id, but cannot be used to control IMAGE databases and are limited to the file level rather than the file, group, and account levels.

Third-party access rules can be used to obtain the same degree of control as available on mainframe systems. Access rules allow flat file, IMAGE database, device, or other objects to be shared in a secure and auditable manner. Implemented at the operating system level, access rules do not require any application changes. As on mainframe systems sharing can be based on a user's logon-id, access type, source of access, time, or date.

In addition to providing increased security, file access controls permit greater flexibility in the type of applications and users supported by the system. Effective controls permit the introduction of applications which may have previously presented an unacceptable risk. Utilization of mainframe standards and techniques allows introduction or expanded use of applications such as electronic mail, customer service programs, bulletin boards, and Electronic Data Interchange (EDI). Thus the use of mainframe access controls not only increases system security, but also system flexibility and productivity.

"What I Don't Know Can't Hurt Me" - Audit Data Collection, Reduction, and Analysis

Perhaps the weakest area of control on the HP 3000 is verification of authorized system access and detection of unauthorized access attempts. This weakness is due to a lack of tools and audit data.

While application auditing is widely practiced on the HP 3000, some administrators may not be aware of the need for auditing at the system level. The primary system audit function performed on the HP 3000 is the configuration audit. A configuration audit specifies which system settings may lead to security exposures, but does not report actual system activity. On the other hand, an activity audit allows review of events which occurred on the system. Configuration audits report what might happen, while activity reports describe what has occurred.

Although activity reporting may be performed on the HP 3000 using system log files, they have several shortcomings which limit their usefulness. Often the data collected by system log files is not comprehensive. The MPE utility used to display log files allows selection of audit events by logon-id, but does not provide reporting by program or data file name. This approach is not efficient since it provides user rather than file based reporting. While reporting by user is important, the ability to examine program and data access is even more critical. Of course the ideal system will contain both functions.

The mainframe environment offers several important characteristics which are applicable on the HP 3000. When we look at mainframe environments we find not only configuration audits, but also tools and data to monitor system activity from a security and control perspective. Most significant is the integration of audit data collection at the operating system level. This insures that all security related events are captured for all classes of users. As a result system management and application development staff are subject to audit review, not just end-users.

One of the shortfalls of having comprehensive data collection, is that reports often contain more data than can be easily comprehended. To increase audit productivity, data reduction and analysis tools must be used. Audit data reduction and analysis tools extract only those audit events fitting user defined selection criteria. These machine based filters greatly reduce the audit workload, increase productivity, and raise the confidence level in audit results.

The audit function is a critical element of any security program. It is the methodology by which management confirms appropriate system use. In the last year tools have become available which provide mainframe type system audit functions with minicomputer ease of use. These tools allow a user to examine the state of the system and look at system activity. Using these tools to implement mainframe audit standards and techniques, users of the HP 3000 can insure the accuracy, integrity, and availability of system and information resources.

Conclusion

What should we look for in a secure minicomputer environment today? Given the roles minicomputers are playing in many organizations, it is important to use those techniques and methodologies which have proven successful and are applicable. Solutions which provide standards concerning individual accountability, auditability, and allow separation of duties should be implemented. To be effective, security tools must be integrated at the system rather than the application level, must provide comprehensive controls which apply to all user populations, must protect against inside abuse as well as outside attacks, and must furnish a complete audit trail of system use and configuration. Given the current state of HP 3000 security, tools which implement mainframe standards and features represent a quantum leap in information asset protection. With these elements in place, one is well on the way to assuring a secure and reliable minicomputer environment which can withstand the tests of contemporary security threats.

PAPER NO.: 3015

TITLE: Negotiating Win-Win Agreements

AUTHOR: Robert Andrews
Dynamic Information Systems Corporation
5733 Central Ave
Boulder, CO 80301
(303) 444-6960

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

Paper No: 3016

Mainframe Downsizing As "Rightsizing"

Author: David J. Rubinstein
Vice President
Innovative Information Systems, Inc.
63 Nahatan Street
Norwood, MA 02062

(617) 769-7511

INTRODUCTION.

With technology constantly on the move and changing at such a rapid pace, it is easy for anyone to become confused over the future direction of Corporate IS. It seams like each day we have a new 'standards' committee, or a 'strategic' relationship between this vendor and that vendor. Although all of these developments appear confusing, the end result will be of a great benefit to the user community.

Due to the rapid increase in computer performance and the evolution of standards (isn't better to have 5 standards rather than 100 standards) business now has an opportunity to achieve two important goals. These goals will help to reduce and contain costs, as well as implement an information technology architecture which will allow the business to grow as it needs to and have the appropriate support from its information technology infrastructure.

In this paper, I will address the justification behind 'downsizing' of hardware. Perhaps downsizing is not an appropriate term, and that rightsizing is. The meaning behind these 'buzzwords' is to migrate from the older mainframe architectures into an environment which will provide a stronger foundation and growthpath into the future. The added benefits of this migration is cost reduction (20% to 60%), cost containment, increased functionality, and many other intangibles.

The objective of the presentation is to describe:

1. Why downsizing is feasible, practical, and a reality.
2. Provide an overview of the direction of technology. 3. Describe a few real life experiences and the lessons learned.
4. Describe an approach to downsizing.

THE EVOLUTION OF TECHNOLOGY.

In order to best understand where we are today, it is best to look at our routes. This is important because a great deal of today's mainframe architecture is based upon hardware and software philosophies created in the late 1950's and 1960's.

The one of the first major business application to be automated was a payroll system for General Electric. This was the introduction to the era of large centralized processing. The 1960's and 1970's gave us large batch processing machines. The emphasis was on the automation of repetitive tasks, processing huge amounts of data. Program development was done on punch cards. A great deal of floor space and environmental controls became necessary in order to keep the machine alive. Today, that same processing power can be found on our desk, even on an HP Palmtop.

In the late 1970's and in the 1980's we saw the emergence of midrange computers, allowing businesses to place computing capabilities at the department level. The development tools began to mature, and by the early 80's, we now had minicomputers that required no environmental needs. The computers came out of the fish bowl (aka: computer room with lots of windows) and entered the office place.

The first half of the 1980's witnessed personal computing emerge, furthering the model of decentralized computer processing. Personal computers could even act as terminals to a host and eventually swap data back and forth. Computers could begin to talk to each other however not many of them spoke the same language. (I refer to this as the Tower of Babel affect).

Through the mid 1980's to the present, data communications has begun to tie all the 'decentralized' pieces together. The 'islands of information' are being bridged together. Computers were beginning to understand each other.

This leads us to the present, the 1990's. The 1990's will usher in the era of 'enterprise' wide computing. With the increase in power and the decrease in cost, computers will be available to users at all levels within an organization. Not only will computers be available, but the data and the application software will be available.

The common thread in the evolution of the computer has been the increase in the power of technology and a significant decrease in its cost. Combining these two benefits with the emergence of industry-wide standards, and the user can now implement an IT infrastructure that can truly support and play an important role in the growth of the organization. And who is the primary beneficiary of this technology evolution...the user.

WHY RIGHTSIZE?

In projecting the decade of the 1990's, BusinessWeek Magazine selected 'downsizing' as one of the buzzwords of the decade. For the business community downsizing has taken on the definition of revamping an organization's cost structure. In terms of the MIS organization many people have interrupted downsizing as 'shooting the mainframe'. I like to view the opportunities technology has presented to us as a chance to 'rightsize' our information systems. The cost savings are there and a Cost Benefit Analysis will certainly bear this out.

As part of the strategic planning process, business will evaluate its internal operations from an efficiency and competitiveness standpoint. As we have seen with the history of the computer chip, information processing is now cheaper, and more efficient, thereby allowing our organizations to become more creative and competitive.

A new foundation of computer processing architecture has been set in motion. Industry standards have been established, computers can now talk to each other (and actually have something to say), and the reliability and maintenance issues associated with today's processors far outstrip that of the past decades processors. All of this means faster, cheaper, and better systems for business.

By no stretch of the imagination would I advocate scrapping the mainframe and implement a host of microcomputers. Hundreds if not thousands of person years have been invested into mainframe systems. In many situations it is more appropriate to offload applications from the mainframe onto a midrange system, such as end-user reporting. I have been in situations where a client either had to spend over $1 million on a mainframe upgrade or offload their reporting to another processor. The offload was the route taken, for economic and strategic reasons.

What we are witnessing is a gradual shift (ie: offloading) of mainframe applications towards the client/server environments. Processing of information is moving back to where it came from. To the desk, the department, the regional office. The benefits of the client/server architecture are numerous. Most importantly, it will allow business to react quicker and more efficiently to their customers' needs for less dollars.

The flexibility of this architecture will have a yet unmeasurable effect upon business. In the past, many businesses have restructured themselves in order to accommodate the computer systems. For some companies, this added bit of structure was welcomed. For others, the computer system represented a torture chamber. Jobs, departments, information flow was adjusted in order to make the best utilization of the computer system. This should no longer be the case.

Businesses can now 're-engineer' themselves. Processing architectures will facilitate this process. Individuals have the power of their own cpu to work with, and the ability to share their data and knowledge with other members of their corporate community. The power of synergy can be realized. What I find most exciting is the capabilities that the user will now have. By allowing users to become more productive, by giving them more powerful tools to get their jobs done quicker and more efficiently, more creative thought time will be available. This will result in a more competitive and productive business environment. Business can more proactive to the climate, rather than reactive. This will also translate into the heart of the MIS organization because mainframes operations typically require more maintenance resources rather than development resources. Under the new technical environments, far less resources are expended on maintenance, which can translate into more resources for development.

THE BENEFITS.

When it comes to making a business decision, it is the dollar which weighs the heaviest. There are many items to review when downsizing. However, the number one issue is typically money. The processing power found in many of today's midrange computers, (ie: HP 3000 Series 960) is equal and in many instances greater than the IBM mainframes. The cost per MIP has decreased from $100,000 to less than $25,000. Not only has the price performance drastically improved, but the cost of operating the systems has also decreased.

Hewlett Packard's RISC architecture (HP-PA) is the best example of why operating costs have decreased. HP-PA has created the power of one mainframe on one computer chip, while the IBM mainframe has hundreds of chips performing the same function. The simpler the design, the easier it is to maintain the hardware. This translates into lower support costs.

Also, mainframes require special environmental controls, either water or air cooling equipment. There are on-going costs associated with maintaining this environment such as power consumption and equipment maintenance. This equipment, as well as the mainframe and its peripherals, will also occupy a great deal of space. All of these items translate into higher operating costs than with a midrange computer.

We have seen that the hardware alone costs 30% to 60% more to operate in the IBM environment than with HP. The software costs make the economic argument even more compelling. As an example, the software licence fee for a 4GL tool can cost half as much on the HP system than on the IBM mainframe. This cost ratio can also be translated to the software support costs. This provides more evidence to the cost benefit analysis.

In my experience, the cost justification of the HP system versus is the IBM mainframe is overwhelming. Studies have shown that by replacing the mainframe with a HP 3000 where appropriate, the business can realize a payback of less than 3 years. The economic justification is overwhelming, but I also believe it is important to examine other factors when making the downsizing decision.

THE BENEFITS OF INDUSTRY STANDARDS.

Because of the developments in system standards (ie: Open Systems or a standard within a product line like the HP 3000) MIS and business can be a lot more flexible in their choices of hardware, software and development tools. Standards has brought us the many 'abilities'. All of these abilities have helped to lower operating and development costs, as well as increase productivity.

With standards based hardware and software, the size of a system can be configured to match the organization it is supporting. This is called scalability. As an example, an company creates a sales force automation system. The system is developed in 4GL. The 4GL can operate on any HP 3000. It also operates on a PC. Each regional office can have an HP 3000 of appropriate size (917, 937...). Corporate headquarters requires a HP 3000 Series 980/100. The field sales force uses laptop computers. All three groups (sales force, regional office, corporate headquarters) are utilizing the same software on the appropriate size computer. The portability of the software makes scalability possible.

In this example we see two types of cost saving. There is a one time investment in software development because the software is portable across platforms. The appropriate size computer is implemented with each business group due to the scalability of the architecture. What makes all the systems communicate properly is the interoperability of the systems.

The idea of scalability and portability is extended into the development environment. Many of today's development tools can be used on PC's, allowing a developer to be more productive. A developer no longer has to wait for a program to compile because the compile stream is competing against the production system. Many MIS departments are purchasing low end systems (HP 3000 S/917) for development. The software can then be ported to the production systems. This strategy provides a dedicated development environment whose test environment can mimic the production environment. Also, by being on a low-end cpu, the development licence and support costs are less expensive than being on a larger cpu.

Development tools have also become more flexible and powerful. Under a NewWave or Windows environment, a program can multi-task his/her work. The programmer can be executing a program in one window, utilizing a debugger in another window, and reviewing the source code in third window. And all of this is being performed on a dedicated cpu. Studies have shown that the Improved development tools along with a dedicated processor have increased development productivity by about 30%.

The increase in productivity does not stop at development. The implementation of client-server architectures have had a great impact upon business. Client-server systems place the power closer to the user, utilizing the features of scalability.

A major bank on the West Coast implemented a client-server system for processing loans. Instead of loan applications going through a back room process of data entry and batching, the loan officer could process the application on a PC. The PC would access the network to retrieve credit and other relevant information. A customer knows within 15 minutes if they qualify for the loan. Under the traditional mainframe scenario, the customer did not receive notification for 2 weeks, which many times resulted in lost business for the bank. This client-server model saved business for the bank, provided superior customer service, and allowed an employee to be more productive with their time.

Training is also receiving a boost from today's standards.

Graphical user interfaces (GUI) are making systems simpler to use. Users no longer have to type in commands or navigate through cumbersome menu trees. GUI's allow users to relate to graphical objects. By concentrating on objects, it is easier to train users, and for users to retain system knowledge.

The net result of all of these benefits is a more competitive business. Downsized systems eliminate unnecessary costs, allowing the business to invest more resources in development or more money to the bottom-line (a favorite of many stockholders). Downsized systems can place the resources at the appropriate level within the organization, thereby allocating more computing power at less cost to those who need it. Systems can be developed more proactively and productively, allowing the users to become more proactive and productive in their functions. This has a rippling effect on the organization as a whole, making your company more profitable, productive and competitive.

THE DOWNSIZING PROCESS.

Now that the benefits have been laid out, how do we get to this new technological vision. What I would like to address here is only one portion of downsizing. This portion reviews how to move your application software from the mainframe environment to a midrange machine, either the HP 3000 or HP 9000 (Rather than specifying the 3000 and 9000 I will just refer to HP). By moving to this platform, you will be able to take advantage of mainframe processing power on a midrange computer. The HP also has the ability to act in a client-server role, one which we highly advocate.

Downsizing is not a simple process. There are many challenges which lie ahead, technical, functional, and people. Moving from the mainframe is not a simple one to one exchange, going from the mainframe to the mini or micro. The downsize will likely require moving to more than one system. Remember, we are moving the processing to where it belongs, going from centralized to decentralized. This may mean putting a lead tracking system into regional sales offices, a warehouse/inventory system in each distribution center. This will impact data communications, the MIS staff, and the business users.

The people issues are perhaps the greatest barrier to a successful downsizing project. Technical issues will also pose some hurdles, but these are easier to overcome. From our experiences, we find the MIS staff the most reluctant to change. They do not want to learn and operate in the new environment. After all, change represents the unknown and it is human nature to fear the unknown. Therefore, an education process is required. Management commitment is necessary. FUD (Fear, Uncertainty, and Doubt) will rear its ugly head. The incumbent vendor will help to spread this. MIS staff help to spread this. The education process is critical.

To begin the downsizing process, we look to the beginning of the systems development life cycle. We term this Information Planning. This is the segment where the vision of future computing is defined and established. Business issues are analyzed such as; what operational advantages can be established which provide a competitive advantage, how can we lower our cost structure, how can functionality be improved?

The downsizing process gives us the opportunity to answer these questions. Moving from the centralized mainframe architecture provided the opportunity to mold our information systems into a framework which can truly support the future growth of our organizations with a flexible, powerful architecture. This is the time to build that strong base. This is where management commitment is vital.

Management commitment can be garnered in many ways. One tool which we often use is the cost benefit analysis (CBA). As mentioned earlier, the economic arguments for downsizing is quite apparent. The CBA is a powerful tool to get management involved. In order to execute the downsizing plan, a budget will need to be allocated. This is a strong indication of management commitment. This is an instance of spending money to make money, and the CBA should bear this out. As I once heard someone say about downsizing, "It takes bucks, guts, and people".

Now that we have embraced a 'vision', we need to get down to the dirty details. First, you want to identify a candidate application. I would not recommend trying to migrate your universe. There is a learning curve involved, so a methodical, phased approach is important. What can be learned from the first application can certainly be applied on future project.

There are two basic approaches that can be taken in migrating a mainframe application. You can rewrite the application or migrate the current source code to the target hardware platform. Both approaches have their pluses and minuses. I will list them briefly.

REWRITE OPTION

Pros	Cons
Increase functionality	Retraining of MIS & Users
Improved maintainability	Lengthy development time
'State of the art'	Business risks
Best fit for new platform	Cost

AUTOMATED MIGRATION

Pros	Cons
'I know what I have'	status quo functionality (may not even be a con).
Length of project	
Ability to quickly prototype	
Reduce business risk	
Quick 'win'	
Leveraging an existing investment	

Performing a rewrite is very similar to our traditional application development project. Therefore, I will not even examine this option. The migration approach poses an interesting opportunity to due the brevity of the project and the ease of cost justification.

There are several products available today which allow a traditional mainframe user to migrate their applications from that environment to an HP system. The tool we have chosen to work with is Conveyor, from Infosoft Gbmh of Germany.

Conveyor allows the following migrations to occur.

(place chart here).

From this migration we have application programs up and running on the target system, 'bug for bug' (for lack of a better term). Conveyor does not perform any emulation of the source environment. Conveyor will also set up your applications for the world of Open Systems. Many of the features Conveyor migrated coded are POSIX compliant.

Cobol is migrated to ANSI standard cobol, the screen handler is Vplus or Curses (Posix), the data structures are Turboimage or C-isam (Posix). There is also the option to go to Allbase/SQL (Posix). The only manual effort is translating the JCL to either the Unix Shell scripts (Posix) or to MPE/XL job streams. Conveyor will also perform the data conversion.

Conveyor and other similar tools make the migration option very attractive. The ability to perform a prototype migration facilitates the learning curve and improves the opportunity of success. These tools do work, however they are not the total solution. The actual migration is only half the work. There are many other steps which I will present as part of our downsizing methodology.

GETTING FROM HERE TO THERE.

Earlier, we discussed the planning process, obtaining management commitment, creating the vision. I believe this is necessary in any large-scale development project. These preceding steps are vital to any project of this nature. The following are a list of steps we have incorporated into our downsizing methodology. Detail is provided where necessary.

Approach to completing the work:

1. Organize the project. A project work plan is necessary in order to determine resources, such as people, hardware, software, forms, quality assurance checkpoints, status meetings, budget utilization and timeline.

2. Project Orientation. The project team should meet for a walk through, reviewing project objectives, methodology, business environment, resources, and project reporting procedures. This is also a good opportunity to identify any project training issues.

3. Application Orientation. Functional understanding is critical to the success of the migration project.

4. System Technical Design Overview.

5. Inventory of Screens, Reports, Files, and Processes. These inventories are used to a control mechanism in order to monitor the migration process.

6. Data Structure Design. A great deal of this work is done by Conveyor. However, we recommend reviewing all data structures for complexity and completeness. This is includes work and temporary files typically found in batch processes.

7. Create Application Tapes. These tapes will transport the source system to the target hardware platform.

8. Establish the target (HP 3000/9000) Environment.

9. Load source tapes onto the target system. This is a good checkpoint to ensure all source application programs have been loaded onto the target system. This is just one of the points at which the inventory of system components is utilized.

10. Prepare Conveyor for Conversion.

11. Execute Conversion. Before actually converting all programs, a prototype should be run in order to ensure that the environment and Conveyor are properly configured.

12. Monitor the Conversion Process.

13. Review Conversion Results. Conveyor will create conversion reports which will list any exception conditions. Exceptions are addressed in steps #14 and #15. The inventory of system components will be used to audit all converted programs and files.

14. Address Exception Conditions. The exception reports will list out any file or program which contained a condition that Conveyor could not handle, such as an IBM system call. These conditions will most likely require manual intervention.

15. Create the JCL equivalents for the HP 3000/9000.

16. Conduct Acceptance Test.

17. Documentation of the System.

18. Training. Training is available for a review of the application system as it resides on the HP 3000/9000. Training will be custom tailored for the clients needs.

Experience has shown that the migration tools work. As stated earlier, downsizing is not a simple process. However, the rewards awaiting the project team, MIS, and business are immeasurable. Costs will go down, MIS will become more of a proactive development group and spend less time in a support role, and the business will benefit in terms of competitiveness, productivity, and profit.

WHERE DO I START.

Downsizing may not be for everyone. There are a lot of questions that should be asked before jumping into this process. Downsizing is a major commitment full of challenges and changes. Here are a few items which you may want to examine:

1. Compare your support costs for hardware and software against those of a comparable midrange computer.

2. Compare your software license fees against those of a comparable midrange computer.

3. Is over 30% of your programming staff dedicated to maintenance of current systems.

4. Compare the cost per MIP.

5. What is the cost of maintaining your current computer facility, including power usage, cooling units, cost per square foot? Is there a need to expand the physical facility?

6. How large is your development backlog and how productive is your current development backlog? Does compiling programs compete with production resources?

7. Are your systems mainly batch?

8. Would the business benefit from better functionality and improved response time?

These are some of the many questions you should be asking. The economic argument is easy to prove. The next step is to begin tackling the process. It is not simple (is anything in hi-tech simple?). However, the rewards are tremendous. Now that American business has entered the era of downsizing and fine tuning their operations, MIS has the opportunity to set the tone and lead the charge.

3017

LEADERSHIP IS A MATTER OF PERFORMANCE

J. B. WATTERSON
Computer Data Systems, Inc.
20010 Century Blvd.
Germantown, MD 20874
301-903-2179

LEADING

"Leading is guiding people beyond the current context"

History demonstrates that individual performance is the key to the success of any endeavor. Individuals step forward, accept the challenge, and become leaders. Leadership is based on the performance of you and your people.

Each and every one of us holds a position of leadership, power, and authority, though we often fail to recognize it. Some forms of leadership are obvious. We immediately think of government officials, administrators, managers, and CEOs. But there are many other positions of leadership, power, and authority - in the office, in the home, and, as a citizen, an office-bearer in an organization, and even amid friends.

A mother's four kids were in the livingroom playing with baby skunks. The mother walked into the room, saw the skunks, and yelled, "Run." Each kid picked up a skunk and ran. Leadership in this illustration was not a matter of performance, but a matter of panic.

Leadership is guiding people beyond their current context. We all have the potential for better leadership by taking our gifts, abilities, and potentialities, and developing them to the fullest.

For our first illustration, let's go back some four thousand years, and look at the story of Joseph; how he accepted leadership and guided people beyond their current context. Joseph did this by using what we call, the "URP" principal, or **understanding, risking, and performing.**

At the age of seventeen Joseph was sold into slavery in Egypt by his brothers. Joseph could have languished in self-pity as a servant of Potiphar, and thought about the weaknesses of his brothers and his captors and all that he didn't have. But Joseph was proactive. He became involved and committed to service. He understood himself, understood others, and quickly learned from others. Within a short period of time, he was running Potiphar's household. He was in charge because leadership was a matter of performance, and the trust was so high. Potiphar's trust, the household's trust, and trust in him.

Then, the day came when Joseph was caught in a difficult situation with Potiphar's wife. He took a risk, and refused to compromise his integrity. As a result, he was imprisoned for thirteen years. But, again, he was proactive, involved, and committed. Based on his leadership capabilities and knowing that leadership is a matter of performance, Joseph was soon running the prison, and, eventually the entire nation of Egypt, second only to Pharaoh. Joseph achieved victory through excellence.

Leadership is such a big topic to discuss in less than one hour, so for my presentation, we will focus on the URP principle - understanding, risking, and performing.

UNDERSTANDING

"Some come to the fountain of knowledge to drink, others just to gargle"

KNOWING YOURSELF

NO INVOLVEMENT, NO COMMITMENT

Without involvement, there is no commitment. Write it down, asterisk it, circle it, underline it. No involvement, no commitment!

One of the best ways to know yourself is to listen to yourself - listen to your language. Is your language reactive or proactive? Reactive language is negative. For instance, "There's nothing I can do," "I can't," and "I must." Proactive language is positive. For example, "Let's look at our alternatives," "I choose," and "I prefer." Our language is a very real indicator of the degree to which we see ourselves as proactive individuals.

You want to become self-expressive, listen to your inner voice, and have a passion for leadership. The qualities of a good leader include integrity, decency, honesty, and caring. *Accept responsibility, blame no one. Listen to yourself, become involved, and be committed.*

A new road to weight loss, recently refined by author/psychologist, John Bear, is to create a situation where you are involved, committed, and dare not fail. Keys:

1. Pick a reasonable goal.

2. Devise an appropriate penalty (painful enough to really motivate you).

3. Select a reliable trustee who won't allow you to wriggle out at the last minute.

In Bear's case, he put $5,000 into a trust fund, payable to the American Nazi Party, if he failed to lose 70 pounds in a year.

Result: He lost the weight. No involvement, no commitment. Listen to yourself.

CAPTURE THE ESSENCE OF YOUR VISION

You are empowered to catch the vision and dream the dream. To understand yourself is to have a vision and create a mission. "What do I want out of my life, my profession, or my career." Most companies and organizations today have established "Mission Statements." They consider them vital to their success. You should consider writing a mission statement for your professional career role. This takes a good deal of thought, but it can show where you are headed. Your vision, ideas of roles, and goals can be identified in a simple mission statement. For example:

My mission is to be the best programmer or analyst by providing my company and my customer with quality products and services so they can achieve their goals. To achieve this mission requires that I:

1. Provide innovative talent, adapt to changing circumstances, be flexible to implement new methods and technologies as they become available, and above all, be a committed team player.

2. Provide added-value products and services.

3. Work in a straightforward, honest way with my peers, supervisor, and customer.

4. Project leadership by being involved and committed to servicing my customer.

Notice the positive, proactive key words in this mission statement. Use these types of words in yours. The mission statement will help you to reflect on your experience and where you are headed. It may give you insight to shape events rather than being shaped by them. On the last page of this paper, I have included a mission statement template for use in developing your own vision.

Robert Townsend, author of Further Up the Organization, has advice on work commitments. "If asked when you can deliver something, ask for time to think. Build in a margin of safety. Name a date. Then deliver it earlier than you promised. The world is divided into two classes of people: the few people who make good on their promises, and the many who don't. [Get in Column A and stay there.] You'll be very valuable wherever you are."

GO BACK TO THE BASICS

Encourage yourself to follow your intuition. I've read where intuition is 100% right, but it takes time to hear it correctly. Listen for it. Use your own best judgement and have faith in yourself. Put trust in yourself - be positive, enthusiastic, committed, and honest in your approach; again, listen, listen to yourself. Use your intuition or gut feeling along with logical facts.

Whenever Vince Lombardi would get his veteran players back to football camp each year, he would start off by saying, "This is a football." He started with the basics, and you should, too.

UNDERSTANDING OTHERS

"Seek first to understand, then to be understood"

LEARN BY LISTENING TO OTHERS

We are taught at a very early age to speak. Then, we spend years learning how to read and write. But how many of us have been taught to listen? What training enables us to listen to others so that we really understand their frame of reference from our own? The ability to listen is absolutely critical to your effectiveness as a good leader. You need to build the skills of good listening on a base of character that inspires openness and trust.

Most of us listen with the intent to reply, and not the intent to really understand the other person. Seek first to understand the other person, and then for them to understand you. Try to understand where they are coming from - get inside the other person.

Communications experts estimate that only 10% of our communication is represented by the words we say. Another 30% is represented by our sounds, and 60% by our body language. Yes, you do need to listen with yours ears, but you also need to listen with your eyes and with your heart. Listen for feeling and for meaning, and watch for behavior.

Set up one-on-one time with your staff. Listen to them; understand them. Try to get honest, accurate feedback. When you listen, you learn and you inspire loyalty. Ask questions. For example, "Now, let me see if I understand what your objectives are and what your concerns are about our ....." By taking this initiative, it does not mean being pushy, obnoxious, and aggressive. It does mean recognizing your responsibility to make things happen. Here's a good example.

Two battleships assigned to the training squadron had been at sea on maneuvers in heavy weather for several days. The captain was on the lead battleship and was on watch on the bridge as night fell. The visibility was poor with patchy fog.

Shortly after dark, the lookout on the wing of the bridge reported, "Light, bearing on the starboard bow." "Is it steady or moving astern?" the captain called out. Lookout replied, "Steady, captain," which meant they were on a dangerous collision course with that ship.

The captain then called to the signalman, "Signal that ship: We are on a collision course, advise you change course 20 degrees." Back came a signal, "Advisable for you to change course 20 degrees. "The captain said, "Send, I'm a captain, change course 20 degrees." "I'm a seaman second class," came the reply. "You had better change course 20 degrees."

By that time, the captain was furious. He spat out, "Send; I"m a battleship. Change course 20 degrees." Back came the flashing light, "I'm a lighthouse." The captain changed course!

Seek first to understand before problems come up, before you try to evaluate and reply, and before you present your own ideas. It's effective - try it!

LEARN FROM FEEDBACK

Feedback is critical. Poor feedback continues to be a major problem in our organizations today. Survey after survey indicates the staff, management, and our customers are not getting feedback from each other. As a first step, ask for and listen to feedback, and ensure you are providing the same to others.

To increase feedback, ask your customers or clients if they are pleased with your services or products. Do this regularly. Ask your supervisor and your staff if there are things you could do to improve. Continually evaluate your own performance. Ask yourself if you were your boss, your staff, or your customer, would you be pleased with your performance?

The more individuals know about how they are performing on the job, the more enjoyable the job becomes. Your boss, customer, and even your peers can help do this through feedback.

Provide encouragement in feedback situations. Encourage others that they might grow. Encouragement is like peanut butter; the more you spread, the more it sticks.

Understanding yourself and others, and encouraging feedback is essential to leadership and performance. Joseph did, will you?

RISKING

"Feel the fear and do it anyhow"

When I spoke at Interex last year, my subject was "If It Ain't Broke, Don't Fix It," or how to make your services and products better even if nothing is broken. It included the "IRA principal." What is the IRA principal? - Idea, Risk, and Action. Risk was a major topic.

When I was doing my research for this paper, almost everything I read on leadership again addressed risk. Risking is a daily activity, and it is very important in leadership. That is why again I want to again discuss risk.

Federal Express has been an advocate of the wilderness experience and found it to be very helpful. One of the exercises they go through is the "zip line," a rope with a T-bar on a pulley. Each participant (with safety harness attached) must jump off a cliff and slide 500 feet to the ground on the other side of the river. "Jumping off a cliff can be terrifying," Mr. Yamahiro, Federal Express, admitted, "but the purpose of this activity is to confront reluctance and take the risk. It helps participants build confidence in their ability to handle any situation."

BOOST YOUR TOLERANCE FOR RISKING

To boost your tolerance for risk and to make wise choices, ask yourselves the following questions with the "I" at the center of the inquiry:

1. What will I gain if this risk pays off?

2. What might it cost me if the risk boomerangs?

3. Is the gain worth the possible cost? (Will I suffer more if I don't take the gamble at all than if I take it and fail?)

4. Have I made this sort of choice before, and, if so, am I about to repeat a pattern of mine - negative or positive?

5. Will I be better able to make my decision if I first discuss it with someone I trust?

6. Is there a chance that my risk-taking behavior will seriously hurt someone, myself or others? Is it worth it?

With clear answers to these questions, and, with a fully-raised consciousness, if you still want to take the risk, then go to it!

A comic strip once depicted Nancy resolving to "stop smoking cigars, stop eating sushi for breakfast, and quit parachuting off office buildings. "You don't do any of this stuff," responded Sluggo. "What kind of New Year's resolutions are these?" To that Nancy replied, "Easy to keep." On the other hand, it is common for people to make resolutions that they have no intentions of keeping. Why? They just don't want to take a risk.

Here's a new risk you can try. Begin taking limited authority to do what needs to be done. If no one complains, continue the process.

DELEGATE AUTHORITY

Delegation of authority can be a risk too, but, again, one that leaders must be willing to take. Whether in the office or in some outside organization, the leaders' can't do it all. They must be willing to risk and delegate authority.

Supervisor or managers cannot delegate their responsibility. They are and will remain responsible for their work or organization. Everything that happens - good or bad, including the actions of their staff, remains their responsibility. Yes, management can assign responsibility, but they can't delegate it.

However, to give away your authority or power to one of your staff empowers them. This is what "delegation of authority" is all about. Such delegation gives the other person the power or authority to make decisions in your place so that you can turn your attention to other matters. The limitations you place on a person's authority is a measure of trust, and limits what they can do for you. But, when fear, inertia, or ego causes leaders to not delegate their authority, they are limiting themselves as well as others.

While giving authority to subordinates means that they now have the power to get you in trouble, they can also make you look very good through the high performance that comes from having a powerful, motivated staff.

"Delegation of authority" can be the antidotes to the "Peter Principle." The person who can delegate effectively, can command armies; the person who won't, will always be "small potatoes."

THINK WIN/WIN

Negotiating with others is always a risk, but a chore that must be done if you are a leader. Who wins and who loses can affect each ones performance. We need to strive for the Win/Win situation. That is - it's not your way or my way, but a better way. It means that the agreements or solutions are mutually beneficial, mutually satisfying to each of us. We all feel good about the decision and feel committed to the action plan we agreed to. It is a cooperative agreement, not a competitive one as you might find in labor union and management negotiations. This story illustrates the point.

A prisoner asked the guard for a cigarette. The guard answered, "No." To that answer, the prisoner said, "I'll scratch my face, beat my head against the wall, and tell others you beat me up." The guard replied, "No one will believe you." The prisoner then stated, "I know no one will believe me, but once you go through all the investigations, you'll wish you gave me a smoke." The guard gave the prisoner a cigarette.

Dealing with people isn't a football team where one team **must** win. Most of the people you know really don't want someone to lose. Vince Lombardi said, "Winning isn't everything, it's the only thing." That's OK for competitive sports, but in day-to-day situations what matters is that people get what they want and you get what you want.

When there is no sense of contest or competition, Win/Win is probably the most common approach in everyday negotiations. Think of it in these terms - The Win mentality thinks in terms of securing ones own ends and leaving it to others to secure theirs. Take a risk and use the Win/Win approach in dealing with people. Trust is the essence of Win/Win. Without trust, the best you can do is compromise. Without trust, you lack the credibility for open, mutual learning, communication, and real creativity.

The next time you have a disagreement or confrontation with someone, attempt to understand the concerns underlying that person's position. Address those concerns in a creative and mutually beneficial way.

Stehpen R. Covey tells the following story. "My best experience in teaching university classes have come when I have created a Win/Win shared understanding of the goal up front. This is what we're going to accomplish. Here are the basic requirements for an A, B, or C grade. My goal is to help everyone of you get an A. Now you take what we've talked about and analyze it and come up with your own understanding of what you want to accomplish that is unique to you. Then let's get together and agree on the grade you want and what you plan to do to get it."

At CDSI, we are doing something similar by having employees complete an Individual Development Plan (IDP). The staff develops a plan in three primary areas - product quality, staff development, and customer outreach. Managers do the same, but also include administrative efficiency and fiscal responsibility. The individual develops the plan, then meets with the manager to negotiate any changes. We strive for the Win/Win approach.

Those who do not risk, have nothing. Be courageous and risk!

PERFORMING

"To achieve great things, people must believe they can do great things"

Gordon S. Glenn, President of CDSI, has been asked frequently over the last several years what it takes to "move up" within CDSI - either in the technical or managerial ranks. He always responds with a simple answer, "take your job personally. By this I mean that you should realize that you have a personal obligation to do your very best to meet the needs of your clients, and just as importantly, those of your co-workers."

THE RIGHT KIND OF LEADER

"Lead by example"

The leader is the one who climbs the tallest tree, surveys the entire situation, and yells, "Wrong jungle!" But how does the busy producers or managers often respond? "Shut up! We're making progress."

Leaders can be characterized in one of four ways. It is necessary for you to determine which one of these you consider yourself to be today, and then decide which one you want to be in the future.

1. The leader who delivers on commitments and shares the values of your company. The future of this leader is an easy call - onward and upward.

2. The leader who does not meet commitments and does not share your companies values. The future of this leader is not such a pleasant call, but an easy one - out.

3. The leader who misses commitments, but shares the values of the company. Give this person a second chance, preferably in a different environment.

4. The leader who delivers on commitments, performs well, but doesn't share the values the company must have. Too often we have looked the other way - tolerated these leaders because "they always deliver," at least in the short term. We can no longer tolerate leaders who suppress and intimidate others. Convince these leaders to change, recognizing how difficult that can be or part company.

Mr. Spock from the Star Trek series is a Vulcan. A major trait where Vulcans differ from humans is that Vulcans have no emotional interference when making decisions. As human leaders, you must accept the emotional interface in whatever type of leader you become.

Which one of these leaders are you? If you aren't number 1, you need to change. Yes, it will be difficult, and you may need some help in doing so. But do it! This will determine your future, and it will be based on you and your companies mutual trust and respect.

TEAM BUILDING

Canada geese learned team building thousands of years ago. They change leadership every 20 minutes and continually honk to encourage each other. Through team work, they can travel 70% further as a flock.

The ability to effectively lead teams of people is at a premium in today's work environment. Higher demands for quality, service, efficiency, innovation, and cost control mean that organizations must be able to respond quickly to changing conditions and directions. To meet these demands, organizations are depending more and more on teams of people to work across functions, using their combined brain power to get more leverage on a problem and get the job done. Leaders who know how to organize and run teams successfully are key to creating organizations that excel.

If team members do not fulfill their roles, then someone else - a co-worker, staff member, or boss - will have to pick up the slack. The best team members are those who do their own jobs, but also pitch in and help each other when needed. We can provide top- quality products and services if we all work together and understand our obligations to our customers, our co-workers, and ourselves.

I was a participant in a team building exercise recently and was amazed at the results. The situation: a team was working together in the basement of a 10-story office building in California when suddenly an earthquake hit. The lights were out, the only exit was blocked, but they found a few items such as candles, a flashlight, and food that could be helpful.

The first exercise was for each team member to act independently to prioritize a list of 12 actions they would do first, and identify which ones they should not do at all. The second exercise was for the team to discuss the 12 actions, as a team, prioritize the list, and identify the action that should not be done.

We then scored each listing based on the actions earthquake experts prioritized or would not do, and calculated the differences among the expert scores, our individual score, and the team score. The team variance score was significantly better than any individual's score (8 verses a high of 38). This 30-minute exercise taught us the effectiveness of working as a team.

The following tips are provided for leading a successful team:

1. Employ the "Make Me Feel Important (MMFI) principle." Work to make individuals feel worthwhile and communicate the importance of their contributions. Always focus on the situation, issue, or behavior, not on the person.

2. Provide variety. The more tasks or activities an individual performs, the more enjoyable the job becomes.

3. Delegate authority. We talked about this earlier. People need authority to complete their assigned tasks

successfully.

4. Give feedback and maintain constructive relationships. The more individuals know about how they are performing on the job, the more enjoyable the job becomes.

5. Increase the spice of life. Interest in work is created if people volunteer to assist in social events, research, or small special assignments.

6. Take initiative to make things better. Let the staff members develop their own procedures. They know how to do their jobs better than management does. If they develop them, they will follow them and make things better.

7. Ensure your staff is trained adequately. Trained workers make fewer mistakes, can use their procedures more efficiently, and tend to be more motivated.

8. Reward workers for increased productivity. Workers do those things for which they are rewarded. The rewards may be other than monetary; they can be job recognitions, special awards, peer recognition, and job privileges, or even an atta-boy once in a while.

9. Treat people with respect. Maintain the self-confidence and self-esteem of others. They are an important asset of any organization. When people loose their personal dignity, they lose the desire to produce. Treat people like you want to be treated.

10. Let DP staff select their own tools. We learn as parents that if you buy the toys your children really want, they will play with them, but if you buy them the toys that you would like to have, they often don't play with them. The same holds true with professional people.

VICTORY THROUGH EXCELLENCE

"I can live two weeks on one good compliment" Mark Twain

"It takes at least ten "atta-boys" to equal one 'you jerk'" Unknown

Most people do not reach the performance levels that they could. Why? Simply, because they have grown up learning all the things that they should not and cannot do. We need to forget our old conditioning and change.

There is a place in Australia where the water conditions are just the right temperature for barnacles to grow very quickly. In fact, so quickly that the fishermen must scrape the barnacles off the bottoms of their boat on a weekly basis. If they don't, the barnacles will become so heavy in a matter of a few weeks that their boat will sink. We are all bound by powerful earlier conditioning, and periodically need to scrape off our barnacles. If you don't, like the boats, you too will sink.

Again, let me illustrate with a true story on how a young man by the name of Vincent achieved victory through excellence.

- o A troubled boy.
- o A pig-headed determination.
- o A total committment to pursuing excellence.

Today, we know that boy as Vincent (Bo) Jackson.

People and professionals throughout the country are involved in continuous improvement and are dedicated to quality products and excellent services. These people know that if 99.9% were "good enough," they would have some very bad consequences. For example, if 99.9% were good enough:

- o Every year, there would be:
  - 20,000 prescription errors made
  - 15,000 newborn babies dropped during delivery
- o Every month, there would be:
  - Unsafe drinking water for 43 minutes
- o For 1.5 minutes every day, there would be:
  - No telephone service
  - No electrical power in our homes
  - No TV transmissions
- o And, there would be:
  - Nine misspelled words on every page of every newspaper and magazine.

Excellence through quality, quality through people, and victory through excellence. After all, leadership is a matter of performance. Read the story of Joseph, and see how he performed.

TAKE THIS JOB AND SHOVE IT

It's no longer unusual to spend more time on transportation to and from airports, in waiting rooms, and in layovers and flight delays than actually in the air. Why do new highways develop gridlock? Why are simplified tax regulations harder to follow? Where is your leadership and your performance? Don't, as the old song title said, "Take This Job and Shove It," but "Take your job and change it." Apply the URP principal.

The downfall of America today isn't that we are lazy as the Japanese told us. No, it is because we have the "Save Your Butt" mentality. Or more appropriately referred to as "SYA." Why didn't Chicago officials take action when they dicovered the water faults? Why didn't Mexican officials do anything when they discovered gasoline leaks in the sewers? Because leadership wasn't a matter of performance.

Be a Joseph - understand, risk, and perform. Achieve victory through excellence. Open your eyes to the future, and go beyond what you're doing today. After all, individual performance is the key to the success of any endeavor.

My challenge to you is to step forward, accept the challenge, and become a leader. Why?

Because "**LEADERSHIP IS A MATTER OF PERFORMANCE.**"

MISSION STATEMENT TEMPLATE

THE MISSION OF ________________________________
(name)

IS TO PRODUCE/PROVIDE __________________________
(products or services)

TO____
(customer, supervisor or company)

SO THAT THEY CAN ______________________________
(accomplishment their goals)

Then, briefly state how you expect to accomplish this.

References:

- o Bible
- o The 7 Habits of Highly Effective People
- o Family Circle

Paper # 3020

The Process of Capacity Planning

A Case Study

Barney E. Green

CompuChem Laboratories

3308 Chapel Hill/Nelson Highway

Research Triangle Park, NC 27709

919-549-8263

Abstract

This paper will outline and discuss a stepwise approach to the justification, implementation, and monitoring of your computer systems resources and capacity.

An initial methodology will be outlined, defining specific goals of capacity planning and forecasting. A suggested method of justifying the toolset acquisition to management will be presented. A logical approach will be outlined to assist in this definition. An actual sample case will be discussed focusing on the various implementation issues regarding data collection and analysis. Manipulating and understanding the data is key to its proper use in the real world. After the data collection has been fully defined and functioning for a period of time, how do you make use of this data to forecast resources? Finally, a topic discussion will present different reporting formats available and suggestions on what management really needs to make use of the data.

The Problem

It's 3:30 P.M.. Your phone rings...... "Data Entry is Slow Again", the manager on the other end states. "Can't you make it go any faster". You immediately logon to the machine and guess what, it is slow.... really slow. WHY? Why do you have to hear this from your users first. Why are you letting people sit around waiting on terminal transactions when the work just keeps piling up? Well, take heart. It's really not "YOUR" fault. You aren't the one trying to shove all that work into the computer. Or, is it your fault. Shouldn't you be in touch enough with the business to insure there is enough computing resource available so that other business resources aren't wasted. I submit it is well within your charter as a manager, or keeper of the computer resources at your company, to ask the right questions and get involved enough in the business not to become the bottleneck to its success and growth. You must translate and characterize the computer systems, that may be the backbone of your company, into resources with a known, predictable capacity for performing whatever type of business or technical function your company requires.

How many times have you told management, "Well, if we had a bigger computer, Data Entry wouldn't be slow today." Probably too many, and they're probably tired of hearing it. While they sit at their desks running PCs with dedicated processors and storage, coupled with instantaneous response, you are headed for disaster. The sad thing is, so may your company. It is difficult for technical people to translate needs and reasoning into business justification for resource acquisition. Many times remarks like the one made above are just the icing on the cake that management needs to replace you and your computer system with a PC Network. Use the resources at your disposal. The computer itself can tell you what it is doing at any time. What files it is accessing, what programs it is running, which discs it is using. All of the information is there if you have the tools and the knowledge to present it to management in an understandable business form.

The Goal

The goals of capacity planning are as individual as each of your own businesses or organization. Ideally, they should directly relate to the company's goals. It is difficult to take a goal like, "We are going to increase Earnings Per Share to 30c this year", and discern what impact this will have on the computing environment. Is the company going to introduce a new product line? Are they going to cut expenses? Will the additional sales staff require more computing support? Is the volume of the product or service you produce going up or down? You will need much information from management to come up with the answers to the above questions. ASK!!! Go find out. If you are sincere in your interest, the answers are available. I will offer you a goal that seems to fit the times.

> Capacity planning is the art of providing your customer with what they want, when they want it, at a reasonable price.

Simple, eh. Your customer wants unlimited computing capability, available just before they need it, and for free! Just in time computing. Wouldn't that be great? Sounds very much like business, doesn't it? Well I'll give you a hint. IT IS. Computing resources can be a substantial investment. It is your job to make sure your customer gets his moneys worth out of that investment. It is much better to have a plan to attack capacity problems than it is to use 'gut feel' and intuition. Typically, you are just one manager fighting for scarce capital resources within your organization. You need to spend these resources wisely, but confidently. Capacity planning helps insure the computing environment won't become a surprise capital expenditure for organizations.

The Justification

Typically, this is the toughest part of the process. It is sometimes difficult for techies, and even management, to come to grips with capacity planning and the real need for it. You must present a 'business case' for the acquisition of the proper tools and training to perform capacity planning. Armed with the answers (obtained above) concerning the business goals, etc., you should easily be able to formulate the proper justification. Stay away from intangible benefits. They tend to be the easiest to blurt out, but the hardest to describe in terms of dollars. If they are substantial and quantifiable, include them. It is best to focus on real world, quantifiable benefits that can be obtained from the toolsets. If, for instance, you are a bicycle manufacturer. You have the information on the business plan. You also have information on seasonal trends. You also know of a few rogue programs on you system that you've been waiting to fix, but they never make it to the top of the project list. If your daily production rate is 200 bicycles/day and you can collect enough data to get a few of those programs fixed to increase the capacity to 225 bicycles per day, that is money. That money just paid for your new capacity planning toolset. Be conservative, but quantifiable in your estimates. Prepare a full business case with a detailed project implementation plan and you will be on the road to capacity planning. Don't get discouraged if you are turned down. Every business has priorities and sometimes the resources just aren't available at the time. Continue building your case and your rapport with management. If you cannot get the funding on the first cut, try to at least get some assistance from the vendor of your computer system. Most offer services that will perform rough cut data collection and analysis at a fraction of the cost for the toolset. You can use this information to make computing environment decisions that can positively impact your organization. With a little success under your belt, the toolset acquisition proposal may be more easily approved next time.

THE CASE STUDY

CompuChem Laboratories is a major Forensic Drug Testing Lab as well as an EPA Superfund Environmental Testing facility. The growth of CompuChem's business over the past decade has required the discipline of capacity planning be adopted.

This was not always the case. As with many HP shops of the mid to late 1980's, CompuChem was running several, well tuned, HP-3000 Series 70 computer systems. Even though we were getting a serious amount of work from our computer systems, we always seemed to be behind the requirements of the business from a computing capacity standpoint. Periodically, we would retain the services of HP to perform a "Snapshot" of our computer systems. These were usually well announced to the users, and consequently they responed in kind by hopelessly overloading the machines during the data collection periods. Their strategy usually worked and we continued to divide our applications and purchase more HP-3000 computers. Not long after the third purchase, we decided to perform a capacity planning study with the assistance of HP. Our major concern was that one segment of our business would soon be outgrowing one of the Series 70 computers, and the application would be difficult, time consuming, and costly to divide. After unannounced data collections at average and peak times, the work of data reduction, modeling, and forecasting began. During the reduction of the data, we became very familiar with terms like "workloads" and "thruput" and "queuing theory". We also learned an awful lot about many applications which we thought we could characterize reasonably well. Although we ended up ultimately dividing the application and purchasing yet another HP-3000 Series 70 computer system, it was totally justifiable and the acquisition was made in a smooth, planned manner.

Upon introduction of the new Precision Architecture HP-3000 9xx series computers, CompuChem was among the first to make the migration. With the capacity of the Series 70 well documented and actually validated, we were eager for the promised performance of the initial Series 950 computer. After extensive stability testing, and substantial batch testing, we made the transition to the new architecture. While not quite as smooth as originally predicted, there was a substantial performance increase in some applications, while other applications seemed to get worse. What had happened? We again began collecting data, characterizing workloads, and scratching our heads. Even HP was scratching their heads. How about more memory? Is the disc IO balanced? Why did you write the application that way? After much data collection, continued migration, increased memory, and ultimately a faster processor, everything was wonderful. Batch thruput had never been better, on line response time was sub second, the users were happy, but management was not. They were not happy with the unplanned capital expenditure for additional processing power so soon. They were also not happy spending time dealing with computer issues while needing to manage the substantial growth in the business. Upper management also became familiar with terms like "workloads", "thruput", and "queuing theory". Computer systems staff and upper management agreed to pursue the art of capacity planning. This meant teaching computer people understanding the business and asking those hard questions required to continue the process. It also meant that upper management must considered computing resources impact in assessment during business planning .

The Solution

A big part of any capacity planning solution is truely the communication between systems support and upper management. Armed with business planning information, you can start collecting data on your applications. You can even characterize new applications and their projected performance. CompuChem chose to implement an entire package consisting of Laser/RX, RX/Forecast, and SPT/XL. Laser/RX uses the same data collection software that your friendly performance engineer uses and is full of features. RX/Forecast helps you look into the future for resource utilization based on many different variables. SPT/XL is a tool that samples executing native mode code and reports utilization within the software. It points out the 10% of the code that takes 90% of the resources. While there certainly are other tools on the market available, we felt that the reliability of these products, the quality of the information producted, and the integration of the package made them the right choice for our environment.

Our approach was simple, configure the data collector for known applications, lump the rest of the programs into their own group, then let it run for a while and just see what really is happening on the CPU. This configuration of the application groupings takes several passes, and must continually be updated to remain correct. After continued refinement, we felt we really had the collector gathering information on our applications appropriately. Shortly after the installation of these products, the laboratory operations group changed the workflow timing through the laboratory. This ultimately caused considerable performance problems during our peak utilization periods. We were able to point out this fact to upper management and let them decide whether the operational change was worth the capital expenditure for a bigger processor to handle the workload in it's new configuration. It was decided that the change would be worth the expenditure, but management wanted to be sure that the new processor under consideration would perform as expected. We arranged to borrow a processor for a week from HP. During this time, we continued to collect data on the borrowed machine. At the end of the week, we swapped out the processor, sent our data collection off to our friendly Performance

Engineer, and changed the laboratory workflow schedule back to it's previous form. This data was then run through several modeling iterations on various procesor configurations at various business levels to determine several things. First, how long would the proposed processor last based on business projections. Second, with future business projections, which configurations would be the most optimal for consideration in the future. These questions, and several others, were answered with ease. It really made the modeling effort effective to be looking at data collected on our applications as we defined them. We presented this data to upper management. It was understood, accepted, and several acquisitions were worked into future business planning scenarios.

Since our initial effort, we have continued to refine our application configuration, and our business unit files with current information available. We have also been able to re-validate our assumptions derived from the modeling effort. This re-validation is easy, and costs much less that a full blown modeling project. To date, we have successfully met the growing needs of the business and are continuing to plan for the future. Armed with the proper information, both performance related and business related, it is easy to provide the service level required to our users. Unless other methods are the best fit for your business, we found it much more convenient, and accurate, to use the business units method of forecasting. Through this method, simply distill the business down to it's lowest level that is tracable, and reportable to upper management. It could be units of production, dollars shipped, average dollar volume of orders entered, etc. If the variable does not jump right out at you, ask management how they track the success of the business. Ask the production supervisors what numbers they look at to determine how good a day it was. Chances are that you will find that unit of resource consumption that can be correlated to computer resource utilization. Using the forecasting tools, track the variable against performance of the selected application set. A little brushing up on statistics will help, but always do a reasonableness check of the forecast. If the numbers look right, but the utilization is way off, dig deeper.

Management Reporting

Many report formats are available, and quite useful. We have discovered several formats that are the most communicative. By the way, a color printer or a plotter is a must with these packages. We have found that the global utilization reports summarized monthly give the best information. We also include one specific day of the month summarized at 5 minute intervals. This gives you the overall feel for your utilization and also shows you a granular look at an average day. There are also times we will track the release and performance of a new application for a period of time. Use the application groupings to segregate the new application from others currently defined, then report on it's utilization daily to the development managers or monthly to upper management.

SUMMARY

What has been presented to you are some specific goals of capacity planning worth implementing. A template for justification of the required toolset. And a methodology for taking the first steps with the toolset. As well as some hints on management reporting strategies.

Remember, Capacity Planning is an ART. Although much data collection and scrutiny are used throughout the process, you must always ask yourself, "Is this reasonable". Learn your business or service, justify the toolset acquisition, monitor your system loads, educate management, and by all means, be successful.

3021
THE EXPERIENCE OF RELAXATION

Margaret Brunner
Northern California Cancer Center
32960 Alvarado-Niles Road, Suite 600
Union City, California 94587
(510) 429-2500

Relaxation is the abatement or relief of bodily or mental effort or application. Relaxation produces physiological changes and is the physiological opposite of tension and anxiety. When a relaxed state is induced in the presence of a stressful threatening event, the physiological reaction to stress is inhibited. Mind and body must work together, therefore, a relaxed body cannot co-exist with a stressed mind and vice-versa. One method of handling stress is to induce a relaxed mental or a physical state.

One of the physiological phenomena that occur when one induces a relaxed state is that brain wave rhythms change as measured by an Electroencephalogram. The slower the brain wave, the more relaxed the body is. There are four types of brain waves and they are measured in cycles per second. The brain waves from most active to least active are BETA which measures from 14 to 40 cycles per second, ALPHA which measures from 8 to 13 cycles, THETA which measures from 4 to 7 cycles and DELTA which measures from .5 to 3 cycles per second.

BETA brain waves are indicative of the waking state and are best suited for reasoning, logic, and decision-making. ALPHA waves are indicative of the relaxed trance state which best supports learning, creativity, imagination, meditation and receptivity to new ideas. This is also your REM (Rapid Eye Movement) state where you have memory of your dreams. THETA waves occur during dreams and deep meditative states while DELTA brain waves indicate deep sleep.

High brain wave activity (BETA) will occur during stress. To induce relaxation, one must slow down one's brain waves into the ALPHA zone. Becoming more relaxed is inducing your brain waves to change. There are many activities that can induce this change such as Biofeedback, Meditation, Autogenic Training, etc; but they are all based on one or more of 4 simple techniques which are breathing deeply and slowly, relaxing the muscles of the body, focusing internally, and imagining a pleasurable experience.

These 4 techniques can be used at any time. For example, during a meeting or while driving or when you are under pressure, you may start breathing deeply and slowly or you may become aware of the tension in your body and allow your tense muscles to go limp and relaxed. This will induce a state of relaxation.

Focusing your awareness into yourself and allowing yourself to imagine a calm, safe pleasurable experience will induce deeper relaxation. This can be done at home, or during your lunchtime.

The following exercises are all techniques to use to induce a more relaxed state. They will give you the experience of slowing your brain waves from BETA to ALPHA. Each of these techniques will allow you to proceed from tension to relaxation or from relaxation to greater relaxation. Each of the four exercises listed below will use one of the 4 techniques mentioned above.

The best way to use the following exercises is to either record them into a tape recorder and listen to the tape or to have someone else read them to you while you experience them. These scripts must be read slowly, very slowly and rhythmically. Read them one after another without a break for the greatest benefit. Pace it so that it takes an hour to go through all 4 scripts listed below.

When you do the exercises, anchor all of your attention into the listening and the doing of the suggestions given. They are experiential exercises requiring a person's whole attention. This is why reading them to yourself is not effective. This is like swimming, the value is in the experience, not in the reading about the experience.

During relaxation, the body may discharge tension through tingling, involuntary jerking, etc. Do nothing, this is normal. This is a physiological non-damaging release of neuronal impulses. They are considered to be normal responses to the body's attempt to self-regulate toward a state of increased balance.

Prepare to relax by taking a comfortable position. Loosen any tight or restrictive clothing. Close your eyes softly.

The breathing exercise listed below should be used all the time. This is the way babies breathe. It is a natural way to breathe to counteract stress.

DEEP BREATHING:

Settle back comfortably ... and become aware of your body... relax a moment... perhaps even close your eyes ... and take a deep breath... Let the outbreath be a letting go type of breath... do it one more time... and again...

Breathe in slowly through your nose as you expand your belly. Imagine that you have a balloon in your belly ... and, as you inhale, ... you are slowly inflating it, ... causing the abdominal area to swell...now... breathe out slowly through your nose... pull your abdominal muscles in ... as you press all the air ... out of your lungs...

Take another deep breath... and as you release the outbreath let it be a real release, ... a real letting go type of outbreath...

As you breath in, breath in energy... as you breath out, release tension, discomfort... you do not need to hold on to anything right now... nothing to do ... nowhere to be ... and as you relax more deeply ... just come back to this simple breath ... breath in deeply, ... breath out until your lungs are empty... breath in more deeply, ... releasing completely with the outbreath... just a natural, ... gentle process...

PROGRESSIVE RELAXATION:

Now focus attention into your body... focus your awareness on your left foot, ... and invite your left foot to release ... and relax any tension it may be holding... notice the beginning sensations of relaxation in that foot... in the same way, ... invite your right foot to release ... and relax any tension that might be there... invite the muscles of your left calf and shin to release... and your right calf and shin... just notice and allow uour lower legs to relax in their own way, ...

becoming more comfortable ... and at ease all the while...

Remember as each part of you relaxes, ... all of you relaxes more deeply... and as you relax more deeply, ... each part can relax even more easily...

Invite your legs to soften ... and relax... invite any tension to melt ... and to release... allow your legs to relax more comfortably... allow your knees and your thighs to relax... invite your lower back, ... hips, ... pelvis to soften and relax... to join in this letting go ... and release of tension... allow your entire lower body to release ... and relax... and notice the relaxing sensations... allow it to be a comfortable and pleasant experience...

Allow your belly to relax... the muscles of your abdomen ... and your midback... to join in this deeper, ... more comfortable state of relaxation... invite the organs in your abdominal cavity to also join in this letting go, ... this release... and just allow that whole lower half of your body to let go ... and become even more deeply comfortable and at ease...

Invite your chest muscles to soften and relax... and the muscles between your shoulder blades to release ... and relax... becoming soft and at ease... the organs in your chest joining in this deeper, ... more comfortable state...

Relax your muscles up ... and down your spine and ... across your upper back... imagine your shoulders ... and neck muscles becoming soft, ... releasing any tension that may be there... allowing them to take a well-deserved rest... and this relaxation flowing down over your shoulders...

Breathe into your shoulders ... and imagine they are softening ... and releasing tension on the outbreath... become aware of any tension in your shoulders as you breath into your shoulders... imagine the tension releasing with your outbreath... let your shoulders soften and relax...

Imagine that very pleasant sensation flowing down your arms, ... down thru your elbows and forearms... wrists and hands... all the way to the tips of the fingers and thumbs... invite all the small muscles of your hands... in between the fingers... to release and relax ... and become very comfortable ... and deeply relaxed... your index fingers... middle fingers... ring fingers... little fingers... and your thumbs... deeply relaxed... all the way to the very tips...

Just a gentle, natural, comfortable peaceful sense of letting go...

Let your scalp and forehead go soft and at ease... feel the muscles in your face relaxing... softening and releasing tension... the little muscles around your eyes relaxing... to become smooth and relaxed... invite these muscles to release any tension ... and to feel that sense of letting go flowing through your face and jaws... your jaws relaxing ... your tongue relaxing...

And as your body relaxes more deeply ... your mind becomes quiet ... and peaceful as well...

Take a deep breath ... and relax even more comfortably... allow your mind to become quiet and still...

DEEPENING AND INTERNAL FOCUSING:

Now to deepen this comfortable state of relaxation and concentration, ... imagine yourself at the top of a stairway that has steps leading down from where you stand... let it be any kind of stairway... one you've seen before ... or one you just make up... and take some time to observe it in detail... notice what the stairs of made of... how steep or shallow they are... how wide or how narrow... do they go straight down, do they spiral, or is there a landing halfway down?... are the stairs covered with anything?... what is the texture beneath your feet?... hard, soft, cushy ... is there a bannister or handrail to hold as you go down?...

When you are ready, begin to descend the staircase one step at a time, ... counting backward from ten to one... allowing yourself to feel more deeply, ... more comfortably relaxed with each step you descend... let this imaginary staircase help you reach an even deeper, ... more comfortable level of body and mind with each step down...

... ten... feeling safe, ... calm, ... relaxed, ... limp...
... nine... deeper ... and more comfortably relaxed...
... eight... each step takes you deeper...
... seven... going down limp ... and relaxed ...
... and six... easily ... and naturally...
... five... halfway down... with nothing to worry ... nothing to bother...
... four... deeper... more comfortably relaxed...
... three... no need to worry about exactly how deeply ... or how comfortably you go...
... two... calm, relaxed, safe...
... and one... at the bottom of the stairs... very comfortable ... and deeply relaxed in body and mind...

GUIDED VISUALIZATION:

Now imagine yourself in a very beautiful peaceful place... some special place you may have visited in your life or some place you may have imagined... a meadow or a beach... just let the image come to you... it does not matter where you are, as long as it is beautiful, peaceful, safe, ... a beautiful place inside you where you can go for deep relaxation, for comfort, for rest... a serene, quiet place... a place where you feel very connected to the natural energies around you... a place where you feel secure and at one with your surroundings... maybe you've had a place like this in your life... somewhere you go to be quiet and reflective... a lake, a mountain, a meadow, or a beach ... somewhere special and healing for you... or it could be a place you've seen in a movie... or read about... or just dreamed of... It could be a real place... or an imaginary place like floating on a soft cloud...

Take a few moments to look around in your minds' eye to see what you see there...or sense or feel there ... let yourself explore whatever quiet imaginary place you go to as if you were there now... notice the details, the colors, the shapes... just notice what you see or feel in that place... feel the air or the breeze blowing softly across your face ... notice any sounds you might hear there... rustling leaves or crashing waves perhaps ... is there is a scent or aroma in the air you can smell... just notice if you can see, sense or feel these things...

But especially notice how it feels just to be there... the deep sense of peacefulness and quiet connectedness that you feel in this place... immerse yourself in the beauty, the feelings of peacefulness... of being secure and at ease... No where to be, nothing to do...

As you explore that special inner place, find the spot where you feel most comfortable there... a place that feels particularly good to be in... a spot where you feel especially calm... centered... and at ease... let yourself become centered and comfortable situated there... let this be your power spot, a place in which you draw from the deep sense of peacefulness you feel here... a place of healing... and of rest... and a place where you can explore and use the power of your imagination to best effect...

Take some time to relax into the deep feelings of peacefulnes, quiet, and healing you can sense in this spot... take as much time as your need... sit down or lie down in your minds' eye there...

Once you are comfortably seated or lying then notice the quality of the light there... look at this light from the sun... it nourishes all the life on earth...

Now take this light into your body... bring this sunlight inward... let this inner sun radiate light throughout your body... imagine that you become more and more aware of this light and that it grows stronger, larger, brighter, warmer... it is a very comfortable, pleasant experience... imagine this sun glowing in your chest... imagine it filling all the space in your belly, moving down your legs, moving up your chest, moving down your arms and up into your head... imagine that it fills all the space in your body so that the light and warmth is radiated and touches every cell in your body...

Just imagine that you have a control over this light and can increase the light in your body... you can turn it up like a TV or radio... turn it up now so it becomes even brighter ,stronger and warmer... always staying comfortable, not too warm but imagine that the light shines brighter and fills a larger space so that it not only fills your whole body but begins to flow into the space around your body and fills the space around your body for a foot in all directions... just warm, peaceful light... and it keeps expanding, radiating out from your body...

Imagine that it fills the space around your body for 3 feet in all directions... just relax in this peaceful, warm light... just enjoy it for a few moments...

Now in a moment, I'm going to ask you to come back... do so gently...

When you are ready, prepare yourself to come back to your waking state... remember, this is your special inner place, a place you can return to at any time of your own choosing... a place within where rest, healing, and peace are always available... a place that is always with you... and a place you can draw from when those qualities are needed... you will be able to remember everything that occured here, whenever it is appropriate for you to remember it...

To return to waking, but bringing back with you the sense of peacefulness and healing you have experienced here... all you need to do is to recall the imaginary staircase your descended... imagine yourself at the bottom of the stairs... as you ascend the stairs, you become more and more wide-awake, alert, and refreshed... feeling better than before...

(The following instructions should be given is a progressively louder voice and at a faster brisk pace to return the person to a normal waking state).

... one - two - coming up, becoming more awake -
... three - four - bringing back with you a sense of peace, of relaxation -
... five - when you reach ten you may open your eyes, stretch, and come all the way wide-awake -
... six - feeling refreshed as if you had a very good nap -
... seven - at ten you will be wide-awake and alert, feeling very good and refreshed -
... eight - your eyes may start to feel like they want to open -
... nine - and ten - at the top of the stairs and wide-awake, alert and refreshed

Open your eyes - stretch - smile - and become alert, refreshed, and wide-awake again.

3022
In House Computing: Survival of the Fittest

Doug Samuels
Space Coast Credit Union
20 So. Wickham Road
Melbourne, FL 32901
(407) 724-5730 ext. 201

INTRODUCTION

In the 1980's, the information technology department at British Petroleum, Inc. was flying high. BP had a heavy investment in technology, including a Cray super computer and a compliment of computer service and support staff. In April 1992, the internal information technology department was entirely disbanded with information services being outsourced.[1]

The cost of technology continues to drop. The processing power that used to cost millions of dollars and take up rooms full of equipment can now be had for under two thousand dollars and can fit on a person's lap. When computer equipment was prohibitively expensive most small companies contracted with service bureaus to handle their computerized information processing. As the cost of equipment decreased, most companies could afford their own computer equipment to handle their own needs. The cost of computer equipment has continued to fall, but more and more companies are outsourcing their computer information processing. Why? When a firm outsources its computer services, it can invariably be attributed to one of two reasons: **THE INTERNAL INFORMATION SYSTEM DOES NOT ADD VALUE TO THE ORGANIZATION or THE INTERNAL INFORMATION SYSTEM IS PERCEIVED TO NOT ADD VALUE TO THE ORGANIZATION**. It is this simple. Either the firm does an objective analysis and finds that value realized from its dollars invested in the information system are not adequate, or the firm has the feeling that the dollars being spent for technology are not enhancing the company's competetive position.

For many years, the computer departments in companies wielded considerable power because they held the power of computers. Computer departments were often enhanced and upgraded while other department's budgets were cut. Many data processing projects were undertaken on faith and intuition, and did not yield the cost savings expected. The investments in technology continued, and with it, the cost of data processing training, maintenance and upgrades continued to increase.

Many data processing shops have considerable investments in in-house applications that must be maintained.

BUSINESS WORLD CHANGES

In recent years, the mystique of data processing has worn. Part of this is attributable to the data processing department's expanding roles. Many small firms had used computers early on strictly for accounting, finance and other number crunching and counting functions. When computers were crunching numbers, it was very easy to quantify the value gained over manual computation. It was easy to compare service bureau costs to in-house costs because the functions were very limited, and easily quantifiable.

As computer resources became more prevalent, the computer became utilized in more and more areas of the core business functions. As the usage became varied, it became more difficult to quantify the value of the computer department.

As the computer departments became integrated into the rest of the organization, they became susceptible to the same scrutiny as the other departments. No longer could the computer departments expect to receive unquestioned budget allocations.

Problems arose when many computer department managers did not understand the changes that had taken place, and have not adjusted like many other classical departments have had to for years.

In the last several years, economic conditions have made most businesses very cost conscious. Formerly healthy companies, large and small, are scrutinizing budgets and investments over concern for survival. The change in the function of the computer department, coupled with lean economic conditions, is exposing more and more IS departments to outsourcing propositions.

In order for computer department managers to survive, they must deal with the realities of their new environment or disappear.

SURVIVAL TIPS

How is an IS manager supposed to cope? This paper discusses issues and observations based on a perspective of a business background rather than a technical background. Entering the technical field through business rather than the other way around has given the author the opportunity to view a company's technical department from the outside. The issues discussed are based on the opinions and observations from this perspective.

PERCEIVED VALUE

As stated earlier, there are only two reasons that an organization would choose to outsource its data processing needs. One of the reasons is that the organization does not perceive that the internal data processing resource is valuable. If the upper management does not see the value of your department, it is your fault! Nobody is going to look out for you, but you.

As the data processing department became integrated into the organization, nobody taught the data processing people any of the corporate survival skills that all of the other departments have refined for years.

Many data processing managers feel like their work is not appreciated, and that nobody really understands how much the data processing department really does for the company. If this is true in your case, you are probably right; nobody does appreciate or understand you. If this is your situation, then you are susceptible to extinction. You must understand that regardless of how great a job you are doing, if the perception varies, YOU are the one responsible and YOU are the one with the problem.

If your company does not understand you, you must make yourself understood. If the company does not appreciate you, you must make yourself appreciated. If you do nothing, there are plenty of very capable consulting firms that can make their value understood and appreciated at your expense.

You must get a feel for how you are perceived. The best way to discover how you are perceived is to ask. It is often very effective to create "customer" surveys that solicit users about the service the IS department provides.

In developing a survey, be careful not to tailor questions that may censor candid feedback. Take a deep breath and ask about the issues in which the IS department is deficient. The survey results should be accessible to all of the IS staff. It is not a necessity to quantify and tabulate the results. But it is important to keep the results for future comparisons. A survey should be sent out about once a year to measure progress in perceptions.

REAL VALUE

What have you done for the company lately? Are you worth keeping? Are you progressing in your level of knowledge and expertise? Are you learning new and different technologies?

How well will you stack up against an outsourcing company's proposal? Is your operation enhancing the profitability or viability of the company? As stated above, one of the reasons a company decides to outsource is that it determines that the internal information system does not add value to the company.

APPEARANCES

Physical appearances are extremely important to influencing perceptions. The IS department should reflect precision and accuracy. Always pay special attention to the cleanliness and order of your department. The computer area should be free from clutter and garbage. Although everyone gets busy trying to do more things than are possible, do not let unfinished projects find permanent residence on tables and shelves, etc. Do not let clutter and disorganization damage your department's credibility. Someone observing a messy department may unconsciously associate the disarray to inaccurate and imprecise work.

Get rid of old equipment. There are instances where a company decides to do an upgrade, while the old equipment is still depreciating. Vocational, or Academic institutions are often grateful for any equipment donations. This equipment could be used for their data processing needs, or for disassembly and reassembly experience. If you are in such a situation, check with your finance people before committing to anything to see if the expense can be absorbed in trade for some good public relations.

Personal appearance is also very important to perception. Although dressing comfortably is a nice idea, if it falls outside of the norm of the company, the whole department may be perceived as being removed from the company itself. A good rule of thumb is to dress in accordance with the management of the company. It is hard to appear integrated into the company if you appear to not belong with the company. Protective smocks can be worn over shirt and tie and a tie can always be removed.

TECHNICAL EXPERTISE

It almost goes without saying, but in order to survive, you must maintain your expertise with constantly changing technology. It is easy to get left behind. There will never be enough training dollars to ensure that everybody on the IS staff is technically up to date on the latest and greatest technology, so how do you keep up?

Not to sound too obvious, but a good way to keep abreast of technological changes is to read. It is easy to put off reading until you get some "quiet" time to yourself only to find that you do not have adequate time to read with the focus and depth that you prefer. Technical literature, such as *Interact* and *HP Professional* should be required reading for ALL IS staff members. Read these and other publications purposefully and thoroughly, and assign specific articles to be read by your staff members.

Reading goes beyond the specific technical literature mentioned above. In order to sharpen your business acumen, you must broaden the scope of your reading. Find out what other publications are in circulation around your company. Technical literature is fine for honing your specific technical knowledge, but other specialty journals and publications often contain technologies that are affecting the other departments in your organization. It is very valuable to your ability to sense the needs and expectations of marketing if you are somewhat familiar with some of the issues discussed in marketing's publications. The articles that marketing reads in their journals may be used as a gauge by which they will measure the service they are receiving from you.

Other general publications are also excellent resources to keep in touch with changes in technology. Publications like the *Wall Street Journal*, *Business Week*, and *U.S. News and World Report* are excellent sources of information regarding how technology is being applied. Consider that most of the higher decision makers in the organization have a more general view of technology, and are exposed to more general type of publications. It is extremely valuable to be dealing with the same type of information.

Everyone would agree that staff development and training is a fundamental survival necessity. The survivor will be very selective and careful when it comes to which training to undertake. Before sending one of the techies to that old standby technical conference where an incremental level of benefit might be realized, consider sending him/her to a conference or training session which will expose him/her to something different. This may cause a paradigm shift in this individual's perspective of his/her career purpose.

The whole point is, you will have to be good at what you do to continue to do what you do. You are responsible for either possessing or accessing the technical knowledge and skills to deliver information services that will enhance the organization's core business.

BUSINESS KNOWLEDGE

As well as developing specific and general technical expertise, the survivor will possess the business skills necessary to keep up with the other departments. Learn the business of your company.

A computer operations person worked for a business lock box processing firm for a number of years. Not till several years after he left the firm did he even know what the company's business was.

IS professionals, because of their systems experience, have a way of understanding work flows and relationships. Exploit this advantage by piecing together the BIG system; the company at large.

As you come to know why your organization exists, you become familiar with the key components that determine the organizations viability and competitive advantages. Understanding these components will help you prioritize the allocation of your resources. For example, if you discover that your organization's ability to process telephone customer orders quickly and accurately is a competitive advantage, give these departments higher priority on development than the pet EDI project you have been working on.

Understanding the financial ratios and trends is also important to the survivor. All organizations, with the exception of a few government concerns, must make more than they spend. Whether you work for a not-for-profit service organization, or a sales and manufacturing concern, your organization cannot operate in the negative for very long. By becoming knowledgeable with the financial measures of the health of the organization, you will better understand the goals and concerns of top management. Knowing these goals and concerns is helpful in understanding where efficiency and automation efforts should be focused, and also helps you sense your company's likelihood of wanting to outsource your department.

You must set your budgets and spending practices with the knowledge of their affects on the health of the company.

In justifying expenditures in technology, you should be familiar with fundamental cash flow and financial analysis. Consulting the internal finance people to make sure that you are using the same standards and constants the rest of the company uses is recommended. While not attempting to teach the fine points of financial analysis, it is important to know some of the basic concepts that may help explain the usefulness of these tools.

Payback - Payback analysis is an old standard that gives a company an idea of the time it will take to recoup the initial investment in a project. In order to determine a payback, you will need the amount of the initial outlay, or cost of the project, and the net cash flows that will result.

Let's look at an example. You want to upgrade some disc drives in your data center with some new, high density drives for a cost of $73,000. You must identify actual cash differences that result from your upgrade, such as decreased maintenance costs and decreased electrical and A/C usage. The identifiable cash flows will be $1780 per month lower with the new drives. How long would it take to recouped the cost of the investment?

Payback Analysis

Year *n*	Cash Flow	Residual Value
0	($73,000.00)	($73,000.00)
1	$21,360.00	($51,640.00)
2	$21,360.00	($30,280.00)
3	$21,360.00	($8,920.00)
4	$21,306.00	$12,386.00
	Payback =3.42 yrs	

Table 1

As you look at Table 1 you will find that the annual cash flow is subtracted from the initial investment until it completely wipes out the initial investment, which in this example is 3.42 years. In other words, it will 3.42 years for the company to recoup its investment.

Payback calculations are quick and easy, but they do not take into consideration the time-value of money. $1,780 received three years from now is not as valuable as $1,780 received today. There is a cost involved in parting with the initial $73,000 of your disk drive upgrade. This cost is called the **cost of capital**, which is the required rate of return your company must earn on its investments. The cost of capital has an affect on cash flow analysis. The cost of capital can usually be obtained by someone in your finance department. If the cost of capital in the previous example were 6%, the affect on the payback period is illustrated in Table 2.

Discounted Payback

Year *n*	Cash Flow	Residual Value
0	($73,000.00)	($73,000.00)
1	$20,150.94	($52,849.06)
2	$19,010.32	($33,838.73)
3	$17,934.27	($15,904.46)
4	$16,919.12	$1,014.66
	Payback =3.89 yrs	

Table 2

As can be seen in Table 2 the recovery time is extended to 3.89 years. The consideration of the time value of money is usually termed **discounted cash flow analysis**. Two popular discounted cash flow analysis methods are discussed below.

Net Present Value (NPV) - Net Present Value (NPV) is a capital budgeting concept representing the present value of a project's future cash flows, less the project's initial investment. This helps determine whether, investment return speaking, the company would be better off having undertaken the project in consideration of its required rate of return, or cost of capital. As a general rule, if the NPV of a project is greater than zero, it indicates that the company netted an increase in value over and above its cost of capital and initial outlay.

In order to undertake an NPV analysis, you must determine the life of the expected cash flows. Most spreadsheets will calculate NPV. The NPV of the disk drive upgrade, over 5 years, was $16,015.18, which indicates that the lower cost of operating the new disk drives actually saved the company $16,015.18 in today's dollars as illustrated in Table 3.

Net Present Value

Initial Outlay		$73,000
Cost of Capital		6.00%
Duration of Cash Flows		5 yrs
Annual Cash Flows		$21,360
IRR=	14.19%	
NPV=	$16,015.18	

Table 3

If the NPV is negative, the company does not financially benefit, in terms of an investment, by undertaking the project.

Internal Rate of Return (IRR) - The IRR is a technique that reflects the rate of return of a project at the point where NPV equals zero. The IRR value is compared to the cost of capital to determine if the project is financially beneficial. In our disk drive example, the IRR was 14.19%, which is good rate of return compared to the company's required rate of return of 6%.

Disk drives will often need replacing regardless of their financial attractiveness, but understanding standard analytical methods is very helpful in gauging how your projects and expenditures compare to other projects being proposed within the company. The disk drives may be a necessity, but what about some of the other enhancements and new technologies that you are wanting to undertake? By taking the time to do these analyses, you become more sensitive to expected results in comparison to the initial investments. Obtain a book on financial analysis, and talk to your company's finance people to learn more about how to use these methods.

Be careful of expected savings. Be conservative in your expectations, and stick to hard dollar savings, since those are the ones that will be observed to measure the actual benefit of the project.

COMMUNICATION

Understanding that the term "communication" has been beaten to death, it is important to talk about communication, since it is an essential survival skill.

If the IS department is unappreciated, you must communicate what is being done to benefit the users and the company. No one can appreciate what you are doing for them unless they know what you are doing and how it is a benefit. Accomplishments must be communicated in general terms to which the top management can understand and relate. A commonly understood benefit is cost savings, or increased revenue.

An IS department was proposing the acquisition and implementation of an on-line collections system, but the Board of Directors was hesitant because of the very high price tag. Progress was made only after someone explained to the Board that the system could pay for itself by the prevention of a single account loss. The benefits of the proposal were not appreciated until the benefits were communicated in terms applicable to the Board of Directors.

The survivor must create channels of communication to and from decision makers and top management. Project management systems can be good to communicate the progress and accomplishments of the IS department. There is a difference between project tracking and project management. Identifying tasks and recording their progress is project tracking. Project management is more active, where the progress is not only tracked, but influenced and managed. Project tracking is easy, but project management takes a significant amount of communication, organization, follow-through and prioritization skills.

In managing a project, the project manager must keep track of constraints on resources and make adjustments to priorities as they change. As a changing priority changes the progress of a project, the project manager must be proactive in communicating the impact to the sponsor of the project being affected, and getting the sponsor to cooperate with the change. The project manager must do this, while giving the sponsors a feeling that they are being served. Most people are cooperative if they feel like they have a say in how things are done, otherwise they feel victimized and become distrusting and territorial. As a project manager, you must be sure to communicate to everyone how and why things are getting done the way they are.

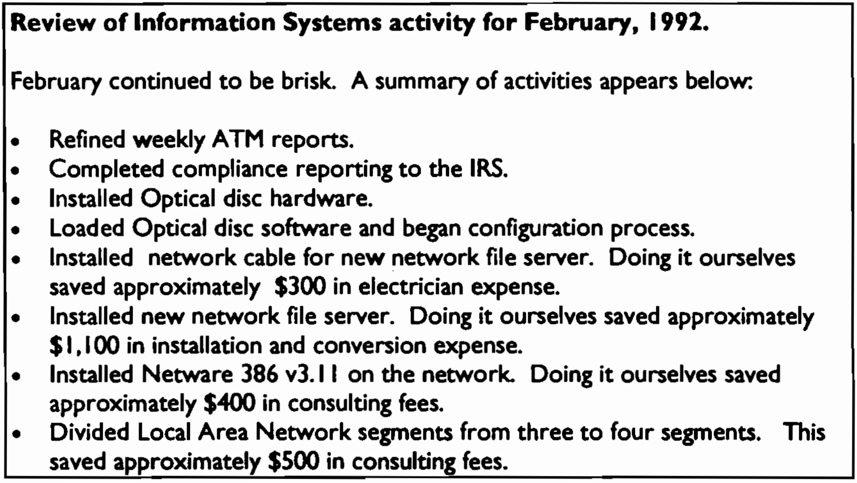
A project management system will allow for the better tracking of your resources. By tracking the resources, you will have a better idea how productive you and your staff are.

In development and problem solving projects, always insist on end-user participation. The end-user knows the processes better than anyone. Ask for help from users, and try not to be defensive when a user is ignorant of system terminology and function. Use this ignorance to your advantage by taking a fresh look at the system from this person's point of view. It is always surprising how much common sense is present in users and absent in systems.

The IS manager cannot simply rely on paper project reports to hope that everything is understood by everybody. The IS manager should personally contact those people in the organization who are relying on the IS department to help them accomplish their own objectives.

Try to establish a monthly executive summary which outlines the activities of the previous month, such as the example seen in Illustration 1.

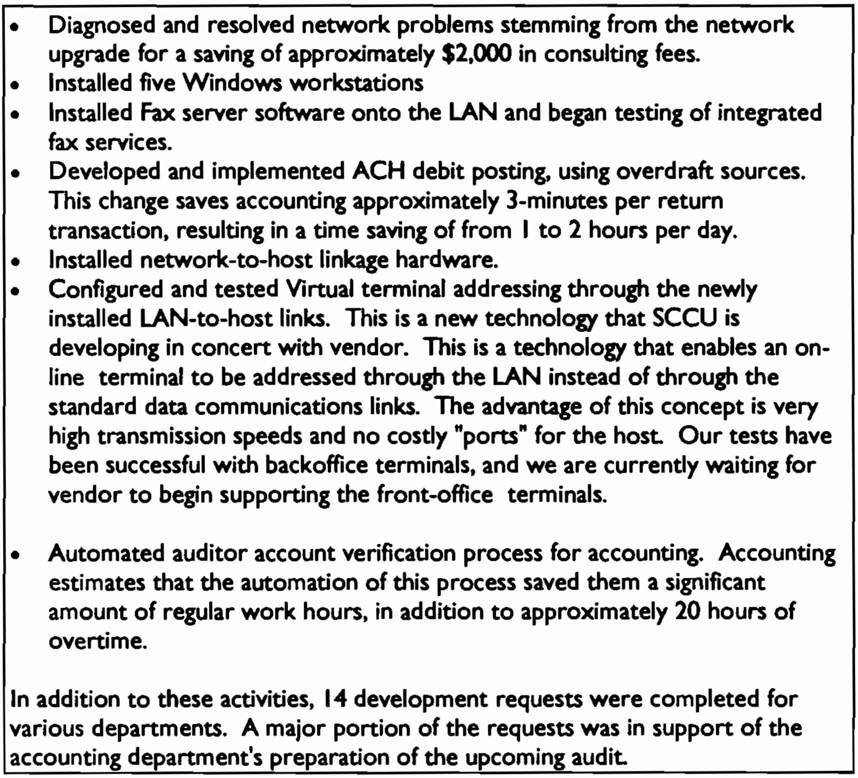
Illustration 1



Review of Information Systems activity for February, 1992.

February continued to be brisk. A summary of activities appears below:

- Refined weekly ATM reports.
- Completed compliance reporting to the IRS.
- Installed Optical disc hardware.
- Loaded Optical disc software and began configuration process.
- Installed network cable for new network file server. Doing it ourselves saved approximately $300 in electrician expense.
- Installed new network file server. Doing it ourselves saved approximately $1,100 in installation and conversion expense.
- Installed Netware 386 v3.11 on the network. Doing it ourselves saved approximately $400 in consulting fees.
- Divided Local Area Network segments from three to four segments. This saved approximately $500 in consulting fees.



- Diagnosed and resolved network problems stemming from the network upgrade for a saving of approximately $2,000 in consulting fees.
- Installed five Windows workstations
- Installed Fax server software onto the LAN and began testing of integrated fax services.
- Developed and implemented ACH debit posting, using overdraft sources. This change saves accounting approximately 3-minutes per return transaction, resulting in a time saving of from 1 to 2 hours per day.
- Installed network-to-host linkage hardware.
- Configured and tested Virtual terminal addressing through the newly installed LAN-to-host links. This is a new technology that SCCU is developing in concert with vendor. This is a technology that enables an on-line terminal to be addressed through the LAN instead of through the standard data communications links. The advantage of this concept is very high transmission speeds and no costly "ports" for the host. Our tests have been successful with backoffice terminals, and we are currently waiting for vendor to begin supporting the front-office terminals.

- Automated auditor account verification process for accounting. Accounting estimates that the automation of this process saved them a significant amount of regular work hours, in addition to approximately 20 hours of overtime.

In addition to these activities, 14 development requests were completed for various departments. A major portion of the requests was in support of the accounting department's preparation of the upcoming audit.

Illustration 1

This summary should be designed to communicate to top management what is being done, how it benefits the company, and how investments are being utilized. Try to communicate the events in a general, summary fashion, and, whenever possible, list quantified benefits. Doing this once a month is a good exercise in communication because it will force you to translate technical issues into general descriptions.

SERVICE

A wise IS manager will provide service better than any outside provider ever could. Service is as much attitude as it is aptitude. A service oriented person may not have the solution at hand, but is willing to get the solution to solve the problem. Service makes the customers feel important and confident that someone will meet their needs or solve their problems. Service is actually more important in the perception of quality than price.

Almost everyone in the company is a customer of the IS department. If you provide exceptional service to your customers, they will not go shopping for another vendor, even if the vendor under prices you. To provide superior service you must concentrate on several areas.

Empathy - This is often called "walking the service path" which means to put yourself in your customer's shoes. The best way to provide service is to understand your customer's perspective. If an order entry clerk is having a problem with a terminal and calls the IS department, it is important for the IS person to understand that the clerk is probably under extreme stress, possibly has a customer holding on the phone and needs some assurance of support. The IS person can view this clerk as a hindrance to his/her work flow, and an interference to his/her day, or can treat the clerk with courtesy and reassurance. The IS person can either view problems as nuisances or as sales opportunities.

Were it not for problems, most service industries would not exist. A customer needing service is not part of the job, it is the reason for existence.

The IS person who views customers as nuisances might put the clerk through all kinds of troubleshooting exercises on the terminal, datacom, cables, etc. The IS person who understands how the clerk may feel will try assist the clerk dealing with the customer on hold by assisting the clerk in transferring the customer to another clerk. The IS person may suggest getting follow-up information from the customer so the clerk can call the customer back after the problem in resolved. After some of the stress is alleviated, then the IS person may walk the clerk through some simple diagnostics. The IS person should express appreciation to the clerk for his/her assistance and also gather some follow-up information to check on the clerk later.

Wants vs. Needs Assessment - In an interview, Peter F. Drucker once talked about MIS management and discussed how most managers "...believe the information system should give them the information they are used to getting rather than the information they need."[2] It is an important skill to discern what information someone needs as opposed to what they request. Better service is possible when you know what the objectives are. Instead of just delivering what is being requested, see if you can dig deeper into what the user is trying to accomplish. Usually the result will be a more effective solution and, more importantly, an increased level of confidence in the requester of the IS department's ability to service his/her needs.

Follow-through - A very powerful attribute of excellent service is follow-through. Unfortunately, follow-through is also very rare. When you tell someone you are going to do something, you should do it. Follow-through does not mean that you do everything right then and there, but rather you communicate when you will have a response, and then do what you say. In the above example with the clerk, the IS person could fix the problem and the clerk may feel like he/she received good service, but when the IS person calls back as a follow-up to make sure that the problem was resolved, the clerk can't help but felt like he/she has received excellent service. The clerk feels that IS really cares about not only whether the system is working but how well the system is working.

STAFFING

Outsourcing agreements are not normally based on savings in equipment costs, since equipment costs have steadily decreased. Staffing costs are probably the biggest cost benefit derived from outsourcing agreements.

Like every other department, IS must pay continual attention to how it is staffed. Are there replacements for key positions? Some companies have felt trapped or blackmailed by IS people who possess exclusive, proprietary knowledge. An outsourcing vendor can alleviate this problem. Do not put your company in a position that depends on any particular individuals to be present.

In many outsourcing agreements, the outsourcing vendor provides staff specifically designated to serve a particular client. In most cases, the vendor uses less staff than the client was using in-house. In these agreements, there are not economies of scale at work, but simply a matter of working more streamlined and efficiently. The wise IS manager will streamline staffing to ensure that the staffing costs are held to a minimum. There is no magic answer for how to streamline, but it is an issue that must be recognized.

One exercise that is helpful is to rank employees. In other words, if you were to curtail your staff, which staff member would be the first to go? This may sound like a cold, callous exercise, but this exercise helps break through the fallacy that all of them are needed. The truth is, some of your staff is better than others and you owe it to yourself to determine this order of importance.

Beware of "glass house" techies who growl in the corner, and enjoy eating users for lunch. The IS manager ends up spending most of his/her time doing damage control. Being rude and non communicative is not a prerequisite for technical prowess. Create job descriptions with service standards spelled out as part of the job.

Each individual staff member should be viewed as a consultant. Performance appraisals should be based on quantitative information in regards to "billable hours" and number of projects completed during the period. IS staff members need to understand that the survival of the department will depend on how well the "customer" is satisfied each time.

CONSULTANTS

It will not always be possible, or sensible, to do everything in-house. The IS manager must either possess or access the resources to support the company with its information needs. A good way to access resources you do not have is to use consultants.

When using consultants, remember that a consult is a "hired gun". The consultant inherently has no loyalty to the organization, but rather to the assigned project. The value you derive from the consultant will be largely determined by how the consultant is directed. A good consultant will analyze a project, provide a proposal and estimate regarding the implementation of the project, and communicate the progress as it develops.

Unfortunately bad consultants are not hard to find. Before hiring a consultant, check his/her references. Good consultants should have numerous, verifiable references. If an organization is happy with the work of a consultant, it will often use the consultant for many projects. Try to verify references at the user level. Sometimes the people that hire and pay the consultants are not in touch with those affected by the consultant's work. There are instances where the consultant appeared to have done the job, but in fact had been inept. The consultant's deficiencies had been obscured by the users who had been correcting his errors. The consultant was very adept at self promotion and appeared to top management to have been effective.

A consultant can be an IS manager's best friend or worst enemy. A good consultant can accomplish amazing things because of specialized expertise, good organizational skills, good communication skills and the opportunity to focus on projects without most of the interruptions that plague regular staff. If the IS manager is proactive in recruiting and utilizing the expert consultant, it will reflect well on the IS manager's foresight and perspective. A consultant can also subvert the manager's credibility if the consultant tries to take credit for accomplishments. This is often done by the consultant going around the manager to higher levels to tout his/her greatness at the manager's expense. Make sure the consultant knows who he/she works for and remain in control.

POLITICS

Many people say, and some actually believe, that they are above politics. Anyone taking such a position will either come to a realization of reality, or become extinct. Nobody is exempt from politics and nobody ever has been. The IS people have historically not been very good at politics, but they wielded significant power so they didn't have to be very good to get by.

It is advisable that you understand and use politics for your survival. Find out how the "grapevine" works and link into it. Again, many see grapevines as gossip, but it takes two to gossip. You don't have to spread gossip to scan the company airwaves for changes and trends.

Create sponsors. If you are dealing with your system users like customers and your customers are well serviced, you are creating sponsors. You also need sponsors in top management wherever possible. It is amazing how some IS people treat managers from other powerful departments like any "user". When these managers want something done, try to give them special accommodation. Take inordinate measures to create sponsors of those influence your funding.

CONCLUSION

The outsourcing of BP's information technology department could have been avoided if it "...were more responsive to the actual business of BP and considered developing innovative alternatives that quickly satisfied the company's needs."[3]

In many situations outsourcing all or part of the information services will make sense. In businesses where information services can enhance the company's core business, the management of the IS system will be the determining factor whether the company outsources.

Technology is changing rapidly and not all new technology is applicable to all organizations. "Downsizing" is a major focus of many IS departments. Downsizing may enhance or threaten the position of the IS department. Whatever changes your organization is undergoing, take a look at your department and try to project where your department will fit in in the next three years. Do you see yourself as an integral player that is helping to drive the company to change or are you fighting for each square foot of territory and authority?

As companies continue to downsize systems and work forces, more and more highly skilled outsourcing contractors will be vying for the available business. In order to survive in the trend to outsource, IS departments will have to continually refine their technical, communication, and business skills.

Pay attention, be prepared, and protect your company as well as yourself.

[1] Kristi Coale, "British Petroleum outsources I.S.," *InfoWorld* (Feb. 3, 1992)
[2] Robert E. Umbaugh,"Peter F. Drucker on MIS Management," *Handbook of MIS Management* (Boston: Aurbach Publishers,1988)
[3] Kristi Coale, "British Petroleum outsources I.S.," *InfoWorld* (Feb. 3, 1992)

COPING WITH THE NEW I.S. REVOLUTION - WHAT I.S. EXECUTIVES NEED TO KNOW

Roger Lawson

Proactive Systems Inc
4 Main Street
Los Altos
CA 94022
Tel: 415-949-9100

CONFUSION REIGNS

This paper was prompted from some thoughts that I had at the Interex San Diego conference. As a former M.I.S manager for a large HP3000 user, it occurred to me that if I was still in that role I would be mighty confused at present. There is an ever growing variety of hardware and software options, and often fewer staff to cope with them in these tough economic times. I feel the typical user of HP equipment may be looking for some advice to help them cope with this "future shock" where new technologies appear more rapidly than people can learn how to use them.

HISTORY ISN'T BUNK

Where are we coming from? For a period of 15 years, Hewlett Packard typically sold the following environment as a commercial multi-user system solution:

```
                    HP 3000
           IMAGE data base managment
      V-PLUS screen handling on dumb terminals
          COBOL language development
                 NS networking
```

Note that they did occasionally flirt with 4GL's (eg TRANSACT), but mainly left this to third party products to supply on a "pick your own" basis. The above package was a very good solution to meet the needs of commercial systems users (and was very successful given the limitations of HPs traditional marketing, sales and distribution approach). The components were generally reliable, high performance and cheaper than some other alternatives (I am referring to both hardware and software here). The standardisation on this "package" over space and time resulted in low cost user systems on two additional grounds:

- ■ Software compatibility was maintained so users didn't have to rewrite systems every few years (HP have done a remarkably good job of maintaining compatibility across hardware platforms for example).

- ■ There was a ready pool of expertise available that could be drawn on (for example, IMAGE is used by over 95% of HP3000 users and it is still probably the most numerous data-base management product installed on mid-range systems).

We *KNOW* we can build high performance, low cost, reliable systems based on these tools. HP (and you as users) shouldn't ignore what can be learned from its history in this market.

FUTURE OPTIONS

What happens if a new prospective customer talks to HP now? The options he is given are:

1. Do you want an HP-UX system or an HP3000? Or to put it another way, what sort of Open Systems compliance are you looking for?

2. Do you want to use SQL (if so, which flavor; ours or someone elses?), or IMAGE?

3. What form of user interface do you want to use; terminals/PCs running V-PLUS, networked PCs, X-Windows, or MS-Windows, etc.

4. What networking environment are you looking at?

5. What's your favorite 4GL or development language?

I guess HP can now claim to provide what every possible customer can possibly want, but a prospective customer can also get very confused. Is this a sensible marketing stratagy for HP? Maybe. Personally, I bought an HP3000 as a "package" that I could see was well integrated and worked, not because I liked the store.

Does the mid-range DP manager really have all the expertise and experience to make an informed judgement between all the options (including planning migration from existing systems)? Probably not! The available software products have increased several fold in the last ten years so that the average computer manager cannot reasonably be expected to know all of them in depth. Let me cover a few of the above options as they affect existing HP commercial users (eg HP3000 users) and give you my personal opinion of what I would do if I was a user manager. I am hopefully relatively unbiased as we use both HP3000 and HP-UX systems, plus IMAGE and HP-SQL in our offices, as well as selling products that run on both platforms.

SHOULD I MOVE TO OPEN SYSTEMS?

My answer here is an unquestionable YES. There are big advantages in going for a standards based approach. It makes it easy to hire expertise, you aren't stuck if your "supporting experts" quit or go out of business and it assists interoperability with other users (although I have noticed this is much more relevant in the U.S.A. than Europe, because companies are typically much bigger in the U.S.A.). After all, one of the reasons I originally bought an HP3000 15 years ago was because it employed an industry standard development language (namely COBOL), whereas most other minicomputers didn't at the time.

Should I buy an HP-UX box for a new application? Maybe if it's the best, most economic package, (but don't assume UNIX systems are cheaper - that's a mirage). That also assumes that the staff retraining and learning time costs are acceptable and I have a large enough staff to retain expertise on two operating systems. Should I drop my HP3000 system now or in the longer term? Probably not. Let me justify these answers.

As has been said by many other people (but is often ignored), open systems don't necessarily mean UNIX, and thank god for that. If you were to design a commercial operating system from scratch, UNIX would not be the result.

It's not user friendly, with lots of meaningless acronyns, a case sensitive command syntax, etc. (commercial systems need to be more user friendly than technical systems because the expertise of users is often lower). It doesn't provide job processing/managment capabilities, has very limited printer spooling and lacks many of the other capabilities commonly used by commercial users. Its security is poor in comparison with most commercial operating systems.

The preferred 3GL is "C", which you really need to know to get the most from a UNIX environment. Personally "C" scares me to death when I look at the maintainability of "C" programs by inexpert programmers. Shareability of code between programmers and maintainability in the long term is much more critical in a commercial environment than it is for technical applications. Incidentally we use "C", COBOL and PASCAL extensively in our company for different purposes, so again I hope I am not too biased. UNIX is also significantly slower, and less cost effective than MPE for transaction processing (ask HP for the figures if you don't believe me).

Now you can turn UNIX into a 'nice' system by adding SQL databases, 4GL development systems, 'shells' or other user front ends such as HP VUE, Spoolers etc. but in that case it certainly won't be an industry standard, non-proprietary system. The 'package' that you end up with is unlikely to have the same (30,000+) number of users as the HP3000 mentioned above. In fact, UNIX is not even standardised with two opposing camps fighting it out. HP is backing OSF-1 which may be a good solution and certainly aspects such as the data management and networking standards (DME, DCE) look to be step forwards, if they work in practice.

From the few user experiences to date (and I think HP has learned the same), I don't think migrating from HP3000 to HP-UX (or other UNIX) is likely to be a very pleasant or productive process.

So where does that leave my commitment to open systems? The answers are:

A: I would select and use industry standards wherever it appears that the standard is, or will become, commonly accepted. Note: these standards could be those set by standards organisations or simply be those standards (such as MS-DOS on PCs) that have become defacto standards.

B: Therefore, I would encourage HP to release POSIX compliance to the lowest level as soon as possible and write my new HP3000 application systems to use it.

C: Also I would move over the next few years to the use of SQL databases (see later for more on this subject) even on an HP3000 system.

D: As there are no standard, non-proprietary, 4GLs I would continue to use 3GLs such as COBOL, PASCAL or "C" (particularly where transaction volumes are high or performance is critical), with 4GLs being used also for certain applications (the chosen 4GL being on the basis of features, price, cross platform availability, support quality etc). The last thing I would want to happen is to get locked into a proprietary 4GL with no control over upgrade license fees. Note: this assumes that you intend to continue working longer term for your company. If you intend to quit or get fired within 2 years you can ignore this advice. Incidentally isn't it about time someone invented a standard 4GL for use with SQL databases?

E: I would go for standards-based networking products (did you notice that DCE will be available on HP3000 systems). In fact HP also intends to provide CURSES (the typical UNIX user interface) on MPE (it already supports X-WINDOWS and OSF MOTIF), support the same MICROFOCUS COBOL as they do on the HP-UX system and provide other UNIX capabilites on MPE. There is already object code compatibility at the link module level between MPE/XL and HP-UX.

CONCLUSION:The renaming of MPE/XL as MPE/iX is not just meaningless hype, it will soon be a reality. My problem of choosing between HP3000 or HP-UX systems as my open system platform goes away. I can get the best of both worlds with the disadvantages of neither! I can move my HP3000 application to an Open System based UNIX compatible operating platform (called MPE/iX) without doing anything at all, apart from changing the badge on the machine!

IMAGE VERSUS HP-SQL

As a long time user and lover of IMAGE, why should I advocate looking at SQL? Because SQL is truly an industry standard, will provide cross platform interoperability once the SQL ACCESS standard is provided by all data-base vendors, provides a better base for client/server or distributed database systems, and is known by every college student. It may be limited in many areas, have an archaic command interface language, have difficult performance characteristics on some applications, but it's rapidly becoming the same kind of standard as COBOL did for commercial development. Neither HP nor HP users can afford to stick their heads in the sand and ignore it.

What I am NOT saying is that existing HP3000 users should stop using IMAGE. Converting existing applications from IMAGE to HPSQL is technically tricky (although there are some tools that may help). You can't easily translate or emulate a network data-base in SQL. In practice I think it's best to convert to SQL only when an application is rewritten or a new application developed unless you have very good reasons to try a migration. Neither would I recommend HPSQL to smaller HP3000 users if they are already using IMAGE. Why bother learning two data base systems and supporting both? (A lot of users don't even know much about IMAGE and SQL databases are, if anything, more complex to optimise and maintain). Span of expertise is a limiting factor for todays DP departments and they neither have the time nor the capacity to learn everything about everything. With IMAGE the performance characteristics are probably more predictable and better known, so any real time transaction processing systems may be better in IMAGE at present.

If HP continue to develop and maintain IMAGE (as they have committed to do), there is not much point in ceasing to use IMAGE. Every study I have ever seen says that unless there is a good financial incentive for purchasers to move to a new technology, nobody ever makes a success of a new product in the commercial field (good managers do not behave like kids in a toy store apparently). That's why HP3000 users haven't rushed to move to HPSQL. Therefore take your time and pick the right opportunity to learn more about SQL and implement its use.

Incidentally if you a VAR there is no contest. To get access to the largest market you need POSIX compliant applications based on SQL databases, even if you wish to continue to push HP3000 based solutions because of its reliable, high performance and friendly operating system.

As a company we have decided to support both IMAGE and HPSQL users in our database management products. For the reasons given above we see both systems in use for the forseeable future (and on the same computers in many cases). Whose flavor of SQL would I use? Probably HPs ALLBASE on HP3000 or HP-UX because I believe it is significantly faster than others, but it may depend on which 4GL you like.

CLIENT/SERVER SYSTEMS AND USER INTERFACES

If you look at the existing HP3000 installed base there is a mix of dumb terminals and PCs on users desks (more of the former than you may think according to our recent customer survey). The PCs are probably running ADVANCE LINK, REFLECTION or other terminal emulation software. However only 40% of them are likly to be capable of running MS-WINDOWS efficiently (ie 386s or high end 286s) (see ref 1.). On HP-UX commercial systems I suspect it's a mix of dumb terminals/PCs with a sprinkling of X-Windows users.

I think we need to clearly differentiate at this point between "knowledge workers" (ie managers, researchers, secretaries) and others (accounts clerks, order-entry operators, etc). The former need to retrieve information from a range of different data-bases and collate it. They need to manipulate the data and massage it with spreadsheets and word-processing software; and present the results with graphics or DTP products. The non-knowledge workers are typically entering data, or processing transactions, in dedicated application environments. Nobody seems to have pointed out that the needs, skills, learning capabilities and the justifiable training investment in the two groups differ substantially.

What is a windows-based GUI on a 386 PC or workstation running in client/server mode really going to do for an order entry clerk? Very little in my opinion, except add several thousand dollars a year to the hardware/software cost to support that user. They may find it easier to get on-line help information if they get stuck, but I see very little other advantage. The alleged benefit of relieving the central CPU of some processer workload is negligible for most large multi-user commercial systems where the bottleneck is usually the main data-base software or disk I/O systems. However knowledge workers are a different matter. Their salaries are higher for a start so the incremental cost is less significant.

Coming back to the non-knowledge workers, their typical user interface at present is a V-PLUS screen. This is HP3000 specific of course and if you intend to stay with HP, that's no problem. However you may want to look at a more cross platform solution such as WINGSPAN or JAM (the former even provides a V-PLUS emulator I understand). With V-PLUS you can even move partly to client/server architecture by using V-PLUS/WINDOWS.

For knowledge workers, I would look at providing an MS-WINDOWS interface (or even NEWWAVE which runs over MS-WINDOWS). It's already a de-facto standard on PCs and it's easier, I am told, to program than X-WINDOWS. It also provides less overhead on the back-end main systems in the scenario where the HP3000 or HP-UX system is acting as the main transaction processing engine.

By providing a PC LAN network using NOVELL or LAN MANAGER, you can get the best of all worlds. You can run your HP3000 or HP-UX systems as data-base servers and network servers. You get the advantage of reliable, automatically backed up systems, with shareable data while off-loading the cpu intensive applications to the users PCs.

HP also intends to support several PC, MS-WINDOWS based, SQL development products in client/server mode to HP ALLBASE/SQL such as GUPTA's SQL-WINDOWS, POWERBUILDER etc. As noted elsewhere, HP is currently sitting on the technological fence and not packaging a solution for you (who was it that said HP had become market driven?). These may well provide an easy way to replace the V-PLUS/COBOL/ IMAGE combination. Whether they can really provide high performance, complex application support to replace V-PLUS/COBOL/IMAGE remains to be seen in practice. The vendor demonstration I have seen have not been promising but I think they are worth a look. HP is also releasing a product to make it easy for user written PC applications to access HP-SQL data-bases in client/server mode. Note that various traditional 4GLs such as POWERHOUSE and INGRES/WINDOWS support X-WINDOWS or MS-WINDOWS user interfaces also.

End users are unlikely to justify the cost of developing WINDOWS based applications directly, but third party venders should be able to, particularly those selling generalised tools type products such as report writers.

IS THERE STILL A ROLE FOR MINICOMPUTERS

If I am going to purchase powerful PCs and a PC LAN network, why not go one step further and throw away the central mini, ie, replace it by a PC data-base server? For several reasons I would say no:

A: Because the cost of moving your data-base and existing applications to the PC platform is very high.

B: Because the reliability and quality of such systems (eg in terms of data recoverability) does not seem so high as existing HP minis.

C: Because their architecture is not optimised for high volume transaction processing as minicomputers have been over the last 20 years.

D: Because the operating system features on minicomputers are much richer than on PCs.

I believe it's really still only a viable solution at the low-end (with a limited growth path) and even then the cost advantage versus a minicomputer system may not be there. However it is obvious that minicomputers are tending to drift up the power range and compete more against mainframe systems. Both PC makers and minicomputer manufacturers are talking about downsizing so as to attack a market where they see higher margins and less competition, while ignoring the overhead cost and skill sets required to do business in those markets. If the current recession ends quickly, I think the hype will disappear just as rapidly.

WHAT WOULD I DO AS REGARDS CLIENT/SERVER AND GUIs

I think MS-WINDOWS is a de-facto standard for PC software, but as regards client/server, I would do what HP is doing. As there is no clear technology/market leader with a significant installed base to give me confidence, I'll sit on the fence until there is. The VARs and software companies like Proactive Systems are going to do my work for me sooner or later by pioneering these new technologies. Only if I worked for a user with a big budget would I risk any great investment in replacing existing methods and systems, even for new developments, and certainly not for strategic business ones. However I would gradually move to SQL data-bases from IMAGE as that at least gets me in the likely direction of future trends (and enables me to separate the database from the application development language). It's a pity that HP is not leading the way by recommending (and marketing forcefully) what it thinks is the best package of solutions for common user needs.

CONCLUSION

This was a long paper and I didn't have space to include all the supporting information and evidence to back up my ideas. I hardly touched on the networking area where HP also seem to be backing several horses at the same time. For further information, suggested reading matter is given in the list of references at the end of the paper. You may not agree with all that I have said but I hope it has stimulated some thought. When you next sit back and contemplate where data processing will be in your company in two years time, I hope my ideas will help you.

If you got the impression that I believe Hewlett Packard have got their basic strategy right for the future (as best as anyone can when the prediction of future technology trends is involved), then you're right. It will put HP back into the mainstream of commercial computing instead of the backwater it drifted into during the early 1980s.

REF1: Computer Intelligence Dec 91 Report.

REF2: HP3000 Open Systems White Paper. HP part no. 5952-2490.

REF3: POSIX and the HP3000. George Stachnik, June 91, Interact.

REF4: POSIX Interfaces To MPE/XL. J.Vance and D.Conner, Oct 91 Interact.

REF5: A Standard Operating System Inteface for MPE/XL, R.Lalwani Interex San Diego Conf Proceedings 1991.

REF6: Client Server System Design, S.Palmer Interex San Diego Conf Proceedings 1991.

REF7: HP Motif XL: The X-WINDOW System on MPE/XL, S.Cressler. Interex San Diego Conf Proceedings 1991.

REF8: The ALLBASE/SQL RDBMS "Optimised for HP platforms!", R.Nagar. Interex San Diego Conf Proceedings 1991.

REF9: Open Systems-Where They Came From, What They Are and Why You Should Care, R.O'Donnell Sept/Oct 91 Supergroup Magazine.

REF10: HP3000 Open CASE, P.Petigura Interex San Diego Conf Proceedings 1991.

REF11: Windows- When the Time is Right, R.Bradford Interex San Diego Conf Proceedings 1991.

REF12: Developing Client/Server Applications in HP System Environments using Industry Standard APIs, S.Safe Interex San Diego Conf Proceedings 1991.

$pending to $ave on $oftware

presented by

Shelby Robert
General Manager
Proactive Systems
Four Main Street, Suite 101
Los Altos, CA 94022
(415)949-9100

A. Introduction

As this paper is written, things look pretty grim for the economy. Many organizations are facing budget reductions and cutbacks in spending that are unprecedented in the memory of an entire generation of MIS people. Times are tough and so are the managers who control the pursestrings.

Still there are times when a purchase of software really will save the company real dollars in terms of reduced costs, or in terms of increased productivity. So the question facing the MIS manager becomes "How can I demonstrate to my management that spending a few dollars now will help my company to do better in these troubled times?"

This is the question that must be answered after you have completed a thorough technical evaluation. The product has been selected. You're satisfied that you have the *right stuff* in hand, but now it has to be justified to top management. It is necessary to convince them that the decision to purchase now is the best thing that the company can do to move itself forward.

In the end this paper is about developing numbers that support your decision to move forward with a purchase. Now we approach a different task. The job is measuring the relative value of making this purchase as opposed to not making it. How positive will be the effect on the company of making this purchase?

B. The VALUE Equation

The real question to be answered in looking at acquiring software is *What value will my company derive out of this purchase?* If the answer is nothing, then we might as well stop right here. If, on the other hand, you perceive that there is real value for you and your company, the balance of this paper is designed to help you measure that value.

So how do you measure value? One definition that seems to hold up pretty well is:

VALUE = BENEFITS - COST

We will refer to this as the "Value Equation." It is obvious that as measured benefits rise, and as the costs decline, the value of the purchase to your company increases. The greater the difference you can show between the benefits of acquiring an item and its cost, the more convincing and persuasive your presentations will be. Because the price of your proposed acquisition is fairly obvious at this stage, the real key to understanding the power of the Value Equation is in measuring benefits.

C. Identifying Benefit Areas

Measurable dollar benefits for the MIS department generally come in one of four basic varieties:

a. Reduction of materials costs
b. Increased MIS Staff productivity
c. Increased User Staff productivity
d. Reduced space/power/support services consumption

Individual software purchases will not necessarily find benefits in each of these areas, but most of the major measurable benefits will fall into one of them. At this point, an example may help to illustrate these issues.

One of the issues facing many MIS departments today is the question of replacing preprinted forms with electronic ones. Let's examine some benefits of this way of doing business that touch on each of the four areas mentioned above.

So that we are on some common ground, electronic forms are forms that are printed at the same time as your data on a LaserJet printer. The obvious advantage is that there is no need to have a supply of preprinted forms on hand in order to print such things as invoices, purchase orders, packing lists, and other documents. They simply print on plain white paper, including your logo, with terms and conditions on the back, at the selected printer, and with the appropriate number of copies. All of this sounds great, but what are the measurable benefits here?

Benefit #1. Cost of the forms.

The first and most obvious benefit is that the cost of the preprinted form goes away. But this paper is about measuring things, so let's play with some numbers. Suppose you run a manufacturing company's MIS department and each working day you print 500 sets of four-part invoices that cost you $0.20 per set. At 250 working days per year, your annual consumption is 125,000 for a total annual cost of $25,000.

Unfortunately, there are some materials costs associated with LaserJet printing which must also be factored in here. The toner cost per face is about $0.015, and since our invoice requires terms and conditions on the back of the original, we will have five faces to print. In addition, the paper itself costs about $0.004 cents per sheet, and we need four sheets. So our total costs to produce the same form as above are:

125,000 * (($0.004 * 4) + ($0.015 * 5)) = $11,375.00

To determine the real measurable benefit to the company, we now subtract the old costs without our software acquisition from the projected new costs with the software in place to find the dollar amount of the benefit.

$25,000.00 - $11,375.00 = **$13,625.00 per year**

So far in examining this benefit we have looked at replacing only a single form. If you were actually contemplating such a purchase you would list the savings to the company for each of them. The formulas remain the same, only the names are changed to protect the expensive. So let's explore one more just for good measure.

Another form that you probably deal with is a monthly statement. Let's assume that our fictional company has 1,000 active customers, and that the statement is in two parts, one for the customer, one for the Accounts Receivable department. Using the same numbers as above, the preprinted set will cost about $0.10 per set. Printing 12,000 per year results in an annual paper cost of $1,200.00.

Again using the same costs as above, two sheets of paper at $0.004 each and three printed faces with a toncr cost of $0.015 per face yields a set cost of $0.053 per set. At an annual cost of $636.00, this is still a savings of $0.047 per set or $564.00 over the 12,000 printed in a year.

$1,200.00 - $636.00 = **$564.00 per year**

Just a word here about presentation. All of us like to have information about which we are being asked to make decisions presented in an easy to follow, understandable way. Although the supporting details are important to reveal thoroughness in the discovery process, the major points are the ones that need to be highlighted. It is nice for the reviewer to have available all of the details, but he will appreciate a presentation that helps the important numbers jump out so that his time and energy are saved the task of extricating them from the "Spreadsheet Sea." One of the ways to improve presentations is to tabulate results.

If you have several forms to deal with, you might want to summarize in a format that looks somcthing like this:

Form Name	Sheets per Set	Savings per Set	Annual Usage	Annual Savings
Invoices	4	$0.109	125,000	$13,625.00
Statements	2	$0.047	12,000	$564.00
			Total Annual Savings for Benefit #1 =-->	**$14,189.00**

Benefit #2. MIS Staff Overhead Reductions

The MIS staff have a built-in burden in handling pre-printed forms. When forms printing is needed, the forms must be mounted on the printer, printed, decollated, burst, and distributed. This is apart from a weekly invcntory, and the involved process of reordering when quantities run low. Consider the following numbers for our sample manufacturing company:

Operations time (daily):

a. Printing forms	1 hour
b. Decollate / Burst	1 hour
c. Distribution	1 hour

A total of three hours per day is 15 hours per week times 52 weeks totals to 780 hours per year. Plus additional overhead of one hour per week for the weekly inventory. This leaves us with a total MIS operations overhead of 832 hours. There is a lot of training and investment in MIS operations people. The cost rate for them is relatively high in comparison to office or plant workers. A rate of $50.00 is not unreasonable in figuring total costs for the skill level and responsibility required of these folks.

All of the above overhead in the MIS department can go away entirely with electronic forms, so all of the costs are either direct savings, or opportunities for MIS staff to be productive doing other things. Either way the benefit to the company is:

832 hours * $50.00 = $41,600.00

Every time a form must be reordered, it is reviewed by management. If not only by MIS management, then by other managers in the company. The review must be painstaking because reorder points are infrequent and volume purchases, necessary to get the unit price reasonable, are still quite expensive. To get away with reviewing a form in four hours seems quite reasonable. The people who must do the final review are sophisticated management people. Their hourly cost is reasonably in the $100.00 per hour range. So with 20 forms to review per year our current cost equation is:

20 forms * 4 hours per form * $100.00 per hour = $8,000.00 per year

Because electronic forms can be changed on a moment's notice, lengthy regular periodic reviews are not required. Some overhead will still be necessary, but reductions of 75% are not unusual. This still leaves us with a cost savings of $6,000.00.

The total of reductions in MIS overhead is now the combination of the two numbers from above.

Benefit #2 = $41,600.00 + $6,000.00 = $47,600.00

Benefit #3. Increase in User Productivity

No MIS department survives in a vacuum. MIS embodies the very definition of a services business. We have no reason to exist without our users. To go one step further, it is our users who are doing the business of the company, not those of us in MIS. If we can contribute to the productivity of our users, we will have a direct beneficial effect on the company's ability to do profitable business. One of the ways that we do that is by reducing the time it takes us to deliver information to our users.

In using pre-printed forms, we wait and batch things like invoices until we have an opportunity to mount them on the printer. If the picking list is the fourth copy of that invoice, the goods can't go out to the customer until the invoice is printed. The third copy of the invoice must be carried to accounting for their needs, and the original and remittance advice will go to the mail room to be posted.

With the introduction of low cost LaserJet printers and the availability of software to drive them, we can now print the original and remittance advice in the mail room, the picking list in the warehouse where it is needed, and the accounting copy in the accounting department, all without any delays. The overall result is that needed information arrives at its destination where it can be used in a more timely fashion. Productivity will rise due to increased efficiency of delivery of information.

To illustrate how powerful productivity gains can be, let's assume that our manufacturing company has five people in accounting and ten in the warehouse. If we can realize a modest 1% productivity increase for those people by improving information delivery, we will save 6 hours of labor per week that can be used for accomplishing other tasks.

At $20.00 per hour average cost to the company to have an employee on staff and working, the annual productivity gain to the company is:

Benefit #3 = $20.00 per hour * 6 hours per week * 52 weeks = **$6,240.00**

Benefit #4. Reduction of Space

The 20 forms that we have been using must be individually stored and inventoried each week. They use floor space in the computer room for the working store. In the storage area which is secured, they use additional space.

Computer room space is expensive at about $10.00/sqft per month. At only one square foot per form, the cost is $200.00 per month, just to store forms. In the storage area the cost of $3.00/sqft per month is less, but each form will require an average of 10 square feet. This amounts to $30.00 per form per month or $600.00 per month in the storage area.

Again, the total will not be reclaimed, but at 75% savings (because we now will stock only white paper) we will reclaim $600.00 per month of floorspace. The annual total is:

Benefit #4 = $600.00 per month * 12 months = $7,200.00

D. The Bottom Line

The benefits listed above are not an exhaustive list by any stretch, but they serve to illustrate how to identify and measure the benefits that will accrue to your company by making such an investment. Now we can put it all together for your management.

First, a brief review of the benefits that we have identified and their dollar value to the company.

Benefit #1. Direct Forms Cost Savings	$14,189.00
Benefit #2. Increased MIS Productivity	$47,600.00
Benefit #3. Increased User Productivity	$6,240.00
Benefit #4. Reduced Space Requirements	$7,200.00
Total Annual Benefits	**$75,229.00**

All of this does assume the purchase of a software package to do all of these marvelous things, and the purchase of some LaserJet printers to do the work. The discussions above have mentioned a printer in the warehouse, a printer in the accounting department, and one in the mailroom. MIS will of course have to have a printer to do their testing. Assuming that all of the printers purchased were LaserJet IIIDs, the total investment in printers would be about $8,000.00. The software to do the job above will cost an average company about $7,000.00.

These two numbers represent the total capital investment for the job. The problem is that comparing the $15,000.00 total of these two investments to our benefits list is like comparing apples and oranges, since our benefits are expressed as annual amounts, but the benefit from the investments will last longer than one year.

The best way to evaluate the viability of a capital investment is to use a discounted cash flow methodology. A discussion of this method is a little beyond the scope of this paper. Another, simpler approach is to look at the amortized cost over the life of the investment and the payback period.

Capital investments must be amortized over their "useful life." The tax man defines useful life for computing investments as being five years. What this means is that only one fifth of the cost of the hardware and software should be taken as an expense in any single year. In our example, this represents an annual capital cost for our project of only $3,000.00.

Notice also that the benefit numbers above are all annual numbers. That means that these benefits will accrue to the company each year of use. So now a look at our bottom line numbers:

Total Annual Benefits	**$74,665.00**
Total Annual Capital Cost	**$3,000**

All of the things we have mentioned above are important. They are the essence of what must be communicated to your management, but so far they are only numbers. For many people, numbers don't have a life of their own. They just sit there. In reality, numbers by themselves are just plain

`Boring!!!`

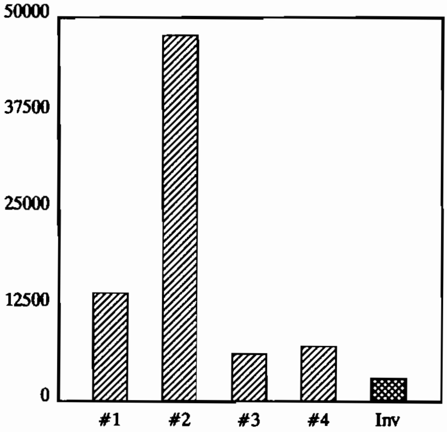
For you as an MIS Manager, this means that you have to do something to get your management's

ATTENTION!!!

How can you do that? If the problem is that all you have to work with is numbers, why not paint your management a picture of what the numbers mean. Pictures of numbers in graphic form convey meaning instantly, intuitively, without having to do the mental work of internalizing the specific value of each benefit. By painting management a picture, you save time, help them to make a better (as well as faster) decision, and make it easier for them to accept your arguments in the process. In the process you don't do any harm to your own reputation for professionalism.

What might a picture of our results look like? Let's take a look at the effect of illustrating the relative value of each of our benefits and our annual investment.

Individual Benefits vs. Investment
Annual Amounts



By looking at the above chart, it is obvious immediately that any one of our four benefits would be sufficient to justify the purchase. The combination of the four is an obvious and overwhelming argument.

Let's go back now and take another look at the VALUE EQUATION. The VALUE EQUATION above stated that VALUE = BENEFITS - COSTS. Solving the equation with the numbers we have developed here yields an ANNUAL VALUE to the company of $69,865.00. Over the five year life the tax man requires, this investment will return $349,325.00. Even the most callous of CFOs will take note of a return like that.

E. Time to Return

One of the calculations that you will want to make has to do with the time to return the investment you are asking your company to make. The easiest way to do this is on a monthly basis. First, divide the annual benefit by 12 to find the monthly benefit.

$75,229 / 12 = $6,269

Now divide the total investment by the monthly benefit to find the number of months it will take to recover the total invested.

$15,000 / $6269 = 2.39 months

Again a very favorable number. This company cannot afford to wait another day for this purchase.

F. Summary

In the preceding pages we have examined only one software purchase, that of a LaserJet printing package. We have certainly not examined all of the benefits that might be associated with such a purchase. What we have examined is a methodology for quantifying the benefits to be derived by a software purchase.

The basic benefits that your company will derive from a software purchase will generally come from one of the four sources mentioned above: reduced materials cost, increased MIS staff productivity, increased user staff productivity, or reduced consumption of other company resources. In order to justify such a purchase to management, the individual benefits need to be itemized, and quantified into dollar savings or productivity gains.

Once the benefits are identified, quantified, and valued, a little care and attention to the presentation of the information goes a long way with management. If you are convinced that this is the right decision for your company, it is worth spending a little time to make it easy for your management to understand your position.

The numbers that you generate in preparing a cost justification may yield totals that seem unreal. Your benefit amount may be dismissed as not being a hard dollar savings. The reason is that your company is probably not going to fire anyone to reduce their costs as a direct result of this sofware purchase. Nevertheless, the arguments for the benefits still hold because the time saved can be put into more productive work.

The important issue in justifying software is that the benefits to your company must be itemized and valued in order to demonstrate to your management that the acquisition you propose is the right thing for the company to do, and that this is the right time to take action. By going through the exercise of identifying and quantifying the benefits of your purchases, you will strengthen your bargaining position and your case for the purchases that you need to make to help your company function better.

LaserJet is a trademark of Hewlett-Packard Company

This article was typeset using FANTASIA from Proactive Systems.

Paper # 3026

Managing Development Projects for New Technologies

Suzanne Harmon
Information Systems Consultant
950 Graham Hill Rd
Santa Cruz, Ca 95060 USA
(408) 459-0802

Project management is always a challenge at best and can be a disaster at worst. As more and more Information Systems groups move to fourth generation development environments, they are finding that their old project management strategies are no longer effective, and yet they are not equiped with new methodologies to replace them. Many new techniques such as entity relationship modelling, data flow diagrams, and CASE management provide tools for design in these new environments, but little or no attention is given to the new management techniques required to optimize the benefits of moving to this new technology.

The Estimate

The foundation of any successful project is a good up front estimate, and I find it impossible to successfully manage a project without one. With a 4GL project, however, there is a significant effort which must precede the estimating process.

A requirements analysis phase must be done which at a minimum includes a rudimentary entity analysis. The results of the requirements analysis help to define the scope of the project. The primary maintenance and inquiry areas, with as much substructuring as possible, are then laid out. For example, Customer Maintenance is a primary maintenance/inquiry area, supported by substructures such as ship to information, bill-to information, contact information, etc.. This is followed by a similar process of enumerating batch processing which will be needed by the system, such as G/L interface, and reporting, both operational and management. When all on-line processes, batch processes and reporting have been identified to the best of ones ability, estimates can be assigned to each entry or "task."

This is the first place where estimating typically breaks down. Estimates are made based on what someone thinks it will take to code the screen, process or report in question. I triple that number. The first third of my estimate is indeed how long it will take to code the task at hand. The second third is how long it will take to keep changing it until it works perfectly and everyone agrees it's done. The final third is reserved for all the changes which will be made and time which will be spent after the users start actually using the delivered product, in training and production, a cost which is almost never included in estimating, but which is almost always considered by management to be part of the cost to complete the project.

The next place where estimating usually breaks down is in two areas which are rarely included in estimates. First is the cost of data base administration. Whether you have a dedicated DBA, or whether everyone does some of it, it is a high overhead item. Usually, you will want to tack on a load factor of at least 5% for this task. Second, and most important, is the cost of project management, overhead, meetings, etc. Include a load factor of 20% for this task.

Your final estimate should look something like this:

Customer Maintenance Screen	40 hours
Ship to Maintenance	20 hours
Bill to Maintenance	20 hours
Contact Maintenance	20 hours
Table Maintenance	40 hours
Customer Archiving Process	40 hours

Customer Alpha List	30 hours
Conversion	20 hours
Subtotal	230 hours
DBA Overhead	12 hours
Training	12 hours
Project Management Overhead	51 hours
Project Total	305 hours

It is imperative that your estimate be broken down into "task" level items. Otherwise, managing the project will break down over time.

You will notice that there is no design time built into this estimate. There is a reason for this. In very large projects, or projects which lack sufficient information or definition to begin setting up a prototype once the requirements analysis phase is complete, it is best to plan a design phase which has its own estimate. In a very large project, I will typically try to reserve a two to three month window at the beginning to interview users, brainstorm design, and begin to set up the dictionary data structures. This is usually best done with two people, preferably two people who complement each other rather than two people who think similarly. Once the design phase is complete, it is appropriate to redo the original estimate and task list to reflect new thinking about what the system will really look like. In smaller or well defined projects, there is sufficient "design" time built into the estimates for a prototyping methodology, that a separate estimate for design is not typically needed.

The Plan

Once you have the estimate, you can make the plan. The plan outlines milestones, dates when things will happen, what will happen, and how many of the project hours will be used to get there. Milestones should never be more than two to four weeks apart, depending on the total scope of the project. I usually like to have user demos be my milestone activities, culminating in training and implementation dates.

The plan is created by deciding what the implementation date is and working backwards. Everything, of course depends on the overall size of the project. Leaving a month at the end for training and implementation makes sense for a 10,000 hour project but not for a 1000 hour project. Similarly, letting several weeks elapse between user demos (formal - the informal ones are going on all the time) makes sense in a 10,000 hour project but it would be a disaster to go more than a couple of weeks in a 1000 hour project. Depending on the amount of material deliverable at each demo, the developer deadline and project team walkthru date should be two to six working days before the user demo, to allow final debugging, changes and polishing identified at the preceeding team walkthru. Once you have developed the plan, it may become obvious that you are going to either have to add more people to the project or extend the implementation date.

Now let's discuss the mechanics of developing the plan. We will use the little customer maintenance project estimated above. The system is due July 1st. I can't possibly get approval and get started before May 15th. I have to leave the week before July 1st for training and implementation. That gives me about 6 weeks before July 1st and 1 week after (post implementation support of problems which is really spread over more than one week but is only part time). So seven weeks to get 305 hours of work completed. I figure each person on average can't be estimated at more than 30 hours per week. So this works out to about exactly one and one-half people over 7 weeks to complete. That's about perfect, because I had planned to have one full time developer and one half time project manager/DBA/developer. So, in this case I don't have to add or take away people from the project, or move the delivery date. My plan will look something like this:

5/29 - Team Walkthru of 6/2 demo
6/2 - First User demo - Review preliminary prototype of all on-line deliverables
6/9 - Team Walkthru of 6/11 demo
6/11 - Second User demo - Review fully functional prototype of all on-line deliverables, review preliminary report output
6/17 - Team Walkthru of 6/19 demo
6/19 - Final User demo - Review of "finished" system
6/23 - Team Walkthru of final changes requested 6/19 and prep for training
6/25 - Training commences
7/1 - System in production (There should still be at least 50 hours left for support at this point)

This project is too short and too small to track the number of hours I expect to use to get to each demo. In a large system, however, I would definitely want that information.

The crucial point here is that dates never slip. Once the project plan is established it is published for users, management and team participants. Slipping dates is not an option.

The Development

The hardest part of transitioning from 3GL projects to 4GL projects, is getting people to let go of their individual egos and create a team ego. The key to successful 4GL development projects is the team dynamic. Every project team which consists of more than one person is a combination of people who are not all equal in their skills, their competence, or their productivity. The individual skills or lack thereof are only important as they affect the composite performance of the team. People who insist on maintaining their individual ego as a member of the team get in the way of the team's ability to acheive its optimum dynamic.

The entire team meets weekly, at the same time every week. It is at these meetings that assignments are made by the project manager which will insure that the next milestone is met. Assignments are only rarely made which do not directly bear on the upcoming milestone. In this way, the entire team keeps focused on the goal. As assignments are handed out, taking into consideration peoples skills and productivity, the individual receiving the assignment needs to be queried to insure that they feel they can meet the commitment being asked of them - to have this work being assigned completed to a specified degree in a specified timeframe. If people make commitments that they cannot realistically meet, they are letting down the whole team. This is in fact a major learning process for most team members unfamiliar with this style of management, because most people are accustomed to overcommitting and really have no true idea of what they can, in fact, accomplish over a given period of time. Ideally, the amount assigned to each person is never as much or more than they can accomplish, for two reasons. The first is that people need to be successful and not get into habits of not completing what is expected of them. From a project perspective, the second reason is the most important, which is to insure that people have room in their schedules to add unexpected tasks that come up or to complete work, assigned to another person, which is behind schedule.

At the team meeting each member reports on the status of the work they have committed to complete at the previous weeks meeting. If it becomes apparent that, for any reason, a member of the team is falling behind or having difficulty completing their commitments, this is a reason for concern and action. The action may be to rearrange assignments, possibly offloading some of that person's commitment. The concern should be for the underlying cause and the effect it has on the team morale. The rearranging of assignments and responsibilities is essential to maintaining the kind of fluid, goal oriented energy that the must continue throughout the project. For every team member to know that the goal is the teams success, and therefore the project's success, puts a burden on them to not let the team down but at the same time guarantees them that the team will not let them down and that they need only reach out to obtain the kind of support they need, without penalty. Every once in a while, a team member will come along who wants to work the system. They never want to do any work and are only concerned with their own comfort and convenience. The team quickly recognizes this type and naturally resents them. You really have two choices, you can get them off the project, or if that's impossible, you can pigeon hole them with useless assignments that no one cares about and don't really need to be done.

At the team walkthrus and meetings, all team members are encouraged to participate equally in designing and critiquing the system as it evolves. This gives each team member a real sense of ownership of what is evolving, while at the same time preparing them to take over work in any area if or when it becomes necessary. It also results in a far better system because many different perspectives and backgrounds come together to suggest the best possible solution.

The weekly team meetings are essential. Other project managers who I have trained in this methodology have sometimes thought they could short cut this by meeting with the individual team members, or smaller subsets of team members, to avoid the high cost of such meetings. It never works, and none of the projects managed that way have come out "successfully" by my measurement of success - on time and on budget and meeting the user requirements. Bypassing these meetings destroys the team dynamic, and in my opinion and experience also destroys the project's ability to be successful.

User demos are an essential way of securing the users' participation as team members. Demoing to users one on one is always helpful and necessary. Demoing to users in a group gets the users to communicate and agree as a group, an almost impossible task otherwise, even within the same department. A user from each major department even peripherally affected by the system should be at these demos, but at the same time the number should be kept manageable - ideally six and never more than ten. A user can either share with the group or keep internalized whatever reaction he or she is having to the direction the project is taking with design, training and implementation. If it is shared, it helps the group focus on issues and resolve them. If is is internalized, the user must later explain to the group why their issues weren't brought up earlier, if they dare.

One major issue is whether developers should be in the demos or not. I favor having a user liaison or "project ambassador" who knows the system inside and out demo the system and give feedback to the developers. This may be the training person discussed below, or it may be one of the developers. If it is a developer it must be someone who has a very special temperament. The problem with having developers in demos is they start explaining why they designed things a certain way and it usually ends up evolving into them arguing for their design while the user argues for what they want. These demos are for listening to the user reactions and issues, not arguing solutions, unless the users want to argue among themselves. The advantage of having a user liaison who is a developer, is that this person can also have the role of resolving design issues which come up. It tends to be counter productive to have each developer handling their own design "loose ends" with the users. This is because of a variety of factors. Not all developers communicate well with users. Many times two or more developers will have unresolved issues which overlap, where the considerations for one affect the others. Finally, it is far more time and cost efficient to have one person co-ordinating and handling this area.

QC, Documentation and Training

Quality control, user documentation and

training are best handled by non-technical, and preferably non-MIS personnel. I have evolved a system whereby I work with a documentation and training specialist who used to be a teacher. I involve her in the project after the first major user demo, and she is involved thereafter until the system is implemented. She merges all three activities.

She learns the application from the user perspective by working with the users to evolve a training plan which will be oriented to their actual daily routine and problems. She learns the system through demos and questions, but mostly through exploring it on her own, trying things, finding out how friendly the system is or isn't, where it is and isn't foolproof, and where things work intuitively and where they don't. This entire learning process on her part provides the developers with a level of testing, debugging, quality control, and user friendliness "tuning" that they would otherwise probably never be able to duplicate.

The biggest payoff of the entire process, is that the perspective she brings is that of a user not a programmer, or developer, or MIS person. The way she moves through the system is the way a user would move through the system, the things she does are things users would do. What this enables us to do is correct problem areas before users ever see them. This gives the entire system far more credibility and acceptance with the user community.

When the system is "ready," it is debugged, the final demo has been done, our trainer holds a training preview class. This class is a test drive of the training process with a very select classroom of handpicked users. This is a real production preview for the development team, as this group finds all remaining issues that our documentation and training person missed. We leave a window of time between this "preview" class and the first real training class to fix all these remaining problems. When training commences the system is clean, user friendly, and functional. Users do not get discouraged and frustrated by encountering problems during training and early use.

Conclusion

The ability to make very rapid changes and corrections, even to basic design, in 4GL development environments, both eases and complicates project management using 4GL technology. One must start with a solid estimate that closely reflects the scope of the project.

An overall plan of deliverables with demo dates attached for the entire length of the system must be established up front and adhered to, even if there are occasionally items that are not ready.

The development team must be trained to work as a team, and this must be reinforced by regular weekly team meetings where the entire team has the opportunity to observe and participate in the evolution of the system as a whole instead of isolated pieces.

There must be a plan in place which allows for intensive testing and shakedown by a user oriented person, so that this is not left to the devices of even the friendliest MIS person. At the same time, the developers must be available and responsive to the change requests this testing generates.

When this methodology is embraced, truly phenomenal results are possible with virtually any team.

Paper # 3027

Transitioning Programmers from 3GL to 4GL

Suzanne Harmon
Information Systems Consultant
950 Graham Hill Rd
Santa Cruz, Ca 95060 USA
(408) 459-0802

Though fourth generation languages and development tools have been around for over ten years, they are still only beginning to gain acceptance and credibility. Many shops which purchased fourth generation tools in recent years only use them for ad hoc reporting and turn to 3GL solutions for system development of any consequence. Some shops have tried to incorporate 4GLs into their system development strategy and failed. Others have had an application developed by the 4GL vendor, or some other consultant, and then proceeded no further in their use of the tools. Why is it so difficult for information systems environmnents to transition from 3GL to 4GL development environments?

The 3GL Development Cycle

In a traditional 3GL development environment, the development cycle typically begins with a feasibility study followed by or combined with a conceptual design and functional specification. The resulting product can be reviewed with the intended users of the system to verify and validate the development team's understanding of the system requirements.

Once this step is complete and the design has been approved and accepted by the intended user community, detail design can commence. This step involves breaking the high level design down into programs for which detail specifications are written, leaving little or nothing to the imagination of the programmer.

When programming specifications are ready, programming can commence. The programmer writes, tests and debugs his or her program or programs until satisfied that they meet the detail specification.

When all programs in a sub-system or system have been completed to the satisfaction of the programmer, they are combined for system testing, or a period of testing to insure that everything fits together and holds together in a simulated production environment.

This entire process is extremely structured. Each step takes a relatively long time, with the average development project taking about two years. Therefore, a great deal of care must be taken to ensure that each step is as complete, as thoroughly airtight, as possible. At least as much time, if not more, must be put into up front design as is put into programming. In fact, it is probably realistic that twice as much time is spent in design as is spent in programming. The programming function is largely an exercise in doing what one is told, exactly and to the letter.

The 4GL Development Cycle

The 4GL development cycle starts out much like the 3GL development cycle, with a feasibility study and high level design. However, due to the structure inherent in most 4GL systems, the high level design takes on a different look, with much attention given to information flow and entity definition and analysis. Whereas the high level design in the 3GL environment tends to mock the ultimate "program units" at a high level, the 4GL design identifies entities, their relationships to other entities, and their relative position and importance in the flow of information.

This design may or may not be subject to a user review process before development of a high level prototype begins. The prototype serves as the ultimate and on-going communication vehicle with the user community throughout the development cycle. Because there is no detail design phase, the 4GL developer must be able to think creatively and convert user requirements into proposed solutions for user review and feedback.

The preliminary prototype is typically a surface level presentation of entities and flow. There need not be substance or functionality underneath the surface, and in fact this can result in time being wasted, as developing substance and functionality should only proceed once the developer can verify that the foundation of the entity analysis, flow and scope is appropriate.

Development proceeds with iterations of prototype refinement and user review, bringing the product closer and closer to a full function, production ready system. Through this process, the user demos incorporate an ever increasing number of system components working together, effectively replicating the system testing function of the 3GL development cycle.

The Hurdles to Transition

1. Letting Go. It is very difficult for 3GL shops to let go of the development cycle methodologies which they are used to using to embrace a totally new and unfamiliar routine. They want to develop with a 4GL language, but keep the 3GL development cycle methodology with which they feel comfortable and secure.

Result: The development benefits of a 4GL, which are to let user input drive the resulting product are diminished or lost by locking into rigid design and excessive functionality where there is much to lose by making user requested changes. Also, the staff is encouraged to maintain the roles they have had as 3GL developers instead of transitioning into the inevitable roles they must assume as 4GL developers.

Other possible side effects: Design problems inherent in trying to provide program specifications for 4GLs which are functionality, not program, driven.

Suggestions: Hire or contract with developers and/or project managers who have a successful history of developing in 4GL environments. Listen to their suggestions and follow their direction. Keep a positive attitude. If you think something is doomed to fail it probably will.

2. Changing Roles. In a 3GL environment there are typically the designers, who do most or all of the thinking but little or none of the implementation, and the programmers who do little or none of the thinking and most or all of the implementation. In a 4GL environment there is little room for either of these roles. Virtually everyone on a project must work as a team player, creatively thinking through solutions to user requirements, and then implementing them in the 4GL of choice. Making this change is extremely difficult.

Result: Designers are afraid they will be unable to produce as developers, and many feel it is a demotion in status. Programmers may become paralyzed by their fear of having to creatively come up with their own solutions to requirements, not to mention the fact that very few are equiped with the skills necessary to think an approach through from a user or functionality perspective. Much time is wasted as people get enveloped by their fears and anxieties.

Other possible side effects: Everyone would feel much more comfortable in their old roles and management would be much more comfortable managing the old roles. Therefore, the project gradually reverts to 3GL development methodologies.

Suggestions: Be prepared to be flexible, and experiment with moving people around in various roles and areas. Remember that the crucial issue is the productivity and output of the team as a whole, not of any one individual. Your ability to work with people to find ways for them to be successful will help relieve their anxieties and fears.

3. Beginning the development cycle without knowing your product (4GL). There is a new

$300,000 development project that is behind schedule and critical to someone's career. Management buys or decides to resurrect a 4GL to "save the day." They expect the MIS staff to wholeheartedly embrace the product and immediately realize the kind of productivity gains promised by the vendor.

Result: This can result in resentment and/or hatred of the product by most, if not all, the staff. At the very least it usually results in horribly inefficient and difficult to maintain code as well as productivity gains which fall considerably short of expectation. These projects frequently fail, or at the very least finish leaving bad feelings all around.

Other possible side effects: Trashing the 4GL and returning to a 3GL. Blame is placed on the 4GL vendor for misrepresenting the product.

Suggestions: Don't use a new 4GL for the first time on a major new development project. Use it initially for enhancements to existing systems or extremely small new systems, until a large percentage of the people who will be involved with development have a chance to become familiar with the product. Use for two years by 10% of the proposed development team doesn't count.

Get proper training. If there are more than a couple of people who will need to know and use the product, get custom training in-house. In the end it will cost you less because your time will be used more effectively. Don't pull people out of class because of other things that come up. One week of training is rarely enough. It is a good idea to follow up your first week of training with some product use or development time. Then schedule another week of more advanced training to address all the questions and issues which come up during the period of using the product.

If you have no choice but to use the product for the first time on a major new development project, bring in or hire people who really know the 4GL and how to work with it. Let them direct the project and mentor the people who are learning the product, and listen to them.

4. Specing the project as you would for 3GL development, then trying to develop it with a 4GL. This is a real killer, and what I would otherwise refer to as trying to fit a round peg into a square hole. This is an especially common side effect of number 1 or number 2 above.

Result: Most frequently, the programmer will claim that the 4GL lacks functionality and sophistication enough to do the task and will resort to their 3GL of choice to accomplish the task at hand.

Other possible side effects: Those programs which were written in the 4GL will be incredibly inefficient and may "look" out of place.

Suggestions: Design "into" your 4GL. Each 4GL has its own unique "style." It is critical in developing an attractive, effective, and efficient system that you design for that style. This includes data base design, screen design, report design, user interface, and even the test plan or QC process. In fact this style is so critical to the way your finished systems will look and act that you should learn as much as you can about it before you even choose a 4GL. This is facilitated by translating the functional spec directly to a prototype and working the prototype into a finished system. Many 4GL vendors have classes or consulting available to help new product users understand their product's style and techniques helpful for designing into it.

5. Assuming that you will have the kind of logical control you had with a 3GL. I have found that typically 3GL programmers feel most comfortable with COBOL generators which claim to be 4GLs or 4GLs which are entirely procedural, and are thus as close to being a 3GL as possible. These products will help somewhat with the cost of initial development, but long term productivity gains due to ease

of maintainability and shortened development cycles as familiarity with the product increases, will be lost.

Result: What most frequently happens when a 3GL programmer is reluctant to relinquish control to the 4GL is that they take every opportunity to use whatever procedure code is available with the 4GL they have. This usually results in lengthy code to do what could have been done automatically with default or design statements, thus defeating the entire purpose of the 4GL.

Other possible side effects: The programmer feels the 4GL is cumbersome, lacks functionality, and it would have been easier to use a 3GL in the first place. His boss is beginning to agree.

6. Performance is Unsatisfactory. If a prospective 4GL user asks me three questions, one of the three will always be: "What about performance?" The answer: "4GL programs are not as fast as 3GL programs." This can be compounded tremendously by the other hurdles discussed above. Inadequate knowledge about the product, inefficient use of the product, and system designs which "conflict" with the product can all contribute to additional degradation in performance.

Result: The product may only be used where performance is irrelevant, or may not be used at all.

Suggestions: First, review the hurdles discussed above. To optimize performance you must know your product, get proper training, design for the product style, and make enough of a commitment to become experienced with it, be willing to use trial and error, and get support from others who have experience with the product.

I usually take a period of several weeks at the end of a project to monitor the performance of the system under actual live conditions and tune it for improved performance where necessary. This is because trying to anticipate performance in a multi-user production environment can be difficult, and users rarely have time or resources to road test a system before it goes into production. Many tools are available to enhance the performance of a 4GL system when necessary. Supertool, MPEX, Omnidex, and a host of other products, offer tremendous performance enhancements to the average system.

There is, however, another issue where performance is concerned. Unless there is a serious problem with the system resources, performance should be in the eyes of the user, not the eyes of the MIS person. Sometimes MIS people feel they must put "performance standards" on systems in order to maintain control. In fact, if the system has functionality that greatly enhances the users productivity, which is more likely with 4GL development, and the user is ecstatic, then it becomes questionable whether externally established performance standards are relevant.

Conclusion

The 3GL programmer who is transitioning to 4GL developer will need to work on altering their mindset and their entire approach to system development.

The following is a list of ten commandments to help you through the ordeal:

1. I will read a section in my manual every day.

2. I will try at least one new function or feature of my 4GL every day.

3. I will not say it cannot be done in my 4GL until my 4GL vendor tells me so.

4. I will never resort to a procedural solution until I have made absolutely sure I can't do it with the language's design capabilities.

5. I will go to every meeting of my local

user's group.

6. If there is no local user's group, I will talk to my vendor about helping me form one.

7. Instead of abandoning my 4GL, I will try to rethink problems to take advantage of my 4GL's functionality and features.

8. I will not swear and have a tantrum when I can't find anything in the manuals which relates to my question or problems, but will try to seek out people who know the product to help me find solutions.

9. I will try at least ten different ways to do something new which I don't know how to do and can't find reference to anywhere (reduce this by two every six months).

10. If I can't follow these commandments, I will find another job.

Paper Number 3028

Managing the Software Process and Outsourcing

Michael P. Mazza
Brown & Williamson Tobacco
P.O. Box 1757, Macon, GA 31202-1757
912-749-8682

1. Abstract

The task of managing the software process is a challenging endeavor. There are plenty of opportunities and alternatives to improve the process. The option to outsource some of the functions gives the software development organization a new set of alternatives. This paper discusses many of the outsourcing opportunities and how they relate to the software process. The outsource evaluation process is discussed briefly along with some of the advantages and disadvantages.

This paper is limited to outsourcing activities related to the software process. There are many other activities including; Hardware Ownership and Maintenance, Telecommunications and Networking, Technical Services, Data Center Management, Etc.

The opportunities are so broad and diverse many organizations fail to recognize areas that they have already outsourced. Many people do not consider contract programmers or purchased software as outsourced activities. They are however some of the earliest ways in which software organizations have expanded their service level. They are often overlooked when companies evaluate themselves, because these options have been around so long. Both may offer reduced project costs compared to in-house development and they often reduce schedules dramatically.

There are many firms that offer outsourcing services. The evaluation process is fairly traditional. Several members of the organization are interviewed and policies, procedures, methods, standards, etc. are reviewed. The degree of conformity to the established methods is determined along with the organization's maturity level. Company and departmental objectives are defined so that the outsourcing priorities can be set. Often financial requirements due to mergers or consolidations drive the process. If there are agreeable opportunities the company may develop a request for quotation (RFQ) and start evaluating vendors.

Once the decision to outsource is made it may become impossible to reverse. The trend today in outsourcing is pushing way beyond purchased software and as more activities are farmed out the potential for lost control becomes greater. The effective software process requires established methods, practices, checkpoints, and controls. As more of the activities are outsourced the burden of communicating these methods becomes critical. Organizations with an immature software process may benefit from their relationship with a well established outsourcing firm. More mature organizations with established methods may get more utility from the technical advantages of outsourcing. Outsourcing may appear to be an easy way to reduce costs, speed project completions, and obtain additional technical expertise. These are all available and the press is full of success stories. There are however opportunities for failure if outsourcing is implemented inappropriately.

2. Opportunities

2.1. Application Development

Outsourcing means turning all or part of a company's software process over to an outside organization that performs such services for a fee. There are many companies that market these services and their profitability is on the rise. What was once a small segment of business for the hardware companies has grown to become an important share of their revenues. Analysts estimate that outsourcing generates about $6 billion now and is expected to reach $13 billion by 1994.

Contract programmers used to be a limited window of opportunity for the software organization. They were used to fill in during project crunches or personnel shortages. Today there are a wide variety of services available. There are data communications engineers, programmers, personal computer specialists, technical engineers, designers, drafters, technical writers, computer operators, and the list goes on and on. Depending on the nature of the project or organization these individuals can be hired for a few days, or permanently. Some companies use contractors to help them make a technology shift, or when they have a special project that requires experience that they do not have on staff.

With the readily available data communication technologies, the distance between customer and contractor is no longer an issue. Many outsourcing firms offer contracted resources via satellite or other high speed links. One benefit to having the work done off-site is the reduced overhead for work space, and possible cheaper labor rates. Another advantage for projects that are on a fast track is that multiple sites around the world can work around the clock. This requires project management skills above and beyond those normally found in Management Information Systems (MIS) shops. The outsourcing firm must provide this expertise.

The environment has changed dramatically within in the last two years. The speed of growth in the technical arena has been tremendous. Companies are finding themselves out of date technically. One option is to invest in training or education and the other is to turn to outsourcing firms. There are firms that specialize in; process automation, banking, insurance, material handling, purchasing, etc. automation technologies. They are available for contract to perform any and all phases of the system life cycle.

Some firms specialize in areas without regard to the hardware or software. They will come in to an organization and perform a feasibility study and make recommendations which are independently based. Other firms incorporate their project services directly with a hardware and software solution. Depending on their internal resources software organizations may choose which fits them the best. Sometimes firms will create strategic partnerships in order to acquire the technical leadership.

Project management is often a side benefit when outsourcing a project. The outsourcing firm may have a distinct advantage in how they manage the process. Other project management related options are to purchase project management training or consulting. The consulting can be limited to specific methods, or it can be as broad as the system assessment identified in Watts Humphrey's book. There are companies that specialize in implementing methods or development life cycle processes. They have consultants that work on projects just to ensure the process is followed correctly. They serve much the same role as Software Quality Assurance (SQA) groups while at the same time they provide training. Some of these firms have their own tools, forms, etc. that they sell with their service.

The demand for technology has surpassed most MIS department's abilities to supply. Some organizations are turning over entire projects to outsourcing firms. This is usually the case when the MIS group has decided to focus on the critical systems or the future systems and they do not have the resources to maintain the current requests. Another opportunity for MIS groups is in the Inspection and Validation area. Some outsourcing firms specialize in this area. The process can be formalized and MIS management can use the firm like a separate group within their organization, but without the overhead of the permanent staff.

Purchased software is another way in which MIS groups can extend themselves and the services they provide. The ideal situation has the MIS group work with the client to determine the requirements and then purchase a software package that satisfies all of the requirements. The logic is sound, however for it to work successfully the process used to evaluate and purchase the software must be solid. In theory the MIS group leverages their expertise and provides solutions for their customers earlier and often cheaper than if they were developed in house. Some of the pitfalls are; a poorly defined process for purchasing software, incomplete or incorrect requirements, unrestricted modifications or custom code to the purchased software, etc. These are often the same problems that plague MIS groups when they develop software.

The purchased software route has other side benefits. Often large software packages have user groups which provide avenues for change outside the MIS group. If the software is used widely many of the errors or problems have already been found and fixed. Most software companies specialize in specific areas so the organization gains the benefit from their expertise.

I consider software engineering productivity tools as an outsourcing opportunity. The tools that the engineers use to develop software can make a major impact on the development schedule and the quality of the product. Most hardware vendors provide the basic set of tools required to develop systems. Most systems have compilers, basic editors and other miscellaneous utilities as standard options. Early developers wrote many of their own utilities to improve productivity. The UNIX operating system and most of it's associated utilities are the result of this situation.

Today there is a vast variety of productivity tools such as fourth and fifth generation languages, relational data bases, analysis tools, simulators, code generators, etc. The MIS organization can purchase these products and extend their ability to deliver products. Many of the newer ones are self documenting and provide structured on-line help facilities or computer based training. Is departments that expect this level of performance without providing their developers with the technology are missing a tremendous opportunity. Their ability to perform and meet schedule dates are in direct proportion to the tools that they have to do their jobs.

2.2. Support and Maintenance

Support and maintenance can become the major portion of an MIS group's activity. As more software is developed it increases the amount that requires support. Contract programmers can provide viable alternatives versus having the MIS group perform strictly maintenance activities. One reason for staff turnover is the limited opportunities to do new development. Although there are programmers that prefer maintenance and support it is usually easier to contract out this function to a firm that specializes in it and has the resources to handle it regardless of the turnover.

Sometimes a mature system will require modifications and there is no one on staff that is familiar with the system or the language. This occurs a lot in the data communication area. Many times this requires contracting with the hardware vendor for one of their specialists. These individuals usually are in high demand and receive a premium rate.

Purchased software can provide the MIS department with substantially reduced support requirements. Maintenance contracts are usually standard and the software evolves along with the demand and all the MIS group has to do is administer the contract. If the product is widely used there usually are annual or semi-annual upgrades and phone-in consulting or a help desk is available.

Engineer productivity in the maintenance area can be enhanced with the proper tools. Again the logic here is do not design and develop the tools to improve your job if they are available. MIS groups need to look at purchased software for their use as an alternative. Data bases for tracking problems or software bugs are common, and data bases that track modules to determine their use in future projects are becoming common. The other miscellaneous utilities that make the maintenance programmer's job easier are often dollars well spent when you consider the amount of time they save.

I personally think the jury is still out on End-User computing tools. These were considered silver bullets because they would free the MIS professionals to focus on the critical aspects of the systems. Some are more successful than others and with the proper training and discipline some users are extending the use of the information. Personal computer software provides a level of service that the MIS group may never match. Imagine an MIS manager proposing that his group begin a major software development project to create a better word processor. Instead most MIS groups outsource this requirement. Their role here is to develop a standard set of acceptable packages to ensure compatibility.

Outsourcing change control functions and Software Configuration Management (SCM) has several advantages. The software organization may not have the personnel resources or experience to address this area without extensive training. There are many tools or software packages available that can be used by the MIS group to address SCM issues. Software librarian products help with the record keeping on software changes. Security modules with various authorization levels are standard components for most of the packages. There are many tools and the services are available. One advantage of outsourcing the entire SCM group is that they can have a different reporting relationship than the development group. The MIS organization could charter the outsourcing firm to improve the change control process and structure their payments accordingly. Regardless of their maturity level the MIS group can improve their process by implementing a SCM function.

3. The Outsourcing Process

3.1. Assessment / Evaluation

Whether a company decides to outsource or not, is the second decision they should make. The first decision should be whether to evaluate it. Most companies should evaluate themselves because it forces them take a critical look at what they are doing and how well they are performing. Customer satisfaction is one of the best measures for an MIS organization. How many projects are delivered late and how many post implementation bugs are discovered by the customer are two of the key questions each MIS group must ask itself . The current backlog of requests and support staffing are two key indicators of unsatisfied demand.
The MIS department should be reviewed in detail and broken down into major activity areas. Percentages should be determined for the various activities. This will provide management with a rough idea of where the resources are being used. Once this is done management should compare it with the company goals and objectives. The question to answer is, "are we working on the right things?".

A cross section of the MIS department should be interviewed to obtain an understanding of the informal structure. One thing to look for is a large application development group, that spends most of their time handling support calls. If this group is also constantly late delivering projects there is probably a misconception about their role. External evaluators can usually find out what is really going on and they have a better chance of informing and influencing management.

Other areas to review are the methods and procedures. Do they exist and are they being followed? Are they routinely updated, and who is responsible for their upkeep? These areas are usually high priorities for MIS management, however they are often neglected because of more pressing projects. It is difficult to get funding for these types of improvements because there is no obvious return. An outsourcing firm can help show where capital and engineer resources can be applied to improve the process and thus reduce the overall costs.

3.2. Organizational Maturity Level

The evaluation will provide a different look into the MIS organization. One of the important areas for review is the process maturity level. Based on the review the MIS group needs to decide if they are accomplishing their objectives in the most efficient manner. Are there strategic systems that require new technologies that the group cannot embrace, for whatever reason? Are the best engineers shackled with support and maintenance tasks on old obsolete systems? Some MIS groups will come to the conclusion that they should not focus on new technology and this may be outsourced. It will minimize costs and speed the introduction. Regardless the focus should be using technology where it provides a competitive advantage. For some organizations there is no advantage in having their own MIS group.

3.3. Establishing Priorities and Plans

After the evaluation is finished the organization will have an area or areas where expertise and resources are needed. Most experts agree that a committee should be formed to evaluate the outsourcing option. The MIS group should head up the effort. It is important to have a representative from every user organization that will be affected by the decision. The committee should plan to take as much time as required to make an intelligent decision. It usually takes between three to six months. They should evaluate the current situation and then request information from several outsourcing firms. Once the top firms have been identified a more through examination should be conducted.

3.4. Contracts / Partnerships

In setting up an agreement the organization should control the relationship from the very start. There are many established outsourcing firms. Start by requesting a written evaluation of the area you plan to outsource. Outline a series of timely updates or monthly milestones that you would like to see accomplished. These will serve as your requirements and they will provide you with a way to measure the project. This should provide a good starting point for contract negotiations. The reports will allow you to see things that may have been overlooked when the function was handled in-house.

The following is list of issues that should be covered during negotiations.

Pricing: Make sure you understand exactly what the price covers. Determine if there are any services or requirements that will incur additional charges. Establish the time frame for price structure reviews, and the procedures required to amend them.

Services and Performance: The contract should indicate the platform where the applications will be processed. Check to see if the platform is shared or dedicated and determine what services will be provided. Some of the more common services are performance standards, measurements, security, backups, and disaster recovery plans. The organization should take a critical look at response times and customer satisfaction. If the outsourcing firm does not offer any real advantages in these areas it may be unwise for the MIS group to consider the vendor.

Ownership and Control: The level of control over the technical decisions concerning hardware, software, and architecture is one that many organizations are not prepared to negotiate. The contract needs to be clear as to the ownership of the software developed during the contract, and about any hardware or software acquired during the contract. The plan needs to have language in place that outlines any confidentiality or conflict of interest issues with the organization's competitors. Also, contingency plans need to be defined in the case of the outsourcing vendor being sold, or filing bankruptcy.

4. The Outsourcing Decision

4.1. Advantages

Organizations turn to outsourcing because they have a lack of in-house resources, staff, or expertise . Cost savings are sometimes an important factor, but this is usually secondary for those companies focusing on the software process. The majority of companies that report enormous savings do it with combined staff reductions and elimination of the hardware expenses. Each organization's advantages will be dependent upon the objectives that were predetermined in the evaluation and selection stages.

When an organization is forced to make a quick technology shift it may be advisable to outsource the change agent. This may reduce the development time and the training expenses. The organization would in effect acquire skilled personnel without the trouble of training or placing them on the payroll. The organization can then stay focused on what it does best.

If the MIS functions are merely service related and of no strategic importance to the organization, it makes sense to focus the internal resources elsewhere. An evaluation into the outsourcing potential will determine which functions are critical and which ones are merely overhead. Increasingly MIS groups are asked to spend more time on the business issues and outsourcing the routine systems or the management of the software process makes good business sense. Again during the evaluation stage the organization must develop a comprehensive set of measurements in order to decide what, if anything, to turn over to an outsourcer.

As the organization begins to work on the business issues it allows the outsourcer work on the software process issues. With a mature outsourcing vendor the organization should feel comfortable that they have a strategic partner who is concentrating on the technology and process. The software process is the core business for most outsourcing firms. They can spend their research and development dollars exploring new areas of improvement or advancements in the MIS field.

Sometimes it is too risky not to outsource a major new system or technology shift. Because MIS is not the core business for many organizations when they are faced with projects or problems that they have never faced before, outsourcing becomes an attractive alternative. There is usually no way that the organization can ensure that the time or cost estimates from an internal study are accurate because of their limited experience in the area or the lack of historical data. Outsourcing may be the least expensive way to introduce the new technology. It may even be less disruptive to the day-to-day operation than a major in-house conversion effort. Sometimes the choice is not between outsourcing or doing it internally. It is between not doing it at all or outsourcing the project.

There are many advantages such as; the speed of change, increased technology, reduced training efforts, concentrated business focus, reduced internal costs, and an improved software process. Because the core business of an outsourcer is the MIS function their level of software quality can be much higher than the average MIS group in a organization. As the outsourcer expands their business the overhead costs of software improvement groups and technologies is spread across the customer base. They can therefore do more for each customer for less.

4.2. Disadvantages

The decision to outsource may be irrevocable. If the decision involves elimination of staff, and if you have to rebuild staff latter it may be a formable task. Getting it back is always harder than giving it away, so if you fail with one outsourcing firm you may have to select another to take it's place.

The level of service may erode with time. Initially when the marketing types are anxious to sign up new clients the service is outstanding. Unless the contract stipulates a service level it may drop off dramatically. There are a few horror stories about outsourcing firms changing computer platforms which resulted in degraded performance. The profitability of the outsourcing firm is a major concern for the organization. If they default or retrench it will impact the level of service and your users will suffer.

From a software process standpoint the outsourcing firm selected should have a history of accomplishments and satisfied customers in the area that you are anticipating using them. This may be the easy part. The harder part is turning over control and allowing them to do their job. An attempt to micro manage their efforts may result in confusion and dissatisfaction. Strategic systems should be kept in-house. If a system is critical to the organization it may become to static if turned over to an outsourcing firm. As the organization changes the strategic systems must evolve as well. It is difficult to bring this about if every change requires contract negotiations. Slow response may result in missed profits and ultimately loss of competitive advantages.

The following is a good example of a company losing a competitive advantage. Let's imagine a company outsourcing the integration of it's manufacturing equipment. When outsourced the equipment was the latest technology and produced sub-assemblies at a competitive rate. The outsourcing firm spent a considerable amount of resources to accomplish the task. A few years later the next generation equipment is available and it requires a different integration strategy. Unless the organization can respond quickly through the outsourcer it may be saddled with old equipment or an incomplete integration.

Here the outsourcing firm must be responsive and forward thinking. In addition the MIS organization must be understanding and allow for them to recoup some of their investment. If this cannot be accomplished there will be missed opportunities. This points out the fact that as the organization grows the outsourcing firm must grow also. Otherwise the organization will find itself lagging behind its competitors.

One of the biggest risks involved with considering outsourcing are the political ramifications. It must be understood by everyone involved that the objective is to make the company more competitive and effective. This is hard to explain to an individual or a group of individuals whose lives are in the process of changing. Care must be taken to address each concern openly and even then management must be aware of the potential risks.

OPPORTUNITIES

A. APPLICATION DEVELOPMENT

1. CONTRACT PROGRAMMERS

ON-SITE SEMI PERMANENT EMPLOYEES

OFF-SITE CODE SHOPS

TECHNICAL EXPERTISE

PROJECT SPECIFIC RESOURCES

2. PROJECT SERVICES

FEASIBILITY STUDY

FUNCTIONAL SPECIFICATIONS

SYSTEM REQUIREMENT DOCUMENT

PROJECT MANAGEMENT

DEVELOPMENT PROCESS CONSULTING

TURN KEY APPLICATION DEVELOPMENT

QUALITY ASSURANCE

INSPECTION AND VALIDATION

3. PURCHASED SOFTWARE

GENERALIZED APPLICATIONS

REDUCED SCHEDULE

REDUCED SUPPORT

OPPORTUNITIES

4. SYSTEM ENGINEERING PRODUCTIVITY

   4GL & 5GL ENVIRONMENTS / CASE

   DATA BASE SUITE OF PRODUCTS

   DOCUMENTATION TOOLS

   COMPUTER BASED TRAINING

B. SUPPORT AND MAINTENANCE

1. CONTRACT PROGRAMMERS

   ON-SITE SEMI PERMANENT EMPLOYEES

   OFF-SITE CODE SHOPS

   TECHNICAL EXPERTISE

2. PURCHASED SOFTWARE

   CONTRACTED MAINTENANCE

   PERIODIC UPGRADES / RELEASES

   PHONE-IN CONSULTING

3. SYSTEM ENGINEER PRODUCTIVITY

   DIAGNOSTIC & DEBUGGING UTILITIES

   SOFTWARE RE-USABILITY TOOLS

   PROBLEM TRACKING

4. END USER COMPUTING TOOLS

   INQUIRY & REPORT WRITER PACKAGES

   PERSONAL COMPUTER SOFTWARE

Opportunities

5. Software Configuration Management
   - System Administration
   - Software Librarian Utilities
   - Change Control Software

6. Data Center Management
   - Job Scheduling
   - System Performance Monitoring
   - Backup Utilities

The Outsourcing Process

A. Assessment / Evaluation

1. Customer Satisfaction

   Software Delivery History

   No. of Post Implementation Errors

2. Information Systems Dept. Review

   Interview Key Personnel

   Examine Methods and Procedures

B. Organizational Maturity Level

1. Software Engineering Objectives

   New Development / Technology

   Strategic Systems

   Support & Maintenance

2. Business Objectives

   Minimize Costs

   Competitive Advantage

C. Establishing Priorities and Plans

1. Review Comprehensive Measurements
2. Recommendations from Findings & Objectives

D. Contracts / Partnerships

1. Measurements for Satisfaction
2. Control of Technical Decisions
3. Contingency Terms

The Outsourcing Decision

A. Advantages

1. Technical Expertise

   Quick Technology Shifts

   Reduced Training Expenses

2. Strategic Business Focus

3. Rapid Technology Shift

   Technology Managed by the Outsourcing Firm

   Technology is their Core Business

4. Higher Level of Quality Assurance

B. Disadvantages

1. Limited Control of Service

   Level & Degree of Freedom

2. Critical Dependency

   Outsourcing may be Irrevocable

   Profitability of Outsourcing Firm

3. Loss of Strategic Direction
4. Reduced Competitive Advantage

   Responsiveness of Outsourcing Firm

   Missed Opportunities

   Growth Expands Beyond Service
5. Internal Political Risks

6. Bibliography

Williamson; Mickey "Outsourcing: The Decision." CIO 15 Oct. 1991: 23-37

Williamson, Mickey "Outsourcing: The Vendors." CIO 15 Oct. 1991: 38-43

Stevens, Larry "Getting By with a Little Help from Your Friends. Manufacturing Systems 3 Mar. 1991 52-55

Ward, Bernie " Hiring Out" Sky Aug. 1991 37-45

Cylix Communications Corp. Questions and Answers about Outsourcing. 1991

Humphrey, Watts S. Managing the Software Process. Reading, Massachusetts. Addison-Wesley

Paper No. 3029

You, Yes You, Can Write Documentation!
A DP Professional's Guide to Writing

Pamela Dickerson
ECI Computer, Inc.
1231 E. Dyer Road
Santa Ana, CA 92705
(714) 434-8841

"I hate writing documentation."

This simple statement sums up the feeling of many data processing professionals. In fact, not only do most DP professionals hate writing documentation, they hate writing pretty much anything. Memos, letters, specifications, documentation, training guides -- any of these tasks can cause the most hardened DP manager to break out in a cold sweat.

There is hope, however.

By following a few simple guidelines, you can reduce the amount of stress associated with preparing the documentation that fills every DP person's life. With a little practice, you can write good documentation. Do it often enough, and you'll find that documentation becomes, well, if not fun, at least easier to put together.

In this paper, you'll learn to:

- Determine what you're trying to communicate
- Determine who you're trying to communicate with
- Outline what you want to say
- Prepare a first draft
- Edit and rewrite your work

- Evaluate the usefulness of spell checkers and grammar checkers

- Use the word processor or DTP system that suits your needs

- Overcome writer's block

- Evaluate good documentation

Throughout the paper, we'll examine specific types of documents most DP professionals come across in their work:

- Specifications

- Technical documentation

- End-user documentation

- Training guides

- Procedural documentation

- Letters and memos

What Are You Trying to Say?

Good writing of any type begins with an understanding of what it is you are trying to communicate. Thank you notes are easy -- you're thanking someone for a specific thing. Letters to relatives who live far away are filled with news and tidbits about the kids.

When you write documentation, you are generally trying to communicate "how" to do something: how to perform a process, how to install hardware, how to run a program. Or, if you are writing specifications, you may be describing what something does. You may be called upon to write procedural documentation where you have to combine how to do several specific things. It may be your responsibility to put together training guides where you not only explain how something works, but you have to teach someone how to make it work.

Specifications are written to describe how something is going to work. This is where you take the preliminary information someone -- usually an end user -- has given you and you write up what you think they want. *Functional Specifications* are general documents that outline what the program or report is designed to accomplish, what interactions will be involved with other processes on the system, and may include how long detailed (or technical specifications) will take to write. *Detailed Specifications* outline in detail what the program is going to do, what the input screens (if any) will look like, what the report (if any) will look like, and what other processes are affected by the new or modified program. Ideally, you'll include a timeline for coding, documentation, testing and implementation.

Technical documentation is the internal documentation that provides a "trail of bread crumbs" for the DP personnel who follow you. This is programmer-to-programmer documentation explaining what you did and why. Carefully written technical documentation can reduce training time for future programmers and QA specialists, and can increase the productivity of your DP staff.

End-user documentation is the manuals that many of us speak of disparagingly. These are reference guides intended to tell users how to use the programs that we have painstakingly written. Well-written end-user documentation adds value to our software product; poor end-user documentation increases the load on our support personnel.

Training guides teach end users how to use your product. These are different from end-user documentation in that they take users through the product one step at a time, with review to make sure that the reader understands the concepts discussed to that point. Training guides may take the form of workbooks, or may be designed to be used in conjunction with classroom training. If you are writing training guides, you'll want to include many examples, and try to be consistent by using the same data throughout the text.

Procedural documentation takes the technical, computer-based product and combines it with the manual procedures required for a person to perform a job. For example, if you supervise an in-house DP department, you may write procedural documentation to cover the backup process. In addition to detailed information on how to spin the tapes and what information to input to the system, you may include information on filing backup listings and preparing tape labels.

Letters and memos for DP staff members are generally written to explain or update. You may explain what projects your staff is working on, or you may be updating your manager on where you are on a specific project. You may explain to customers what is included in an upcoming software release, or you may tell your QA staff when they can expect new software to test. In some cases, you may be disciplining an employee. These are usually short (one page or less) documents that have to convey information clearly and quickly. In some cases, these seemingly innocuous documents can be legally binding.

Who Are You Trying to Say It To?

Before you start writing, know your audience. If you're writing specifications that need to be read by technically competent staff members, you'll write differently than if you're writing a cover letter to the president of one of your clients. Your audience determines your language, the format the documentation takes and even the content of the document. This doesn't mean that other people aren't going to read the document; it means that you've identified the primary audience and you are going to write to them.

As you consider your audience, keep the following in mind:

- Technical expertise -- do audience members have a lot of technical expertise, or are they nontechnical individuals?

- Education -- what level of education are you writing to? Your writing will be very different depending on whether or not your audience has completed high school, and different again if they are college graduates.

- Familiarity with the product -- are audience members familiar with the product or process you are writing about? If so, you can take shortcuts in the writing and speak directly to them. Otherwise, you'll need to be very clear about your ideas.

- Reasons for reading -- why are your audience members reading your document? Are they reading it to make a decision? Are they reading it to learn? Are you trying to sell them something? Knowing why your readers are taking the time to examine your document gives you the ability to better address their needs.

Specifications are usually read by decision makers. *Detailed specifications* are likely to be read by individuals who are familiar with the product; *functional specifications* may be read by individuals who are not so technically adept. The education level is likely to be high in both cases. People who read specifications want to learn about your proposal, but also expect you to be selling them.

Technical documentation is likely to be read by technically competent individuals who have a high level of interest in what you have to say. You can -- and should

-- use technical jargon here since both you and your reader will understand it. These individuals are likely to be familiar with the product, and eager to read what you have written.

End-user documentation is generally read by people who are trying to accomplish something else. They've hit a sticking point and are turning to the manual to help them out. Depending on the product in question, they may have a high level of technical ability, or very little. Readers want information to be easy to find and they want to be able to understand it quickly. While some readers look over the documentation when they are not in the middle of a problem, you have to assume that a large number of them are going to be in trouble when they turn to this document.

Training guides are your opportunity to tell users the proper way to use your product. Readers want to understand your product, and are highly motivated to want to make sense of your document. Technical expertise and education levels will vary, but you can address the middle ground and satisfy the others.

Procedural documentation is also likely to be read by those who want a full understanding of a particular process or product. They may not be highly motivated to read through this document; their supervisor may be requiring that they go through the documentation. Careful attention to detail is crucial in this type of documentation as that determines the success of the product's implementation. Technical expertise is less important in this type of documentation than education level.

Unlike the other forms of documents discussed here, *letters and memos* are unsolicited documents. Whereas users seek out documentation, you are initiating the contact in a letter or a memo. In addition to the other considerations, you must also keep in mind your relationship to the readers, and the level of familiarity you enjoy with them. If you are writing a memo and sending it to department heads, vice-presidents and the president, you will write differently, and more formally, than if you are writing a memo to your staff.

Outline What You Want To Say

This doesn't mean that you have to use roman numerals and careful block style printing. Just jot down the main points you want to cover in your document. At this point, don't worry about the order they are in or whether you are leaving anything out. Go over this list until you are certain it is complete. Now, go through and put the points in order. How do you want to address the various topics? Are there more topics than should be adequately covered in a single document? Don't ask a single document to do too much. Set out with one goal for your document, then stick to it. If a topic doesn't apply to the goal, re-evaluate its appropriateness. If you decide that you must include it, maybe you need to redesign the goal of the document.

Specifications should include an overview of what is proposed, what the proposed program or modification would do, how it will interact with the current situation and what resources (hardware, software, people) will be required to make the project happen. Timelines and costs should also be included, or at least considered.

Technical documentation should set out to explain to other technical individuals what a program or process does. Include database structures, changes and modifications, and use this as a road map from one techie to another. Be specific and concise.

End-user documentation should provide users with the reference material they need to understand your program. For small projects, this may only be a few pages; for large systems, this can take up many notebooks. Don't try to make these training guides; assume your readers have been through training and have just run into a few problems along the way.

Training guides are where you introduce users to the program and tell them how to run it. Use real-world examples, if you can. Use the same data throughout the training guide for illustrations so that users can see how the data changes. Include a glossary, and don't assume your readers understand anything. Users who understand concepts you explain in detail will have their confidence boosted ("Gee, I already knew that") and the others will be grateful. Remember, readers of training guides want to learn.

Procedural documentation should have its scope carefully planned out. It's easy to try to include too much information in this type of documentation. Limit yourself to one job function, one departmental function and so on. Refer to other documentation as appropriate, and be specific when referring to nontechnical tasks, such as labeling tapes or filing reports.

Letters and memos should address a single issue in most cases. You might cover diverse issues in memos to your staff or letters to customers, but you generally want to focus on a single point. Don't make these documents too long: once you've made your point, move on. It is easy to gloss over the importance of well-written correspondence, but this may give the reader the first impression of your company or department. Well-written letters and memos can do much to improve or sharpen your image.

Prepare a First Draft

Using your outline, write your first draft. You may find it easiest to start in the middle, working from the framework your outline provides. You should not be concerned about format at this point, just put the words down on paper. The art of writing good documentation, regardless of the type, is in editing and rewriting. For now, worry about filling in the outline.

Once you've filled in the outline, you can write an introduction and a conclusion. In most cases, you'll use the introduction to tell the reader what to expect and why the document is important. In the conclusion, you summarize what the document is about, and identify the next step to be taken (if appropriate).

Use the introduction in *specifications* and *technical documentation* to tell the reader the scope of the document. You may want to use several paragraphs to do this. You can identify what components are included in the document, and identify other documents with which you expect the reader to be familiar. Conclusions in specifications will outline the steps to be taken next (designing detailed specifications, beginning the coding). Conclusions in technical documentation may describe other documents that the read might find helpful.

End-user documentation requires a detailed introduction that includes not only the scope of the document, but also the conventions you are going to use. You may indicate the level of expertise you expect readers to have, and even include a user profile so that readers can determine if this is the correct documentation for their needs.

Training guides often have more than one introduction. If a training guide is teaching an Accounts Receivable system, for example, and is divided into chapters on Invoices, Statements, Payments and Agings, you may have a different introduction for each of these chapters. Here, you'll tell the reader what they can expect to learn in each chapter, and may identify key words and phrases they should be able to define when they complete the chapter.

Procedural documentation should begin by identifying for whom the document is designed, and exactly what is -- and is not -- included in the text.

Letters and memos should identify in the first sentence (or at least the first paragraph) why you are writing. If you expect some sort of action to be taken, such as the person you are writing to needs to call you, or you want them to return something to you, make that clear in the last paragraph of the letter or memo, as well.

Edit and Rewrite

Once you have written the first draft, you can take a critical look at the document and start to edit and rewrite it. This is an iterative process and you may edit and rewrite a single document a dozen times.

Editing is not the same as proofreading. When you proofread, check for spelling and grammar. When you edit, check for the following:

- Are you saying what you want to say? After reading what you've written, do you have questions? Are you saying too much? Too little? Are you unclear about some areas?

- Edit to make the writing more powerful. Look for words and phrases that weaken the document, or which do not fit your reader's profile. Use the language tips below to help make your writing strong and avoid clichés and stock phrases.

- Don't be afraid to take out text, or to add entire sections, if the text calls for it. If you've included sections that don't help the document, take them out. If you have left out key components, add them back in.

- Rewrite the document based on your edits.

- Give the rewritten version to someone else to read. Ideally, this should be someone who closely fits the reader profile for the document. You will not always be able to do this, but you'll get good feedback from people when you can. Above all, put your ego aside when you do this. People want to help you make the document stronger. That doesn't mean that you take every piece of advice that's offered, but consider each piece of advice carefully before discarding or accepting it.

- Give your work a final reading before deciding it's complete. If you can, put it aside for a few days before reading it one last time. That will give you the time you need to look at it in a fresh manner.

Remember that very few people are able to write well the first time; for most of us, it is a repetitive process that improves each time we return to the document.

Language Tips

- Avoid empty phrases. Instead of beginning a letter "I would like to take this opportunity to thank you for the chance to meet with you last week," try "Thank you for meeting with me last week." This is a much more powerful opening that leaves no doubt in the reader's mind that you were indeed pleased with the chance to meet. *In fact* (as opposed to in make-believe?), *As you already know* (so why are you telling them?) and *Of course* (there's no need to state the obvious) are also stock phrases that can detract, instead of enhance, your writing.

- Use the second person, active voice. That's writer's jargon for speaking directly to the reader. Instead of writing, "The user enters the number at the first field." Write, "Enter the number at the first field." This style brings the reader into the document and keeps them involved.

- Write to the education and technical level you identified when you first started. If you use words that are over the heads of your audience, they will be bored and not reap the benefit of your document. If you use jargon that they do not understand, they will find your work pretentious and unhelpful. Instead, write directly to them. If you've included a "when to call for help sheet" in a manual that is going to be read by clerks, don't write that their call will be "escalated through the prioritization process to a level commensurate with the lack of functionality they are experiencing." Instead, write "calls are assigned priorities based on what tasks no longer work."

- Keep tasks in order. For example, do not write, "Purge all data after performing a backup." Instead, write "Perform a backup and then purge data." If you do not, you will have some users who do each tasks in the order you tell them, without reading the entire document. Help save your readers grief by keeping items in order.

- Don't allow your style to detract from the content. One popular drawing program began a book-based tutorial with the instruction: "1. Draw the left wing of a butterfly." For many readers, that was far too daunting a task to be overcome in order to learn the product. While the goal was to illustrate symmetry and mirror images, the result was that the reader was bogged down in creating the left wing of a monarch butterfly (as illustrated in the text). Keep your reader focused on the issue at hand. If the reader is distracted from that issue, you're not doing your job.

- Use the appropriate level of familiarity. If you're writing to someone you've never met before, use the formal Mr. or Ms. in the salutation. If you're writing to someone you've never met, but you've established a telephone acquaintanceship with them, you might sign only your first name above both printed names in the signature block. If you think the document might be shown to more people than just the primary audience, keep the writing somewhat formal.

- Avoid humor. What may strike you as the funniest way to phrase something may offend someone else. And the first time you read something amusing, you may smile, but it is no longer funny the 12th or 13th time. This is especially important with permanent documents, such as manuals and specifications. For letters and memos, you may be able to include some humor, if that is your personality, but if you are unsure who else may read the document, keep the cuteness factor low.

Spell Checkers

Are spell checkers useful? Absolutely. Do they replace the need to proofread your document? Absolutely not. Spell checkers have become sophisticated tools in recent years, but the English language is amazingly complicated, and spell checkers cannot take the place of proofing your work. All spell checkers can tell you that "owt" is not a word. Simple ones will not tell you, however, that you have typed "to" where you meant "too." There are spell checkers on the market that are smart enough to help you through words that sound alike or words that are often misspelled in place of each other (such as "through" and "though"). Spell checkers are beginning to merge with the next topic, grammar checkers.

Grammar Checkers

A good grammar checker is similar to having your high school English teacher always at your side. They can tell you what grade level you are writing at, depending on variables calculated through standard indices, and they can help you with run-on sentences, the passive voice and other common writing problems. Some grammar checkers enable you to define your own style so that you can break high school English rules (such as beginning a sentence with the word, "And"). Even if you write professionally, you are likely to find grammar checkers to be an invaluable aid to strengthening your writing. By paying attention to the suggestions and problems that the grammar checker points out, you can improve your own writing ability.

This paper has a 60.9 Flesch Reading Ease level and a corresponding education level of 8.9. These figures mean that someone who has finished the eighth grade should be able to understand the document. Flesch and Flesch-Kincaid are commonly used indices for readability: the higher the reading ease score, the lower the corresponding grade level. This document also has a Gunning Fog index of 11.4. This is another way of calculating the education level and, in this case, is several years higher than the Flesch grade level. Since the primary audience of this paper is expected to have finished high school, the readability is within an appropriate range.

Tools

Which word processor or DTP system should you use? The answer to this depends on a number of factors. Are you preparing the final draft? If so, you'll want a powerful word processor that gives you the ability to do page formatting. *Word for Windows*, *Word*, *WordPerfect*, *PageMaker* and *Ventura* are examples of word processors and desktop publishing programs (in the case of the last two) which are PC-based solutions to your problem. An HP LaserJet II or later provides you with the flexibility you need to produce high quality output, or you may want to use a service bureau to produce even higher quality originals.

If you're using the HP 3000 Series, you have access to QUAD, TDP and other mini-based word processors. Laser printers are still your best output device, but you'll have problems finding service bureaus able to read the formatted files. You may have to translate to ASCII files for these products.

Whatever tool you decide to use, read through its documentation to make sure that you use the product to its full potential and to achieve the results you need.

In some cases, you'll only be expected to prepare the rough draft and a production group will put together the final draft. If this is your situation, don't fret over formatting. Put your ideas on paper and leave the formatting to the production staff. If you have ideas, include those when you give the draft to Production, but leave them to perform the actual wizardry.

Writer's Block

Years ago, writers found a blank sheet of paper and a pen, or typewriter, intimidating. Today, the technology has changed, but a blank computer screen can still be intimidating. Procrastination can overtake even the most occasional writer, and writer's block is a serious problem.

You can try to overcome writer's block in a number of ways. If you've set down an outline, that should help you get started. Pick one of the topics on the outline -- any topic -- and start writing about it. You're going to go back later and rewrite it anyway; right now, you just want to get something down. If the outline is where you encounter writer's block, try breaking the project into smaller chunks and work on an outline for the smaller piece. You'll find that just breaking the project into smaller pieces helps create an outline.

Sometimes, the best thing you can do is to walk away from the document and do something else. When you return, you may find that you are able to approach the document with a fresh perspective and you will overcome the block. Or, while working on another activity, you'll be able to put together how you want to begin the document. Of course, this can easily turn into a delay tactic, and writers have been known to shampoo carpets while procrastinating. That is not the same as taking a break to come up with new ideas.

If you're in a place that is isolated, you can try speaking the document. In a conversational tone, explain to an imaginary audience what you are trying to write about. In some cases, just saying, "I need to write a memo to the vice-president about the status of the implementation," can lead to a better understanding of what that document will include. In this case, you can ask yourself the details: what *is* the status of the implementation? Why does the vice-president need to know? Is it good news? Are there factors involved that the vice-president doesn't know about? Such a self-interview technique can help you overcome having nothing to write about.

Be sure to put aside enough time to complete the document. If you're working under an unrealistic deadline, the presence of writer's block will only add to your stress, increasing the likelihood that the writer's block will persist. Give yourself adequate time to prepare, and you'll be able to complete the document even if you do have delays here and there.

Sometimes, writer's block occurs not at the beginning of a project, but in the middle or at the end. Working from an outline can help alleviate this possibility, but you can also use some writer's tricks to avoid this pitfall. For example, always leave a document for the day when you know where you are going to start tomorrow. If you stop with no idea of what the next paragraph is going to be, you're going to have a harder time starting up again. If you plan out where you're headed, that gives you the ability to get right to work and into the groove of the document.

What Is Good Documentation?

If you want to find examples of good documentation, or ideas to use in your own documentation, look around your office, your MIS room, your users' cubicles. Look at each manual carefully. Find the manuals you like, and try to figure out what makes them good. Take the manuals that you don't like and figure out where they went wrong. Keep these characteristics in mind as you write your own documentation.

- Collect samples of documentation that you like and that your users like. Keep notes as to why it is useful and what it does well. If it has specific sections that are especially well done, jot down ideas on how you could do something similar in your environment.

- Critique documentation that you don't like. Write down where the problems are. If the language is difficult to understand, write that down. If the author doesn't seem to know what he's talking about, note that. If there's too much text on a page and no room for you to make notes, write that down, too. Use this list to check against your own documentation.

- Do the same thing with letters and memos. When you come across a letter or a memo that you like, or that you think is particularly effective, keep it in a file so you can pull it out when you have to write your own. By imitating what you find effective, you are likely to increase your own level of effectiveness.

A Few Parting Words

The more you write, the easier it becomes. As with so many activities, you will improve with practice. Writing may not be your primary job, but you are guaranteed to have to write in your job. Improve your writing skills and you improve your value to your company, and your productivity. Learn to write effectively, and you increase your ability to communicate and to accomplish your primary goals. Move to the point where the act of writing is no longer a chore, and you increase the satisfaction you have in your job.

I've suggested a process that involves outlining, writing, editing and rewriting. In the real world, you may not have time for each of these components. The more often you are able to use them, however, the better able you will be able to short-circuit the process when time and resources demand. Strive to improve your writing and you will discover that you, yes you, can write documentation!

#3030
Robert Lund
34130 Parkwoods Dr. N.E.
Albany, Oregon
(503) 327-3800

DP - Life Success Principles

Abstract: *Many years ago Aristotle spoke of three essential ingredients for success. In this talk I will present these principles in the context of a computing department. The emphasis will be on how to not only improve DP staff, but to instill the same vision into the entire company so that the user community relationships benefit. These principles have been implemented within each department I have worked in with excellent results each time. I have recently lectured on these principles to business managers in eastern Europe with great results. The user will be able to immediately take practical steps not only within their department, but also in their personal life as well.*

There is a great need in our time, indeed a renaissance, of a return to the fundamentals of real success. Not only are we crying for such a return in our political arena, religious life, marriages, but also in our business and professional lives. I will call this movement the Great Return. Here's why. As a culture progresses in technology there is a great temptation to ignore or deny the basic needs we have as humans when relating to one another. Some of these needs are:

- **Love and Approval**: In order to have a healthy and balanced life, we desparately need to feel the care and compassion of others. Often, this non-optional ingredient of human health is eclipsed by "weightier" things such as tech nology, projects, deadlines, money, etc. Remember when the doctor's first question was: *"Where does it hurt?"* instead of *"Do you have insurance?*

- **Trust and Faithfulness:** In order to have successful relationships with others, we must know that we can count on their promises and commitments as being good. One of the founding pillars of the United States was the fact that agreements could be counted on. In many other parts of the world, this is not so. Treachery is applauded. Unfortunately, it is beginning to be that way in this country. Remember when gentlemen could shake on an agreement and it meant something?

- **Knowledge and Wisdom:** In order to be balanced and fulfilled, we need to be moving in the direction of growth and greater potential intellectually. Humans have an insatiable appetite for meaningful, constructive information. If that hunger is not being fed properly, we will become frus trated and less successful (all other things being equal). Remember when the United States led the way in high school SAT scores and graduated more engineers than lawyers?

The Fundamental Ingredients Of Success

If you are reading this paper, you are probably involved with computers in some way or another. Can be pretty heady stuff. You are bombarded with problems and information. If you are the typical DP person, you are working considerable hours which squeezes time from family, friends, hobbies, recreation, etc. The question many are asking is, *"What is this all worth ?" or "Am I living to work or working to live?*

I believe that the info-techno explosion has taken many folks (far too many) prisoner. The treadmill of trying to keep the data processing big wheels turning often robs us of true success. Many of our basic values are betrayed on the altar of materialism, pleasure, or progress.

In contrast to selfishness, success can be thought of more like an on-going commitment to worthy ideals rather than a "state of attainment". The three basic ideals that I am proposing in this paper are:

- **ETHOS** - Character or personal integrity.
- **PATHOS** - Compassion or the ability to genuinely value others.
- **LOGOS** - Knowledge and wisdom or the pursuit of truth and technology.

Many moons ago Aristotle proposed these three Greek terms as comprising the components of success for any venture or relationship. I am suggesting that each of us take an assessment as to how we run our DP organizations in light of Aristotle's success elements.

For our departments to be on the cutting edge of customer support (we all have a customer base now don't we...) I believe it is essential to consider how well we stack up to these principles. If we do not, it is likely that we will not last for the long run.

You may have had the experience of hiring (or worse yet working with) a person who lacked one or more of the following qualities. They were perhaps nice, but stupid; smart, but mean; a liar, but a caring person, etc. Occasionally you have also known a caring, knowledgeable, and honest person. What a breath of fresh air that kind of person is! I'll bet you considered that person to be fairly successful.

With that in mind, here are the three foundational success components.

ETHOS Explained

The root definition of this Greek word means "without deception, guile, or cunning". If you possess ETHOS it means those whom relate to you implicitly find that you can be trusted. You have principles and character. You cannot be bought. Consequently, you are morally consistent and predictable. Even when there is temptation for greater profit or pleasure, you do not cave in since you are pursuing long term success. This quality, though sometimes painful, provides great security for those whom you relate to.

Without ETHOS a person is literally building an empire on shifting sand rather than a rock foundation. As much as we may long for it to not be true, payday for our breeches in integrity will come. I am sure you can think of numerous examples of famous people in history who thought they could "get away with it". Building a life on that kind of thinking creates incredible vulnerability. And, when the chips are down, people know to keep their hand on their wallets when these kind of people are around.

Honesty is probably one of the most obvious manifestations of a person who has the quality of ETHOS. You know these people. They are the ones that look like they are in pain if you ask them to tell a lie. Are they silly Puritans? Maybe, but probably not. They have simply made the decision to be successful for the long haul. Consider the following:

> *"Even if we don't receive honesty in return in most of our daily transactions, as long as we never waver from our own deeply rooted values, the score will add up in our favor in the long run. This is one of the most basic, most obvious and, unfortunately, least understood principles of life. Good actions get good results in time."* [1]

The question people ask of you (whether you ever hear them or not) is *"Can I trust you"*. Think of a person whom you really respect and admire. Chances are that person has convinced you (verbally or non-verbally) that you can trust them. They possess ETHOS. I know many of you personally whom have worked to implement integrity in your lives and departments. Keep it up, it shows!

PATHOS Defined

The second foundational ingredient for a successful person is PATHOS. The basic definition of this term is "compassion". It speaks of emotional energy directed at another person with a twinge of human reverence. With respect to success look at what one researcher says:

> *"The most successful people are successful both in work and love. They tend to collaborate well and have good interpersonal skills. They know that when their relationships are in good working order they get more done."* [2]

What can I say? We all need to be cared for and we need to care for others. It's been built into the human machine "from the factory".

All the people whom you relate to want to know if you care about them. Your customers are wondering, "*If there wasn't a buck in it for her, would she be acting so nice?*". Your peers say, *"All he can think about is that stupid computer"*. Your kids are screaming, " *What's more important to mommy, us or that stupid job?*. I could go on...

Genuine pathos is becoming a scarce commodity, but each of us can make a difference.

LOGOS Unveiled

LOGOS carries with it the idea of words or thoughts. It implies technology. It also implies the right use of that technology; that's called wisdom.

The question people ask of you is, *"Do you know what you are talking about?"*. People want to know if they can, as your vendors, peers, customers, and friends depend on you to be a source of useful information and guidance (wisdom). The only way you can insure this is to keep yourself intellectually "fit". Consider the following comment by the management expert, Peter Drucker:

> *"Today, knowledge has power. It controls access to opportunity and advancement. Scientists and scholars are no longer merely on tap, they are on top. They largely determine what policies can be considered seriously in such crucial areas as defense or economics. They are largely in charge of the formation of the young. And the learned are no longer poor. On the contrary, they are the true capitalists in the knowledge society."* [3]

The only way to gain LOGOS is to study. A brain in neutral or simply being entertained can be stimulated to be more creative, but does not increase LOGOS anything like plain-old concentrated mind-effort does.

Denis Waitley, in his book Seeds of Greatness, says that one major aspect of real success is:

> *"...that a large vocabulary--which implies broad, general knowledge--characterizes the more successful persons, regardless of their occupations."[4]*

I have personally found that increasing my working knowledge of English has propelled me into levels and influence I had only dreamed of. Being able to understand, formulate and articulate thoughts and ideas is a necessary asset to long-term success. Consequently, I am a book-aholic. I have surrounded myself with thousands of books on every subject imaginable. I can see how being well rounded on the common subjects of life has gained me an audience with many groups of individuals.

Recently, I was invited by an Eastern European country to speak on the subject of running a business in a free market. On that same trip I had other occasions to speak to various leaders of that nation. While I do not say these things to try to impress you, I would like you to get my point. When I was a teenager I read an ancient Proverb that states:

> *"Do you see a man who is skilled in his work? He will stand before Kings." [5]*

I latched onto this little pearl and made a deal with myself that I would put forth great diligence to study and be the best that I could possibly be in whatever discipline I would get into. So I read, studied, and asked lots of questions. Every employer that I worked for before starting my own company, Lund Performance Solutions, would say that I gave great effort in serving them.

I know how much training and learning contribute to success. When I worked for HP I was amazed to see their commitment to training people. I have carried this as best as I can into my organization. While my firm is still very small (about 14 people), I have a commitment to see them grow in LOGOS. I send them to classes, I encourage them to read various books, listen to and watch tapes, etc. It's a win-win; good for them and good for my firm.

Did you know that only 5 percent of the people living in the United States will either buy or read a book this year? This is tragic. While other nations are cranking out hi-tech entertainment devices (TVs, VCRs, etc.), the United States is using those devices. Is that where the term "couch potato" came from?

You can be an integritable person with a heart of compassion, but if you fall short in LOGOS, you will not sustain the kind of quality, long term success that you could otherwise enjoy with increasing LOGOS.

All Three Ingredients In Perspective

You probably know folks who have ETHOS and LOGOS but they don't get along with people very well. On the other hand there are those that ooze with PATHOS but don't have the foggiest idea what their talking about when attempting to answering a support question. All three of these character traits need to be in balance for anyone to sustain long term success.

What About Exceptions?

No doubt you will think of exceptions. How about the manager that dislikes people, makes a lot of money, has a couples houses, and boats? Or what about the guy who moonlights giving away company secrets to competitors and who is raking in the money because of it? Or that aunt who repeatedly brags how much she cheated on her tax return last year and didn't get caught? What about your friend that had someone else take tests for him in college? I don't know about you but none of these satisfy my definition of success. Payday will come. And how about the nagging guilt (however suppressed it has become) that eats away at their guts?

Day To Day Operation Of These Three Success Ingredients

If you are a DP professional, there is a good chance you are fat on LOGOS but lean on PATHOS. As far as ETHOS is concerned, I've seen them all! Our Industrial-Technological culture rewards material progress... almost at any cost. This often causes us to pursue LOGOS (some) and to forget about ETHOS and PATHOS. Before you try to implement the following suggestions, consider that there is a great deal of momentum against your efforts. But press on, because you will enjoy a level of success that most around you will not.

Whether you are a manager or a "worker bee", if you do not take time to cultivate these principles but expect others to live by them, others will dub you a hypocrite. Then, the whole house you are trying to build comes tumbling down. I have heard that Napoleon said to his officers, "*If you want your men to bleed, you will have to hemorrhage*".

For ETHOS...

Just How does ETHOS "flesh out" in our everyday lives? In the home, my spouse can trust that I will make good on my implied and expressed commitments. My kids know that when mom or dad makes a promise, they can "take it to the bank".

Implicit to the very idea of ETHOS is some kind of ethic. This means that there really is a set of morals that have been woven into the very fabric of the universe, and, more specifically, our culture. While I realize this can be a very philosophical area, with everyone having an opinion, please hear me out.

When any civilization, business, marriage, or department within a company place privilege and freedom over responsibility and character, there is a slow burning fuse ignited to its demise. History is pretty clear that when past cultures consistently walked over the line of ETHOS, they began to deteriorate from within. Integrity says that I will not harm you even if I could profit greatly.

So in the department, you should take every opportunity to cultivate and reward integrity.

Tied in with the idea of ETHOS is a servant's heart. I have listened to many motivational/success tape sets. I am sure you have also. I am amazed at the basic concepts found within most of these series. When I purchased such sets, I was often looking for a silver bullet to propel me into prosperity and fame. Rather, what I found was a set of principles that manifested themselves in such attitudes as:

- **A Servant's Heart** - Realizing that my success is ultimately based on how well I serve others and feed their success.

- **Self-Discipline** - Giving myself to wanton pleasure without controlling my appetites simply leads to a decay of integrity. It means learning to enjoy life, but setting up restraints to keep my need for pleasure from getting the best of me.

- **Character** - Cultivating a personality that wins people because I listen and am faithful to whatever level of relationship I have committed to.

These qualities are some of the results that come from fostering ETHOS. So, here's some ways you can provide a fertile seed-bed for integrity to grow:

- Cultivate honesty especially in small things; it's much easier to justify taking a candy bar from a peer's desk if you have already been pilfering Hersheys Kisses...

- Encourage your staff to be trustworthy; in fact, reward it. When they stick to their word or admit they goofed instead of trying to justify their actions (lying), deal with the problem, but by all means reward their effort to come clean.

- Remind yourself of the following questions. Are you the same person when you are all alone as you are with others? When you hang with a person that you have been friendly to, do you find yourself mumbling something like, "What a jerk"? Can people trust you when they tell you a confidential remark?

For PATHOS...

Take time to listen to the verbal and non-verbal business and private concerns of each of your peers. How many times, when you are asked the question, *"How are you today?"*, is your reply of *"Fine"* really a lie, or a smoke-screen? Inside, you might be dying, but perhaps because you do not feel that others either care or have time to listen, you put up a "smoke screen". Pathos means compassion. This means taking time from my self-pursuits to listen to and love others. Here's some ways to practice this:

- Invite an unpopular peer to lunch... just to get to know him.

- If you are a manager, begin to listen to your people. What is behind the pain you see in their faces? Of course you are probably not a counselor, but you can listen. Such listen ing communicates love like little else. It is very hard to divorce home life from work life. Even when we try hard, forgetting about an abusive parent, a fight with our spouse, a prodigal child, or financial problems is, at best, difficult. That "baggage" comes with us to work. A manager who understands this will be propagating PATHOS.

- Begin to write sincere thank you notes to those who do little things that you appreciate. Believe me, you will not be able to fathom the response from people if you sincerely offer appreciation that is un-solicited.

- When you pass by a fellow worker and they ask, "*How are you?"* You say, *"Fine"*. Now this is probably a lie. Or maybe it's just a way to casually hide from that person. Maybe you are not convinced that person cares enough for you to deserve a "real" answer. Instead, begin to question their response. I usually say, "What does 'fine' mean?"

For LOGOS...

Keep in mind that the human machine was designed to have a near insatiable appetite for learning and growing intellectually. It can be fed or distracted. Feeding involves a plan - with goals, and reading... Distraction is the default these days. It is what most folks are doing (the 95% who don't read a book this year... I betcha they watch a truck-load of videos!).

I can think of no better way to advise you how to implement an increasing commitment to LOGOS into your life as well as those whom you influence than to quote Denis Waitley. He provides a number of action steps toward increasing LOGOS:

1) Continue your education regardless of your age.
2) When you read, always keep a dictionary beside you to look up a new word you don't completely understand.
3) Get a good vocabulary primer. Only 3,500 words separate the average person from those with the best vocabularies.
4) Consider taking a reputable aptitude test.
5) Model yourself after people you most admire and respect. [6]

I will also add a few other things that I have personally found helpful:

1) I rarely watch television. I avoid magazines and newspapers like the plague except occasionally glancing at headlines. Instead, I subscribe to some rather expensive newsletters that have consistently helped me be better informed on geo-eco-political happenings as well as giving me excellent investment advice. While playing sports can be healthful and recreational, I know too many people who are massively distracted by following the latest sports happenings. Funny, most of the guys I know who are addicted to such are mostly broke. Hmm....

2) I buy and read books on subjects that will particularly help me reach my goals and to better serve others. While I enjoy an occasional short fiction story or book, I ALWAYS avoid those scuzzy airport novels (most any recent novel, for that matter). Most people do not realize how subtly novels and science fiction can attack and even erode away ETHOS and PATHOS. PATHOS today has mostly degenerated into a sex-based mutant. While ETHOS is whatever is politically correct at the time (excuse me, I think I am going to vomit...).

3) I provide time for myself and employees to foster the growth of LOGOS on the job.

In Conclusion

While ETHOS, PATHOS, and LOGOS do not insure complete success, they do comprise the solid foundation for long term success. As the info-techno plot thickens we have to hang on to the basic values which make for the long term success for our marriages, companies, and nations. A renewed commitment to these three Aristotelian principles will help us be more successful in all of our ventures.

A Politically Un-Correct Postscript

As an overall suggestion for moving on in success, try reading the Bible. What! Did he say the "B" word? Call out the troops, he can't say that! Remember when the Bible was welcomed in school and porno was not? Interesting how our culture has degenerated.

Please hear me out. I am not bringing religion into the discussion. I personally have had bad experiences with organized religion, but was quite interested in knowing God, if He existed. Sometimes I feel the two are mutually exclusive! I am also interested in knowing what the manufacturer says about how to run and operate the human machine. I used to be a staunch atheist/evolutionist until I stripped man-made religion from the equation and personally researched the Bible and the claims of Jesus Christ. Since then I have walked with God for a good many years. You couldn't pay me to buy into man-made religion, but I cannot tell you what an adventure it has been to personally know the living God! I have found that He is very interested in my life, business, family, etc. In fact, I have applied numerous literal principles out of the book of Proverbs in the Bible and have seen unbelievable results! Please don't "flip out" at my suggesting it but you'll find the success principles I have written about in this paper strewn throughout the Bible. If you are really interested in long term success I dare you to look up in a Bible the following success references:

- **Joshua 1:7,8;**
- **Deuteronomy 29:9;**
- **Psalm 1:1-3;**
- **Jeremiah 17:5-8**

I've been living this stuff for twenty years...ask my employees, customers and friends. It's hard to argue with success!

References

[1] Waitely, Denis, Seeds of Greatness (New Jersey: Revell, 1983), pg. 86

[2] Charles A. Garfield, Director, Peak Performance Center, Berkeley. Reference unavailable.

[3] Drucker, Peter, Age of Discontinuity (New York: Harper and Row, 1968), pp. 372,373

[4] Waitely, Denis, Seeds of Greatness (New Jersey: Revell, 1983), pg. 95

[5] Proverbs 22:29 , New American Standard Bible (California: The Lockman Foundation, nd).

About The Author

Robert Lund is the president of Lund Performance Solutions, which specializes in system performance software, consulting, and training for the HP 3000. Robert is the author of Taming The HP 3000 Volumes I and II. He has designed and led development of three major software packages, SOS/3000 Performance Advisor Q-Xcelerator Resource Manager and FORECAST/3000 Capacity Planner. He has been teaching seminars in the U.S. and Europe since 1988 and is considered an expert in the field of HP 3000 performance.

If you would like to discuss this paper or other issues with Robert, he may be reached at (503) 327-3800

MAINTAINING A QUALITY STAFF STARTS DURING THE INTERVIEW

JEFF ODOM

BAHLSEN INC.
ONE QUALITY LANE
CARY, NC 27513
(919)677-3227

"One job belongs to the leader alone; and that is making sure all the parts and all the people work together."[1]

You can have the best parts in the world - yet the truck you build may not run well, if at all. You may have the best flour, sugar, spices, oils, etc., but the cookies you bake may not taste good and may be hard as a brick. How is it people can shop at the same supermarket, buy the same ingredients and use the same recipe yet the final product be as different as the east is from the west? The functioning of people, their interrelationships, skills, motivation and quality consciousness bring the ingredients together and produce quality goods and services.

We are responsible for blending associates to create a quality team. "In a system everything has to work together harmoniously. The parts and the people have to be optimized to work with each other toward a common goal."[2] You must be sure to fit the right person into the right position by viewing the whole employment setting and pay particular attention to intangibles. Success will always come down to intangibles of fit, personality and relationships.

Unfortunately, key organizational ingredients may be missing if your company does not subscribe to philosophies that create fertile environments for associates to develop. Even so, you must do what you can to create a divisional, departmental or workgroup environment which will promote such associate growth. Some of these concepts include full employment without fear, training, quality, mobility organizationally and so forth. People want to perform in a manner that provides them the satisfaction of a job well done. The work place must allow associates to perform to the best of their abilities without fear of reprisal and be fully trained to perform duties as assigned.

Quality work relationships, products and services depend upon individuals filling appropriate roles on a team. Our world is full of examples; such as consistent top flight college basketball programs built on quality team defense. Another favorite example is of an orchestra. In an orchestra, each member is a qualified and talented musician, but to perform as a team each must fill their role at the appropriate place and time under the conductor's leadership (coordination). If each member of the orchestra were to try and highlight their individual skills, the harmony which creates the beauty of the music is lost.

"Quality is pride of workmanship or joy of work. By allowing and even urging workers to experience intrinsic rewards that come from doing something well, using their innate and acquired abilities, productivity improves, quality improves and customer satisfaction improves."[3]

To be successful, individuals generally have a need to do quality work and to have the opportunity to develop and fit into a defined work environment. For individuals to succeed, they must volunteer their heart to the effort. The key to this volunteerism is relationships. Talent, technical skills, motivation, education and so forth are important qualities, but the ultimate success lies in the relationship between supervisor and subordinate. To that end, the relationship starts during the hiring process and it is the supervisors responsibility to predict candidate success during the interview.

We may say that the keys to successful hiring are:

1. Associate and supervisor fit
2. Organizational philosophies
3. Skill, experience, education fit
4. Team fit

If you notice, we as management, control or highly influence points 2, 3 and 4. We, to a varying degree, influence corporate philosophies and should be responsible for our own team development. As for point three, we select the appropriate skills or they are obtainable by providing education and training to associates. All this leaves us with two dilemmas of making sure we truly have the appropriate skill set and more importantly how to determine supervisor/associate fit.

May I suggest a methodology that has worked very successfully for me and was derived in large part by the teachings of Dr. Kurt Einstein noted behaviorist. Dr. Einstein taught his methodology in a seminar all over the world called "Picking Winners." The following is how I use his method incorporating processes influenced by other behaviorists as may be noted.

The basic process is:

1. Define the position to be filled

2. Determine a candidate's
   a. must haves
   b. preferred to haves

3. Advertise or network for candidates

4. Preinterview planning

5. Interview

6. Second interview if necessary - or is it always necessary?

Defining the position then developing the must and preferred to haves is critical to the selection process. The definition determines who will be interviewed. We don't want to exclude a qualified candidate and we also cannot spend precious time interviewing candidates that do not fit our needs. Further, once hired, this definition is given to the associate as a clear representation of what they will be held accountable for.

To define the position we must determine the following:

1. What will the person do?

2. What will their responsibilities be?

3. What decisions will the individual be allowed to make in 3 months, 6 months and a year. This is an important process in developing associate/supervisor trust. Robert Tannehill

wrote of the importance of this point as it pertains to managers: "For an organization to give managers freedom to make mistakes, to grow and to learn, there must be a fairly high level of trust and confidence among the managers one with another, and especially on the part of top management. When a mistake is made, it is often the practice of management to scurry about to try to find out who is to 'blame' for the error."[4]

I always tell associates immediately you have the right to make mistakes, your obligation is to communicate to the appropriate individual(s) as soon as you realize the mistake. By setting out clearly when decisions are the associate's responsibility, there is comfort and greater ease in the decision process.

4. What technical skills and education does the position need?

Today we have two additional dilemmas that face us in the hiring process that truly complicate our lives. The first is the effect of the American's With Disabilities Act which forces us to take certain steps to insure that disabled individuals are not discriminated against. It is imperative that all individuals in your firm that participate in the interview and hiring process receive training in the applicability and demands of this legislation. For purposes of this discussion, we want to specifically remind ourselves that during the interview, you can not ask any questions about the existence, nature or severity of any physical or mental impairment of the individual. Further, this same prohibition exists pertaining to questions of this nature about the candidates's relationships and associations with others who are disabled individuals.

The second major issue we face today is dealing with the deluge of resumes we are likely to receive based on the current structural shifts in American business. What can we do to increase our probability of selecting the right candidates for interviews? In particular, do not exclude individuals who have been laid off. Today, an extremely qualified group of potential associates have been put in the job market due to organizational change. Some other organization's restructuring fall-out may actually give you access to excellent individuals that previously would never have entertained a discussion with you.

Both of the above makes it imperative that you define the position accurately prior to beginning your search. Not only preparing a well thought out list of must and preferreds, but job descriptions and performance expectation levels must be accurate and current. Failure to do so can possibly lead to trouble complying with the American's With Disabilities Act and/or prohibit you from making the right selection.

Do all you can to gain access to the right candidates. Start today seeking contacts for future positions. Exchange business cards and information with anyone you meet who themselves exhibit attributes you seek or may be helpful in finding candidates. Help your human resources department create a candidate network by free exchange of information internally and externally.

Having done the necessary steps to carefully construct and understand our position, received and critically reviewed resumes and set appointments with qualified candidates, we move to the extremely important process of interviewing candidates. The interview process is an evaluation process - there is a mutual decision being reached. Both the candidate and the interviewer (organization) have to reach a quality decision as to fit. Dr. Einstein taught in his seminars that there are two reasons why personnel selection decisions fail:

1. Failure to evaluate fit of selected person.

2. Failure of selected person to evaluate fit (confusing specifications or poor understanding of specifications).

If we understand what causes personnel decisions to fail, we then can define the goal of the selection process. Simply, our goal is to avoid the failure issues by increasing our ability to make predictive selection decisions. To do this we want to evaluate:

1. The capability of the person. What a person can do leads to performance. What a person can explain to us in detail they can do.

2. The personality of the person. What a person is like leads to behavior.

Knowing these two points, we will be able to predict how a person will perform and act. Prior to the interview, review the candidate's qualifications to determine strengths, weaknesses and fit to your criteria. Determine appropriate questions that will validate the strengths or true attributes and the extent of perceived weaknesses. Remember we all have strengths and weaknesses so our objective is to make an informed decision as to the candidates fit given both. We are not looking for someone who has no weaknesses.

To make this evaluation, we want to conduct the interview as follows:

1. Set a comfortable environment overtly free from interruptions and distractions.

2. Ask for agreement in the process. Ask may I take notes? Do you mind if we sit here?

3. Explain that this is a mutual decision process and the penalty for error is high for both parties. Further, both parties are responsible and accountable for the decision.

4. Direct the interview in a fashion that you ask all of your questions first then allow the candidate to ask any and all questions. Have questions flowing one way.

5. Ask open-ended questions. Avoid yes and no responses.

6. Ask why, why, why! Probe pat and standard response by exposing the reasoning behind them.

7. Ask revealing questions as described below.

One of the problems in interviewing is determining what questions will stimulate responses that will most add value to your decision process. What is proposed here is that you ask open-ended questions to which the response will tell you about the true capabilities and personality of the candidate. Below is a list of example questions from Dr. Einstein's seminar as well as those other successful interviewers in our organization have used.

1. What would you describe as the best attributes of your current supervisor?

2. Tell me about two serious interpersonal relationship problems you've had on your job(s).

3. What are your likes and dislikes? These are from all phases of life; personal and career.

4. Give some examples of situations where you have been criticized. How did you react and why?

5. Under what conditions do you feel you learn the best?

6. To what extent do you feel that your college grades accurately reflect your ability?

7. What do you feel are your three most significant accomplishments? In detail describe how one of them was achieved.

8. Describe the perfect boss.

9. Under what conditions would you consider yourself as "having arrived" in your personal and professional life?

10. How do you resolve family differences?

11. Describe the perfect work environment.

12. Based on the brief time we have spent together, why do you perceive this is an organization you would like to join?

13. Describe a past situation in which you have had to criticize a subordinate. Explain the situation and how you handled it.

MAINTAINING A QUALITY STAFF
STARTS DURING THE INTERVIEW

14. What would you describe as the most positive attributes of your current organization?

15. To what extent do you feel your current position has prepared you for additional responsibilities?

These questions give you a flavor for the types of things to ask to discover who the person is and what their capabilities are. For technical aspects, you would key in on accomplishments and the detailed description of what took place. When seeking a supervisor, look for questions that reveal personal behavior skills (so called soft behavior). In all questions, continue to seek answers which will allow you to make the best predictive selection decision you can.

The complexity of interpersonal relationships is vast. Our probability of success as organizations increases as we select the best candidate for a given position. It is important that we continue the development of all associates once hired. Give your staff an opportunity to take responsibility of their own future by empowering them with training and security.

[1] Rafael Aquayo, Dr. Deming the American Who Taught the Japanese About Quality (Carol Publishing Group, 1990), p. 180.

[2] Ibid.

[3] Ibid, p. 45.

[4] Robert E. Tannehill, Motivation and Management Development (Auerbach Publishers, 1970), p. 81.

Paper # 3032

Software Acquisition -- Getting it Off the Ground

Robert G. Boynton

Warn Industries

13270 S.E. Pheasant Court

Milwaukie, OR 97222

(503) 659-8750

Introduction

Many of us are faced with the situation where the current applications software is no longer viable. When evaluating the need to replace this software is often becomes apparent that the best solution, both in time and money, is to acquire a third-party software package.

Initial evaluation of the task before you can lead you to become somewhat intimidated by the task itself. If you are attempting to acquire a complete replacement to your existing applications software, the very magnitude of the project may seem insurmountable.

We have all read and utilized various project management and design methodologies or tools with varying degrees of success. Most of these methodologies are built around the premise that you are going to enhance and/or replace either an existing software or manual application. When attempting to develop and manage a project that effectively replaces the organization's total applications software, these methodologies seem to fall short of practical solutions.

In this paper I will attempt to show you some of the techniques that work, as well as some of the techniques that do not work. This is a step-by-step approach to making the project happen, while identifying some of the aspects that can consume time without much benefit as well as some of the pitfalls.

A Little Background

Several years ago, Warn Industries had a mix of software. Our manufacturing software was from one vendor, our financial software was comprised of purchased software modules from several vendors as well as in-house developed modules, and our order management and distribution software was totally in-house developed.

While each of these solutions met the needs of the given area of the organization, the need to integrate the various aspects of the organization was readily apparent. Highest on the list of needs was the integration of our manufacturing with order management, so that there was both order visibility within manufacturing (actual demand) and there was production visibility by order management ("available to promise"). Next on the list was the integration of the applicable financial modules (GL, AP, etc.).

To address our most pressing need, we attempted to implement the Order Management module available from the vendor of our manufacturing software. After several attempts to change our policies and procedures to adapt to the software, it was determined that the system was inadequate for our needs. Additionally, there was no foreseeable improvement forthcoming from the vendor.

As a result of this determination, we decided to drop this project. Instead, we created a project to evaluate, acquire, and implement a third-party "integrated" software system. This new system would encompass order entry, manufacturing, sales, distribution, cost accounting, general ledger, accounts payable, and accounts receivable at a level that was acceptable to the user within that area. We would also consider any other software that the vendor could provide at this level of acceptability. In short, we decide to replace our system.

Because of being unable and/or unwilling to commit to the project at the proper level, this project went along for several years with little being accomplished. However, once we hit upon the right method for success, we were fully implemented thirteen months after our first vendor demonstration.

The following is a step-by-step approach on how to get the process flowing, with some of the pitfalls to watch out for.

Identify the Need

This process is so obvious that you may be inclined to skip this step. You may already think you know what the need is. Even if this is true, the need must be identified by the right people in the organization. Although the decision-makers may be aware that there is a problem, their perception of the problem may be such that they may focus support and resources in the wrong direction.

It is easy for the user community to see a single problem area and perceive that the whole system needs changing. It is equally easy for the user to believe that a minor change will correct a system that is totally inadequate.

The main focus in this step is to identify the need, and for all appropriate individuals to agree on that need.

Define the Functional Areas

Once the need has been established, you should then organize several focus areas along a hierarchical structure in order to make the project more manageable.

In our case, we chose to organize our project along three broad areas:

Manufacturing - comprised of:

- Purchasing
- Planning
- Bill of Materials
- MRP
- MPS
- etc.

Sales and Marketing - comprised of:

- Order Entry
- Distribution
- Accounts Receivable

Finance and Accounting - comprised of:

- Accounts Payable

Fixed Assets

Payroll

etc.

The main reason for organizing around this kind of structure is to allow for easy focus by individual project team members. Even though there will be some significant cross-over between the various areas, it can be managed within this structure because on of the main tasks in each area will be the affect of one module on another.

Select a Project Team

After the functional areas have been defined, it is necessary to select a project team. It is imperative that you select qualified and motivated individuals because this is the group that will make or break the project.

The most important member of the team will be the project leader, because it will be this person's job to devote as much time to the project as necessary, which in certain stages will be 100% of his/her time. Despite the standard philosophy that the project leader (in the evaluation phase especially) should not come from Information Systems, this is probably the most likely resource because it may be possible to commit this person full-time to the project, and this person will have access to the technical resources that may not be available to individuals in other areas of the organization.

Once the project leader has been established, you must now select your team members. We found the best solution was to ask for volunteers in a company-wide request, with the final selection being made by a management group, keeping the team size at a manageable level. This allowed us to get motivated individuals, while at the same time selecting qualified people. An added benefit was that by selecting that individual, the management group also gave a commitment to support those involved.

If you do not get a volunteer from a critical area, recruit him or her. Do not make the mistake of not representing every critical area with your team. If this happens, the lack of representation will cause the project to bog down when decisions cannot be answered by the core members.

Each of the members of the team should then establish a "sub-team" made up of users of the given functional area to advise and evaluate this area in detail. This group should consist, in part, of those not selected for the main team.

When the team has been established it must be empowered. This empowerment is two-fold. First, the team as a whole must have the resources and authority to make all of the necessary decisions right up to the final selection. Second, the individual project team members must have the authority to speak for their respective functional areas.

Define your Current System

It is necessary to know what you have in order to best identify what you need. Thus you must document and quantify your current system. Do not waste time and resources in defining the "look and feel" of the current system (screens, reports, etc.). Instead focus your resources on what it is and what it does.

Identify the "global" data, features, and functions of every area. Next, identify those elements that cross over to other functional area. It may be that something that is considered unimportant for one area is critical for another.

Finally, quantify the value of each of these elements (must have, like to have, do not want). This establishes the base parameters for your next step.

Define Your Ideal System

Using the evaluation of the current system, define the functionality of your new system. You are not trying to design a system, you are merely trying to fill in the gaps from your existing system. This establishes a baseline for your software evaluation.

The project members, by having a realistic view of what they currently have, can now describe what they want from a more educated perspective. This step should be "pie in the sky" time. Because they have not looked at any software as yet, it may not be based upon anything that is available, but will be a good definition of what they wish they had.

Develop Checklists

Turn your ideal system definition into a checklist. This checklist will be used to evaluate the software that you will see in subsequent stages of the project.

Using this checklist, develop a simplified "features and functions" checklist of the high level aspects of you system. This will be utilized to select which vendors to pursue. This high level checklist should not be more than a few pages long.

Traditionally, a version of the first checklist is used to develop a RFP (Request For Proposal) to be sent to potential vendors, if one is to follow the standard project methodologies. The problem with this is that the checklist is rather long and detailed. Even if the vendor is willing to spend the time and resources to respond to this kind of document properly, they will need a significant amount of time to evaluate it. Once the responses are returned, it takes a great deal of time and resources to evaluate them. This can be more detrimental than beneficial because it can defuse the projects momentum.

By developing a "features and functions" checklist, you will be able to get quicker, more honest, turnaround from the vendors and it will not take forever to evaluate. We discovered, as an added bonus, that some vendors bowed out of the process immediately when they saw the list. This kept both the vendor and ourselves from wasting time.

Solicit Vendor Responses

Develop a packet to submit to potential vendors. This packet should contain a cover letter, company history, and the checklist. The cover letter should explain the current software and hardware situation. This letter should also request a response to the checklist in a reasonable period of time. If the checklist is concise enough, two weeks should suffice.

Selecting the potential vendors is critical. You must evaluate the product offerings as described in any unsolicited brochures as well as from any number of software catalogues. You can also ask users and personal contacts for potential vendors. Be careful to select only those that you think can solve your software problems. As the vendor will become a "partner" for many years to come, it would also be helpful to find out about their financial stability (e.g., get a "Dun and Bradstreet" analysis).

When looking for potential vendors, do not forget to consider the price of the software. There is no use looking at filet mignon when you can only afford hamburger. Only consider the software price from a global prospective, because the advertised price of the software can be extremely flexible.

Unless there is something significantly lacking in your hardware, do not make this an issue. Select the majority (if not all) of your vendors from those that offer solutions on the platform you currently have. Only if you cannot find solutions on your current system should you expand the search to solutions on other platforms. This is primarily due to cost and training. Why spend money on new hardware (and the associated training) unless absolutely necessary.

Be sure to select enough vendors so that you have a high probability of finding the appropriate software, with several different approaches to solving the same problems. At the same time, do not select so many that this process becomes unmanageable.

Finally, do not send this solicitation to the vendors "cold". Instead contact the vendors beforehand. Explain your situation and request that they meet with you at your organization to receive the documents. This gives both of you a chance to meet and discuss the issues globally. It also gives the vendor an opportunity to tour the facilities and get a feel for the company.

Evaluate the Vendor Responses

Upon receipt of the vendor responses (following up with the vendors who did not respond), perform your evaluation. This will probably entail several conversations with the vendors, answering questions they had as well as getting answers to questions that have been derived from their response. It is important to resolve all questions before going on.

Once all of this has been resolved, determine which vendors to pursue. If they obviously do not meet your minimum requirements, do not waste both your time and theirs. At the same time, if you are not sure, keep them in the process.

Initial Software Demonstrations

Working with the chosen vendors, arrange for preliminary demonstrations. At this point of the process, the demonstrations will be "canned", showing what the vendor does best. This is too early to get an in-depth demonstration from the vendor.

Arrange for the demonstration to take a full day. This is so that you can selectively focus on one broad functional area at a time. This keeps the participants interested and involved. The participants should include both the project team members and the active "sub-team" members.

It will be necessary for the evaluators to do their homework. Using their checklist, they should have developed a list of critical issues that they want to see. This list should be used in the evaluation of every vendor.

Narrow Your Selection

After all demonstrations have been completed, compare all of the vendors' offerings in relationship to your checklist and in relationship to each other. Not all of your questions will be answered, but enough of them should have been to eliminate the software packages that were obviously deficient.

If all of the vendor offerings were deficient, solicit participation from additional vendors. Ideally, you should have 2 or 3 strong contenders at this point. If you have more than that, you have not been selective enough. If you have only one, you need to get another one so that comparisons can be made.

Follow-up Demonstrations

Working with the project team, develop a list of functions and features you wish to explore in more detail. Include in this list anything that you did not see or that you still have questions about.

Provide this list to the vendor. In conjunction with this, provide the vendor test data made up of sample data from your current system. It is critical that you do this in order to see how their system works with your data.

Arrange for the demonstration to take two days. The first day (and possibly part of the second day) should be set demonstrations to explore those issues arising from the first demonstration. The rest of the time should be set aside for more in-depth exploration of any pending issues.

Be sure that you control the format, timing, and tempo of this demonstration. This is not the vendor's show at this point. Have the vendor adhere to your agenda so that you get your questions and issues resolved to your complete satisfaction.

At the conclusion of this stage, you should be able to eliminate any vendor that cannot meet your minimum requirements. If no vendor meets those requirements, you must solicit the participation of additional vendors.

Ideally, you should have 1 or 2 vendors that seem viable at this point. It should now be a matter of confirming what they have said, and making the best choice for your organization.

Site Visits

Now is the time for site visits. The most obvious form this takes is for the members of the project team to visit sites that are using the software in question. You must work with the vendor to find organizations that are similar in size and industry so that the team members can properly relate to the information they will discover. You must once again insure that you control the situation. You must insure that your team members have an opportunity to meet with their counterparts at the other companies, preferably without the vendor representative participating. This is so that you can get more candid responses to your questions and concerns.

Two other types of site visit are also extremely helpful. The first is to send selected Information Systems personnel to the vendor's home office to evaluate the support from a technical point of view. The major focus will be the quality of their response center (phone-in support), the vendor's ability to respond to software customization, and the future directions the vendor is going both in functionality and technology.

The final type of site visit is for a selected group of "top management" people to visit the vendor's home office to meet with their management personnel. If chosen, this vendor will be in "partnership" with your company for a long time, so that some compatibility in future directions should be insured. It is also important that your financial representative evaluates the vendors financial stability at this juncture.

Selection and Implementation

At this point it is time to make a decision. Based upon the recommendations derived from the site visits, you should be able to select a "winner". Now the acquisition can take place. As part of the acquisition, the implementation time lines and goals should be clearly defined.

Utilizing the same people who participated in the selection phase and the representatives from the vendor, you now can establish an implementation plan utilizing the available resources. This plan is unique to every organization/vendor combination, so there is no guidelines for this other than those developed by the specific vendor.

Some General Thoughts

Be sure to communicate with your potential vendor at every aspect of the process. This includes detailing reasons for dropping them from further evaluation. They have invested a great deal of their resources on you, therefore they deserve the courtesy of your detailing the reasons they were not selected.

At some point in time, it is possible that the project may bog down because of lack of direction or focus. This is generally due to such things as of certain individuals seeking out "territories" or the general lack of commitment on the part of some individuals. If this occurs, you should seriously consider acquiring the services of an outside consultant. This serves many purposes. The consultant is usually able to keep you focused, they have no vested interest in personalities, the can tell users things they do not want to hear but must hear, and people make time for a consultant when they will not make time for a fellow employee. This consultant does not have to Information Systems oriented, but should have a well-rounded background in helping companies similar to yours. The selection of this consultant should have all the thought and evaluation that you used in selecting anything else related to this project.

User involvement is imperative. Not only will user involvement aid in the selection of the proper software, if will also aid in the implementation. In our case we were completely implemented six months after acquisition. It was almost totally the result of the quality of involvement by the users. We had almost 15% of the employees of our company involved in the project and although it was unwieldy at times, the ultimate benefits outweighed any short-term problems.

Conclusion

By utilizing the project management and design tools that you may be familiar with, you will spend a great deal of time on aspects of the project that are unnecessary if you are going to acquire the software from a third-party. Most of the management tools teach you not to make this decision until you are well along in the project. Why waste time an resources when you know that you are going to acquire a system when you start the project? Instead, utilize the resources you have in getting the software you want and implementing it in the most timely fashion possible. By following the steps I have outlined, you should be able to get the project flowing to a successful conclusion in a reasonable period of time. Once again these were:

Identify the Need

Define the Functional Areas

Select a Project Team

Define your Current System

Define your Ideal System

Develop Checklists

Solicit Vendor Responses

Evaluate the Vendor Responses

Initial Software Demonstrations

Narrow the Selection

Follow-up Demonstrations

Site Visits

Selection and Implementation

Capacity Planning Part-Time

Javier Diaz
Systems Operation Manager
Goodyear Oxo - Mexico

One of the tasks that a Systems Operations Manager has to perform, is capacity planning. Whether there is ample resources devoted to this task or it is expected to do it in 'spare time', this task cannot be overlooked. This paper is directed to the people requiring to perform capacity planning on a limited budget.

First we will try to define capacity planning. I like to define capacity planning as the technique that will allow us to satisfy our clients expectations regarding systems services. This effort is divided in three phases:

1. Knowing our actual workload by application with our system adequately tuned.

2. Forecast our future workloads. Consisting of the growth of our current applications as well as the estimate of resources needed for future applications.

3. Predict future utilization of our system according to the future workloads.

It is a good idea to measure our CPU performance regularly and publish that data to our users community. They will become good allies when trying to get approval for a CPU upgrade or more disk space or memory. This data should be kept on hand as a basis to determine future growth, although this should be adjusted by our users growth plans and system development projects.

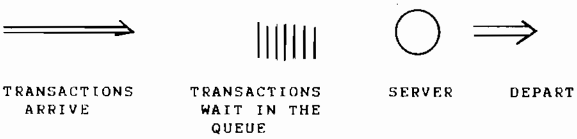
Fundamental to our capacity planning effort is the definition and writing of our service level agreements. Being it response time, service hours, backups frequency and schedule, jobs turn around time, etc. Without service level agreements we have unlimited capacity since there is not a minimal service level to attain, on the other hand users will almost always expect better service that what we can deliver according to our resources. High management should always know the limits and capacity of our computer system.

Knowing the systems development plans is also very important for the capacity planning process. Is there going to be third party programmers in our installation? When are systems scheduled to be productive? How many users will access the new systems? What will be the space required for the new files and data bases? These are questions which the operations area should always know the answers. Don't forget this since the impact of systems development on the data center is frequently underestimated.

Once we have a good grip of how our system is utilized through periodic measurements, and we have a fairly confident estimate of how the demand for computer resources will grow for the next one to two years, we can proceed to the third phase which is to perform an estimate of how well we will satisfy our service level agreements in the following months.

Queuing models.

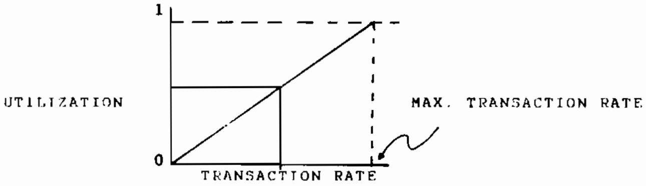
To be able to predict the future behavior of our system, the most commonly used technique is modelling. And the model that best represent a computer system is that of a server attending a service queue.



The parameters of this model are: average arrival rate, average service time, average waiting time and response time (queuing time + service time). The utilization is the fraction of time that the server is busy. From this model, the following formula applies:

UTILIZATION = TRANSACTION RATE X SERVICE TIME

This formula is very useful, since if we could find the utilization and the transaction rate we can calculate the service time from the last equation. Or we can find the maximum transaction rate possible for our system since the utilization cannot be greater than 1.



Another useful formula is the following:

$$\text{RESPONSE TIME} = \frac{\text{SERVICE TIME}}{1 - \text{UTILIZATION}}$$

From this formula we can see that as the utilization approaches 1 or 100%, the response time goes to infinity.

Once we have seen some basic theory, we will go to the practical use. We will base our analysis on HPGLANCE which is a software easily affordable for installations with a small budget, when you become more proficient in its use you may want to get a package with more options.

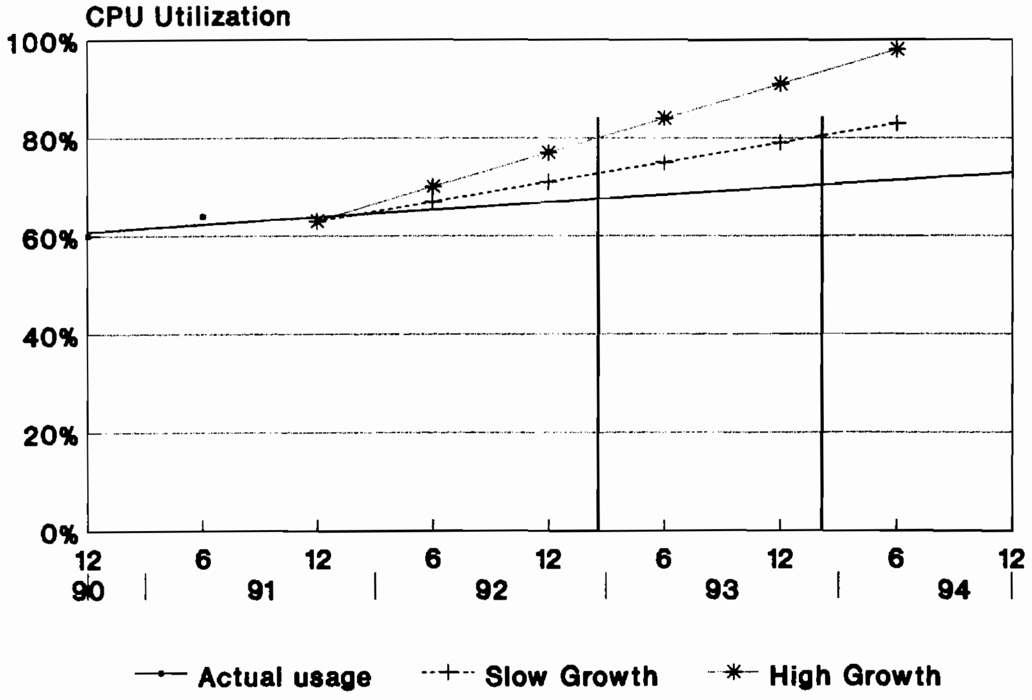
The first thing, is to determine our average CPU utilization. The CPU utilization is very dependent on what time of the day we are measuring. We should focus in our very peak hours, I personally measure only the three peak hours which in my case go from 9:00 to 12:00. You should find your peak hours performing measurements every half hour during a whole day, later you can focus only on your two or three peak hours. Remember always plan for the peak hours not for the hours when you have excess capacity. Once you have find your peak hours, you can set HPGLANCE to print a report every five or ten minutes and average the CPU Utilization. This data is already given to you as a percentage at the right end on the CPU BAR.

One way to do this automatically, is to set on the print option of HPG and set the refresh time to 10 minutes, also set the treshold to show processes consuming more than 0 CPU. You need this to see any active process.

You can save this data on a spreadsheet for future reference along with the average number of active users on the system at the time, you can get this data also from the HPGLANCE report.
With this information you could draw a graph similar to the one in figure 1 using the average number of active users instead of the transaction rate. This quantities although not the same, are directly related. And now you have found the maximum number that your system can handle at the 80% threshold. I would advise not to go beyond the 70% threshold in order to have some room for unexpected sudden growth.

If you store this information over a period of time, you can draw a graph of current utilization and future forecasts of future growht according of what you find in the step 2 of the planning process. The graph should look something like the following:

Projection of CPU usage over time



In this CPU utilization graph, we have plotted the CPU averages for 12/90 which was 60%, 6/91 which was 63% and 12/91 which was 62% from this data we extrapolate a line that shows a small CPU usage over time which don't get to the 80% threshold in our time frame which goes to 12/94. Additionaly we have plotted a dotted line which includes a little CPU increment due to perhaps a new application, this dotted line will cross the CPU threshold approximately on February 94 so we will need to order a bigger CPU and expect to receive it prior to that date. We also have plotted a line that includes a bigger CPU usage growth, that line crosses the 80% threshold on March 93. So we should hurry up with our justification and budget approval to be sure we have the new box installed by that time in order to avoid response time problems due to CPU saturation.

You need to show the actual CPU utilization trend as well as the minimum and maximum growth forecast. So you can see when in time you are going out of resources according to the business growth.

As you progress doing projections, you may find the time to do the same analysis for the disks and memory. Due to the relative low cost of memory upgrades compared to other resources, I would advice to get extra memory whenever you guess that it is memory that you need. You can determine that if your memory bar approaches 100% and your disc bar takes more than 5-10% in memory management (swapping) transfers. I wouldn't advice to spend a lot ot time analysing memory due to the relatively low cost of it.

Buying another disc will almost allways will be justified on a disc space requirement for new systems instead of getting getting better response time, so I wouldn't spend a lot of time analysing disc response time just keep your data files adequately distributed along your disc packs.

The last step and perhaps the most crucial, is presenting your forecasts to higher management and getting your purchase approvals. Try to show graphs relating the business growth with the CPU growth, and make them aware of the lead time in the delivering of new CPU boxes.

Paper #3034

Project Management
Begin By Sticking To The Basics

By: Jeff O'Donnell

DesignAlyze
Corporate Consulting and Training Services
4163 131st Street
Minneapolis, MN 55378
(612) 894-0213

Introduction

Computers have been in existence within organizations for nearly 30 years now, and still many of the information systems being built today are classified as failures. Systems often fail to meet the budgetary, as well as time estimates given at the beginning of the project life cycle. Even upon final completion of an over budget project, systems often do not meet the needs and expectations of the clients. These so called failed systems often require major modifications to even become acceptable.

This problem of failed systems is of great concern to management. Information, and quick reliable access to this information, is a means of providing feedback. Information can also be used as a medium of control in planning, and assist a company in keeping in step with an ever changing environment. Beyond this, information technology can be utilized to help reduce costs, increase productivity, market share, and/or revenue. These combined factors are often critical in insuring the competitiveness of an organization.

Many factors must be considered when planning for the design, development and implementation of systems hardware and software. These include the organization, task, environment, personal and interpersonal characteristics of clients and information services staff, and finally the policies and procedures which surround the system development life cycle. It is the responsibility of the information services Project Manager to consider all the variables surrounding a project, and be prepared to handle the many challenges of systems implementation.

This paper focuses on two functions which Project Managers must always carry out during the course of the project development cycle. Although these are relatively basic functions, overlooking them could lead an otherwise successful project down the path of failure. The first function is control. Project Managers must at all times remain in control of their projects. Controls both internal to I/S, as well as external are key and must be properly utilized and managed to insure project success. The second basic function is communication. The interpersonal or behavioral aspects which come into play during a project life cycle are diverse and wide spread. Project Managers through effective communication must anticipate and defuse the variety of human behaviors that can come from many different people throughout all areas of an organization.

Control

Control of projects can be maintained within three areas of an organization. These areas include top management; information services management, and finally the client community. It is the responsibility of the Project Manager to insure that control of projects stays within the information services organization. This can only be done if a balance of control is maintained between top management, Project Managers, and the client community. Only by maintaining this delicate balance of control will Project Managers be able to truly maintain control of their projects.

Top management has direct control over information services in several ways. These include the top management activities of reviewing and authorizing major new systems work, review of system costs, post implementation reviews, review of the information services organizational structure and internal control practices, and finally overall performance monitoring. If top management is not satisfied with the state of affairs of any project, it is very easy for them to step in and take control.

Top management is continually getting more involved in the information services arena because of the growing complexity and ever increasing costs of systems. Computer information systems are now assuming a greater strategic role in business and industry as competitors strive to use them to improve market share and increase profits. Project Managers must work to make sure the systems planning function is being carried out so as to best meet the needs of the organization. This can only be done by effectively understanding the business needs of the organization and being in tune with the strategic plans of top management. All systems must conform to company standards and policies in order to be effective. Project Managers must think like businessmen and not just as technologists. This will help insure that each project undertaken is to help carry out the business needs of the organization and not just for the sake of technology. If this is successfully done then Project Managers stay in control, and top management does not feel the need step in and take over the course of a project.

Project Managers are responsible for managing the personnel resources of an information services organization. Effective management of people is critical to project success. Project Managers must continually focus on building and maintaining a qualified staff, effective scheduling of work, and proper assignment of resources if they are to maintain control of their personnel resources and their projects.

Policies and procedures which surround the system development life cycle are a third resource available to Project Managers to maintain control of their projects. These policies and procedures basically consist of two types: (1) processing controls internal to information services; and (2) external controls independent from the information services function.

Internal controls are concerned with the integrity and proper handling of data during processing, while external controls consist of review and verification of computer generated data. It is the responsibility of the Project Manager to insure that the proper internal controls are in place and being utilized.

Giving clients a sense of control over projects will often go a long way to insure a smooth and successful implementation of new systems. Project Managers can assist clients in maintaining an acceptable level of control by: (1) giving clients a complete and accurate picture in advance of their likely experiences during and after system implementation - make it predictable; (2) find areas in which clients can make meaningful decisions throughout the process - provide choice; (3) get the clients to sign up to be accountable for tasks necessary to the implementation effort. This will aid in developing a sense of responsibility and ownership by the client.

Training is another tool available to assist Project Managers in assisting clients in maintaining a proper level of control over projects. Proper training of the client base is key in maintaining control during systems implementation. Knowledge is the key to acceptability. If people understand a system and how it will personally affect them, they will be much more open to the idea of implementation. This type of attitude assists the Project Manager in gaining, as well as maintaining, control of the systems development cycle.

Along with the opportunities for success which proper utilization of controls can bring to a project, controls can also contribute to project failure. Too much control can result in diminishing returns, where an organization spends more on controls than it can derive as benefits from control. Too much control can also have a tendency to decrease productivity. Lack of control can result in a system which is out of control. Project Managers must constantly make decisions regarding the trade-offs between control and productivity. Management wants increased productivity and does not often stress control. It is the responsibility of the Project Manager to insure decisions relating to control and productivity are based on the proper criteria. This can be accomplished through: (1) training systems development staff as well as clients on the need for proper controls; (2) going through a risk assessment process for each project to help determine the amount of control which may be required; (3) using internal audit procedures to insure the proper level of controls are in place.

These steps should help top management, Project Managers and the client gain a sense of control over the system implementation process. With top management and the client buying into the implementation of systems, a much higher level of system acceptance and therefore overall project success is bound to be achieved.

Communication

Project Managers are really change agents. Top management and clients are often resistant to change and thus need some force to unfreeze them from the current state of affairs and move them to a new equilibrium. Applying the skills surrounding effective communications can go a very long way in getting top management and clients to understand the need for change and progress. Communication must be a conscious part of initial system design. It can not be an afterthought if it is going to be effective.

Through effective communication, Project Managers can integrate a strategic knowledge of information into the goals of an organization. Effective communication to top management can be achieved by (1) making sure departmental goals are in synch with overall company goals; (2) convincing top management that information technology can improve the company's competitive position; (3) discussing cost savings which can result from I/S solutions; (4) talking in terms management can understand and relate to; and finally (5) by showing that future growth depends on implementing the right information technology now. These steps will help bridge the management communication gap and result in successful project implementations.

The behavioral aspects which come into play during systems design and implementation are often complex and overlooked. Project Managers must take personalities into consideration if a successful systems implementation is going to take place. There are many reasons why a wide variety of behavioral aspects can come into play during a project's development life cycle. One reason is the belief that information technology will bring about profound changes within organizations, and will lead to changes within the client community. It is often thought that technology will eliminate jobs resulting in higher unemployment, redefine the basic concept of management, lead to widespread worker resistance and demands for more job security and create controlled social order within the work environment. It is because of this type of thinking that Project Managers are often confronted with many different types of behavior, some of which can be defined as dysfunctional. It is this dysfunctional behavior that can lead to the demise of an otherwise successful project. Project Managers today must be prepared to face this behavior head on and defuse it before it begins having a negative impact on the implementation of systems.

The key to defusing this dysfunctional behavior is to effectively communicate and interact with potential users of a system, gain an understanding of their work environment and deal with any implementation issues. It is the responsibility of the Project Manager to facilitate the change at hand.

Implementation is a process of managing the impacts of systems. Changes in organizational structure, job design, communication patterns, and interorganizational relationships must be anticipated and managed. It is impossible to anticipate all impacts of a new system. Accidental impacts will always occur and the Project Manager must be ready to deal with these. Simply put, failed system implementation is often due to not involving the clients and communicating with them during the planning stages of a project.

Implementation of information based systems can often change the social interactions which take place within work environments. Project Managers must never under estimate the challenge of dealing with a change to a clients interactions with his/her coworkers. Clients perceive a decrease in their ability to make friends when a computer system is involved in their work activities. Hence a properly functioning system may be sabotaged because users feel it interferes with their social network.

There are four basic groups which are involved and affected by information systems development. These include operating personnel, operating management, technical staff and top management. Each of these groups may have a tendency to exhibit different dysfunctional behaviors to new systems development. These reactions can be summed up as aggression, projection, and/or avoidance.

Operating personnel sense that their jobs may change significantly due to new systems development. Because of the level of these people within the organization, avoidance of the system is impossible. This results in aggression and/or projection to be the most common types of behavior displayed. Often times this can lead to all undesirable effects and any other associated problems to be blamed on the computer and its technicians.

Operating management is usually highly affected by new systems. Often new systems result in additional information flowing up to top management about the operating group. In addition, new systems may often result in operational decisions being made by computer-based systems instead of by operating management. This often results in operational management feeling as though their status is being lowered and their power eliminated. Because of this, aggression, projection and avoidance are often displayed at this level of management.

Technical staff rarely display dysfunctional behavior, as development of new systems is their primary duty within organizations. Because of this they sometimes have a tendency to over promote new systems. This sometimes over zealous attitude towards new technology, and what it may be able to do for an organization can fuel the dysfunctional behavior of the client community.

Top management is usually less affected by and concerned with new systems. Insecurity may result because they do not understand the technology being utilized with new systems development. Their high level positions usually allow them to carry out the behavior of avoidance.

There are several guidelines which can be followed which may help eliminate some of these dysfunctional behaviors from occurring. These include: (1) thorough testing of new systems prior to implementation; (2) not rushing personnel into a new system. Make sure you have realistic implementation schedules; (3) keep systems as simple as possible, while meeting the needs of the clients; (4) recognize and adjust for changes in other peoples job content due to new systems; (5) encourage direct interaction between system implementors and clients.

Bringing the information services staff and clients together before design or development begins helps both sides see and understand how their vision of the proposed system fit together. Group design methodologies put structure into the design process, enabling both the good and the bad points to be considered objectively. Such objectivity ensures that the client is asking for the right system and the developers are developing the right system from the beginning.

Good system design must consider the man/machine interface. By doing this, more user acceptable systems will be derived. Astonishing results can be achieved when people from different disciplines communicate to solve problems. This can be especially true when the gap between Information Services and other client departments are bridged. This bridging can be accomplished by: (1) choosing members from both client and I/S groups to work together who are not incompatible. Make sure a system is not hindered or terminated because of a conflict in personalities; (2) make sure that people from the client group and I/S meet as equals. This will allow for a free exchange of ideas and comments without fear of retribution; (3) management from both groups must free their people up so that adequate time can be devoted to the project; (4) responsibility for completion of the project should be shared equally by both groups.

Summary

Successful implementation of systems is continually becoming more important to the overall success of an organization. It is the responsibility of Project Managers to see that systems implementation is being carried out successfully. By sticking to the basics of maintaining control over projects, and communicating with management and clients throughout the entire project life cycle, Project Managers will help insure successful system implementations.

Lack of control, lack of contact, and lack of communication can contribute to poor relations between Information Services and the client community and even poorer systems. Clearly satisfaction leads to usage, while user involvement clearly and directly influences both satisfaction and usage.

SalesSpeak - How to Understand It

By

Jeff Franz

Of

21st Century Systems Group

12475 Woodford Way
St. Louis MO 63044-1368
(314) 298-9659

Language is a software sales person's number one tool. Correct use of words and phrases can mean the difference between success and starvation. For the software buyer, understanding this language can mean the difference between getting what you want and getting what you were sold.

Most sales people will not deliberately mislead a prospect. They know that repeat business is important to their livelihood. Also, since sales depend upon references, it is in the sales person's best interest to be honest.

It is also in the sales person's best interest to make his or her product look as good as possible. It is their job. This brings us back to language. They will use every tool possible to do their job.

The words and phrases commonly used in the software trade may be called Salesspeak. Listed below are some examples of this language. Also included are the real definitions and some possible corrective action.

SalesSpeak - The Language

It is Possible - This phrase is almost as over worked as the word "very." What this means is that under the right set of circumstances, the stated feature will work. What those circumstances are and what else needs to be done is a great unknown. More information is needed and should be obtained. Remember that it is possible to jump from an airplane without a parachute and live. It is possible, but not workable on a routine basis.

We can do it - Of all the phrases associated with SalesSpeak, this is the most common. The operative word is *can*. It does not mean that it does it *now*. Its severity can range from zone 1 to zone 10. Zone 1 states, "the system handles this without modification." Zone 10 is the statement, "if I can convince the president of the company to put a user exit in this place, you could write a subsystem to do it."

In order to determine this statement's position on the continuum, watch the sales person. If you detect nervousness, you are nearing zone 10. You should approach this zone with caution.

Even zone 1 can be a problem. You must determine that the system handles this *in the way you need it*. Just because it does it, does not mean it does it your way. A car can function as a can crusher but not very well.

When this phrase appears, the word "how" should immediately follow from your lips. If it takes response conditioning on your part, it is worth it. Once an explanation is given, become a Missourian, say, "Show me." If this is an important feature, do not buy the software until you see it to your satisfaction.

Let me check on that - This can be a very deceptive phrase if used correctly. It appears that the sales person simply does not know the answer. The truth may be that he knows the answer and it is bad. Why does he not say it? There are two reasons.

The first reason is that the sales person hopes that someone at the office will dream up a way to accomplish the same thing. This can be called the Anne Sullivan ("The Miracle Worker") approach. This is a completely legitimate reason as long as the sales person gets back to you with the correct answer.

The other reason is that he hopes you will forget about this question. It is remarkable that some potential customers will not note this for follow up. The solution to this problem is to note all unresolved issues for later response.

Flexible - Flexible is an all time favorite word among software brochures and sales people. It seems that every thing is flexible. This sounds too good to be true. It is.

This word means that a wide range of interests must be served. It also connotates compromise. The Swiss Army knife is a flexible tool. It has a scissors, knife, can opener and other necessary devices. However, it does not make as good a scissors or can opener as the real things.

The only way to determine whether a "flexible" piece of software will work for your particular application is to contact references. These people must be in the same industry with similar needs. If the software vendor cannot supply the right type of references, begin amending the license agreement. Include necessary functions and features as binding conditions.

There is another problem with flexible software. Flexibility costs money in programming time. It also costs money to make software easy to use. To make it flexible and easy to use increases the cost exponentially.

Look at each of the flexible features to see if they can be used by your organization. You should check out the documentation as well. Be sure that the vendor has not created another source of consulting revenue (see "Complete Consulting Services", below).

User - What is a "user?" To the customer, a user is someone outside of data processing that works with an application. The software vendor considers a user anyone that uses the application. This includes data processing people.

The sales person likes both definitions and will use the one that best suits the situation. Therefore phrases such as "user controlled" and "___ by the user" should be viewed with a careful eye. Even if the sales person is referring to the customer's view of user, it may not be desirable for the user to control it. You should ask many questions.

User Friendly - This is a special subset of the above. Besides the differing definitions of the word "user," all users can be lumped into one category. Therefore, a user friendly piece of software might be easy for an advanced accountant but unusable by a clerk. It also may be user friendly for a programmer with experience in machine level languages.

Normally a demonstration can expose the pretenders. It also may take some digging to be sure. You should be ready to spend some time looking at the functions and having the appropriate people try them.

Available - They might as well say, "extra cost." The word by itself means the same thing. If it is not extra cost, you will by all means know it. As long as you are aware of the meaning, this is normally a harmless word. The only possible follow up questions are, "From whom," and "How much."

Optional (method or feature) - If the sales person wants to attempt to hide costs, he will use the word "optional." Automobile dealers use the word to denote extra cost. Software dealers can use the word to mean extra cost or free. Every time the sales person says this word, ask, "how much?"

It is also wise to find out more details about this optional feature or method. This feature or method may be more a more cumbersome alternative.

Alternative method - If you have received an answer from the sales person and this phrase follows, watch out! It can mean one of three things.

The first meaning is that the sales person has found a better way to do something. He should be questioned on the benefits of his better way. Aspects such as system performance, manual effort and reporting ease should be considered. This is a legitimate answer.

The second meaning is that the system is not efficient as it was first described and the alternative method works easier. It is different from the first case since the initial method is available but it does not work very well. This method should be explored in the same manner as the first.

The third definition means trouble. The system does not handle it the desired way and the sales person is hoping to high heaven that you choose the alternative. If you are trying to catch the sales person in one of these, have him show you both methods. Under the first two meanings, you will see something. If the last meaning is true, you will know it. You should notify the vendor of your dissatisfaction with the sales person.

The next release - When is soon? This is the question to ask yourself when the statement "in the next release" is made. A worse case is the next version. These statements normally appear when you want to see something that is not in the software. It means, "I heard someone at the office talk about putting it in so it must be going in."

Both of these timing elements provide challenges if you cannot see what you want. The next release is normally in process and can have some date associated with it. Depending upon the number of changes to be made in this release, the date may be relatively firm. Ask to see a list of what will be in the next release and the status of each. Be sure your enhancement is on it and that it has a status other than "under consideration" or "under analysis."

If your enhancement is in the next version, be careful. This can be a double edged sword with poison on the handle.

The first problem is timing the next version. You will want it as soon as possible. The software company may have other designs on the programming time it takes to do it. If you force the software vendor via the contract, they may rush the enhancement to you with little or no testing. This can be a disaster. On the other hand, not forcing the vendor may result in the feature being dropped. If the feature is "in the next version", make sure it is not critical to your operation.

In Beta - This is a more definite time frame than the previous one. It also can have its problems. The "Beta" sites might actually be "Alpha" sites. Therefore your feature is longer coming than you thought. Obtain the names of the Beta sites and contact them. Ask them how the testing is going and how the vendor's response has been to problems. This will give you an idea of the feature's readiness.

The software vendor may not want to release the name of the Beta site because of past problems. Ensure the vendor that you are trying to determine the date of general release and will not worry about past problems that have been resolved.

Full featured - It is hard to find a more ambiguous term than this. (Unless you consider "performance," below.) What is full featured? The vendor will want you to think it has everything you could imagine. The package may offer many features. The features may be useful or they may be "bells and whistles." What ever they are, they can be dangerous.

How can full featured software be dangerous? Consider that all software is a series of paths. The software controls these paths via statements (conditions). The more features that a given piece of software has and the more flexible it is, the more paths it contains. As the number of paths increases, the chances of testing all these paths decreases. The decrease in the percentage of testing increases the chance of bugs. Therefore, more features can mean more bugs.

Another item to consider is that one customer's feature may be another customer's bug. If you have purchased software to work one way, it may disappear in a future release. Many vendors will install "switches" to allow a given function to work two ways. Be sure that this is the vendor's policy.

Easily Interfaced - A square peg can be easily interfaced with a round hole given sufficient quantities of sandpaper. There are two things to remember about interfacing. One is the data transfer itself. The sales person will wax poetic on the data acceptance capabilities of his package. This is great, but is all the necessary data available in your present system?

The other item to consider is the master files. Are you sure they can be kept in sync? There must be capabilities to do this. You should ask who else is interfacing their package with yours.

Configurable - The word itself means that there are options on the way the system will handle certain conditions. It also can be a source of extra revenue for the vendor. When the sales person uses this word, ask, "by whom?" Some of these configuration options are permanent and others may be changed.

Permanent options (note the word "option") must be given more thought than the ones that can be changed. Be sure that everyone who will be making these decisions knows which ones are permanent and which may be changed. You should always know who makes the changes.

If you are to make the changes, you should know how often they can be or must be changed. You also should examine the documentation on them. It could be that technically you could make the changes but no one could understand the documentation. Therefore, the vendor makes the money.

Performance (Power) - Do not believe what you hear or read. These are entirely relative terms and have little basis in fact. The only way to compare one software system's performance to another's is to put them on the same machine. Give them the same data to process one after the other. The machine also must be unloaded.

If you cannot do the test described above, there are alternatives. The most financially attractive is through reference contact. Ask companies with similar transaction volumes and machines about timing. It is not as accurate as direct comparison, but it is better than crunching theoretical numbers.

___*-like* - What are "English-like commands?" German and Latin contain some words that sound like English, so are these "English-like?" The same concept is used in software sales. The vendor will claim that his software has commands that use the same context as a commonly used package or utility. This leads you to believe that everyone who uses package "A" will be able to use package "B" immediately. It may not be the case.

The best way to determine who will be able to use the package is to let the target group use it for a while. See how fast they can learn it. If most of them learn it quickly, the rest will follow and you will have a usable package. If few of them learn it, find something else.

We're working on that - Whenever a good idea comes up in a demonstration and it is not in the software, you will hear this phrase. You are supposed to imagine hundreds of programmers busily putting this into the software. It could be that the sales person has just started working on it now.

The author of this paper is working on being a millionaire. He is not doing a great job at it, but he is working on it. This should be your attitude when you hear this phrase. How hard is the software company working on it? What is its state?

Bad Demo Data - When unexpected results occur during the demonstration, this is the sales person's battle cry. The problem with bad demo data is that it may show a lack of control on the vendor's part. Software sales are the most important part of the vendor's business. If they cannot control this, what can they control?

Another problem also exists. The sales person may be covering a bug with this excuse. All software has bugs. The nature of the bugs and the way they are dealt with is important. If you were asking to see something unusual and it blew up, this may be a bug. If the sales person attempts to cover it up, it may be a problem.

We have ___ packages installed world wide - Software companies want to appear as big as possible. They will inflate the numbers with words like packages instead of customers. A package can be a system such as GL or AP or it could be a module such as a report writing package. It would therefore be possible for a software company to claim 500 packages with only 20 customers.

Another trick is to count multiple sites with multiple "packages." This means that a customer with three sites, three packages each (GL, report writing package and a PC module) would count as nine. If you can get down to the number of customers, get a complete list. You should compare the number the sales person quoted to the number on the list. He should reconcile any differences.

We are talking (working) with an organization that - This entity does *exactly* what you are trying to do. This is too convenient. It is much too convenient. The sales person might have met someone in a bar three nights ago that works as a janitor in the same type of organization as yours. The sales person is not lying, he is talking with such an entity.

You can prevent a misunderstanding (misstatement) of the facts through follow up. Ask the sales person for the name of the organization and a contact. If he will not give you one, discount the story as fiction.

Complete Consulting Services - Such a deal, one stop shopping! This sounds better than it may be. The vendor may be depending upon the consulting side for revenue. This can result in many problems.

The first problem is called the "monopoly syndrome," and it is no game. The vendor will attempt to control all consulting and training on its package. If it is successful in obtaining this monopoly, the price of these services is at the vendor's mercy. Another problem is that your project's timing also may be subject to the whim of the vendor. There is a way to discover if the vendor has this problem.

One way to find this out is to ask the sales person if there are alternative sources of training or other help on the package. If he answers yes, then find out who they are. Any other answer may show an attempt by the vendor to monopolize service on the package.

Another way to find other sources is to ask other customers. They will tell you where to look. This information (not the customer name) should then be given to the person in charge of support at the vendor. The reaction can suggest whether the vendor approves of outside consultants or not.

It should be noted that most vendors want to do their own basic training to ensure it is done correctly. Outside help is generally needed for areas above and beyond basic training.

Another problem with vendors who depend upon consulting revenue comes when determining what service charges should be. These vendors will be more inclined to charge customers for small services. The charges are sometimes excessive. It is difficult to determine where free service ends and charging begins.

It is easy to find this type of vendor. Comments from customers like, "they tend to nickel and dime you" or "there were more charges than I thought," will give them away. If you do not get these comments, ask the customer if there are a lot of charges for service. A yes answer suggests caution.

Response time - This is a subset of performance. There is no adequate way to determine response time without using your data on your machine. There is a trick you can use to determine which areas are weakest in a given piece of software.

The trick is to watch the sales person. If he moves the keys a while, pauses, talks and then hits the enter key, watch out. He may have hit the enter key during the previous flurry and is now covering this up. If you suspect this, have him log off and redo the steps. You should watch carefully.

Summary

The sales person will use several of these words and phrases during conversations and demonstrations. They are carefully chosen for both annotative and connotative effect. Listen to each word and phrase that the sales person uses. Be aware of these terms and note the alternative meaning of each. Follow up any suspicions with other customers. In short, do your homework.

Paper 3036

Help Desk Management Relieves the Monday Morning Blues

Kathleen Spanel

GBS Consultants, Inc.
6179 E. Otero Drive
Englewood, CO 80112
(303) 721-0770

It's a Monday morning. After a super weekend, what could possibly go wrong? Apparently plenty! You arrive at your desk only to find your terminal plastered with sticky notes. The phone is ringing, three people are already on hold waiting for none other than you, and one person is standing at your desk!

A quick glance at the notes show that five jobs aborted on the weekend, the tape drive is down, two user terminals are not functioning, and who knows what else awaits you on the phone.

Ah, yes. Back to reality. A typical start to just another week. Perhaps, if you can just hold out till the weekend.

Before picking up the phone, you sit back and reflect. There must be another way. You think...A firefighter I am not, a data processing professional I am! I've got to get control here!

Sound familiar? As much as we may dislike it, the information systems department is, in fact, a service organization. As such, our success depends largely on how well we provide service to our end users or clients.

One area of service, as depicted above, is that of troubleshooting and resolving questions and problems in a timely fashion. How can we gain control of these notes and phone calls and constant interruptions?

The following is an attempt to answer these questions and more and discuss various techniques that may help us turn Mondays into just another day.

Help Desk Management is a means by which we can bring order to the chaos in our organization. You may be thinking help desk management is not possible because you do not have the staff to man a help desk. Or you do not have the budget to hire additional staff to man the help desk. Or, better yet, you do not have the budget to purchase help desk management software. Implementation of a help desk does not have to mean any of these things. Oh, it would be nice if you had access to unlimited staff and funds, but it is not necessary. So, if you were thinking of closing the book on the reading of this paper, think again!

Before we get into the specifics of implementing a help desk, lets look at a few other issues.

Who are our customers?

I previously suggested that we in the information systems department provide service to our end users or clients. More simply and directly, these individuals are our *customers*. Many of us have the misconception that customers are external to the company. But, in reality, as a general rule of thumb, we could say customers are those who are external to our department. Because the information systems department is a service organization, any individual accessing and utilizing information systems, is our customer. How we respond to that individual can and generally does directly impact their performance and success. The more timely and responsive we are affects how timely and responsive they are in accomplishing their goals. Once we accept the idea of who our customers are, we can begin to understand how and why our performance is so critical.

You may be wondering if this means we have to be supportive of all our customers because invariably we have a few that can be somewhat irritating. Unfortunately, as a member of a service organization, we must be accommodating to all customers. By being responsive and providing high quality support to all customers, a win-win relationship begins to emerge making even those irritating customers less so.

Notification

Lets review the various means of notification that our customers use to alert us of a problem. Yes, this is probably fairly basic, but a quick review will aid us in identifying the value of the various means. Sticky notes, as I mentioned previously, plastered over our terminal or desk top, telephone calls, voice mail, and perhaps electronic mail to name a few. For some of us its even worse..we are the unfortunate victims of having to carry beepers!

Lets look at sticky notes. They do serve some value in that they are usually easily accessible and our customers or ourselves can quickly jot down a note. Unfortunately, they tend to get lost or become stuck to something else and disappear forever.

How about telephone calls? Our customers tend to like this means of notification because it is very direct...with a push of a few buttons they are in contact with us. Unfortunately for us, it is a source of constant interruption and tends to disrupt our work flow.

How about voice mail? Voice mail can reduce the interruptions for us but our customers can sometimes feel that their call may never be listened to, let alone answered.

Electronic mail can be similar to voice mail. Though most electronic mail packages have some means of acknowledging to the sender that the message has at least been read.

And, finally beepers. Well, they certainly are direct and to the point. In an emergency situation, they can't be beat. The difficulty is for our customers to distinguish what really constitutes an emergency.

Some of us have managed to stop the telephone interruptions by having our customers fill out a form identifying the question or problem. This can be somewhat effective. But, some users simply won't take the time to fill out a form. We may feel that if the issue is not important enough for them to fill out a form, then it certainly is not worth our effort in trying to resolve! This may eliminate some questions and problems but I'm not sure if it is effective in providing high quality support.

So, what is the answer? We need a means by which

- Our customers can relate to us a question or problem,
- Be assured that the question or problem will not get lost in the shuffle, and
- Minimize the interruptions.

Help Desk Management..Is it the answer?

So, lets get down to it. A help desk. Is that the answer? I like to think that it is. A help desk not only benefits the customer, but it benefits us, as well.

Implementation of a help desk generally results in a *better rapport with the customer*. Now they have someone specific they can call and leave a message with. There is someone on the other end of the phone line that can sympathize with them and their plight and record some basic information. No longer does the customer feel their request will drop into some black hole never to be seen again!

A help desk also lends itself to *establishing a close and hopefully better working relationship with the customer*. The customer will be kept abreast of the ongoing status of their request and ultimately, when the request is resolved, the customer will be involved in the sign-off of the request. Sometimes it is merely the constant conversing back and forth or providing feedback daily on the status of the problem that can lower the anxiety level of a customer.

You may have noticed the mentioning of a sign-off. Why is a sign-off so important? First and foremost, it provides a means for us to check one more time with the customer to insure that they are comfortable with the resolution presented to them. It is important, as well, for us and the customer to realize that sometimes an answer of "no" or "it can't be done" is acceptable. A formal sign-off confirms that the customer is comfortable with this response.

You may think that the sign-off is an unnecessary step because you have been working so closely with the customer during the resolution that it is intuitively obvious that the problem is resolved. That may be so, but it is still a critical last step for it ensures that you and your customer agree on what the problem and the resolution are. There are no underlying assumptions from either party. And, perhaps more importantly, it gives the customer the impression that they have played an active role in the resolution of the problem and that their opinions count.

Finally, because a help desk can lend structure to chaos, its implementation can *aid in getting questions and problems resolved faster, more effectively, and more efficiently*.

Establishing a Help Desk

So, how do we begin establishing a help desk?

Contacting the Help Desk

First things first. How should the customer contact us? Though other means are available, its pretty much accepted that customers should have a *single phone number* they can call. By having a single phone number, any decision-making is removed from the customer. For instance, some shops have one number for hardware, one for software, one for data communications, etc. This scenario requires that the customer know something about the problem and make a decision as to which number to call. By implementing a single phone number, this decision is removed from the customer and placed in the hands of someone perhaps more experienced in making that decision.

This also alleviates any unnecessary interruptions. If the customer thought it was a hardware question when in fact it was a software issue, they may have inadvertently contacted the wrong group which may turn the issue back to the customer for them to try again. Not only did this result in interrupting the wrong group but puts the customer back at square one. Implementing a single phone number assures the customer that once they have contacted the help desk, someone will contact them who hopefully will be the appropriate person and if not, the call will be turned over to the appropriate group without requiring another call from the customer.

Should the person accepting the calls be a technical wizard?

Actually, depending on the call volume, this can be a detriment. If the person accepting the calls is technical, the person typically will have a strong urge to handle the question at that very moment. And what may on the surface appear to be a quick question and resolution, can and often does turn out to be a long, drawn out problem. If the call volume is high, this can then impede other customers from getting their questions submitted.

The person handling the call, the help desk administrator, needs to merely *collect some critical pieces of information*...Customer name, phone number, a brief description of the question or problem, a concise and accurate description of what happened, the action taken, the result, and what should have happened vs. what did happen...to name a few. The administrator may also want to get a reading on how critical this is for the customer..is the customer unable to proceed with their work?...will they miss a critical deadline?...or is it just a question in passing?

Assignment

Once the call has been taken, someone needs to be *assigned to resolve the question or problem*. Who should be assigned particularly if you do not have a staff dedicated to the help desk? One option is to designate one or more members of the staff to be assigned to the help desk for a one week period. These individuals would be partially relieved of their normal activities for the week. During this week they would be dedicated to the help desk. You may question this approach because these individuals may not be fluent in all applications or all platforms in a multi-platform environment. You may counter by assigning calls to individuals who are application specialists.

This is an option, however, it deters from one of the main goals of a help desk..that is to alleviate the constant interruptions. Assigning calls to an application specialist who is not dedicated to the help desk prevents that individual from completing their previously assigned projects in a timely fashion.

Though the individual assigned may not be a specialist on the particular platform or application, there is a certain amount of preliminary investigation and footwork that needs to be completed that he/she can do successfully. By at least doing the preliminary work, the individual assigned can then use the application specialist as a resource only. This approach relieves the specialist from interfacing directly with the customer which can be extremely time-consuming. Additionally, this approach provides cross-training for those individuals who are not specialists in all applications and platforms.

To aid in this approach, it can be helpful to develop a list of questions by application or platform that can be asked of the customer. This can include such questions as what program was being run, on what device, what action was taken, what was the result, etc. In this way, the individual assigned can be sure to have gathered as much information as possible before going to a specialist.

Call Turnover

What happens at the end of the week when it is time for this individual to return to their normal duties and someone else to assist with the help desk? Should the individual retain his "open" calls or turn them over? It is my belief that the calls should be turned over. It is important to remember what it is we are trying to achieve...minimize the interruptions and provide a high quality of support. By retaining calls, the individual will only get more and more involved in the resolution of not only this call but potentially others. The individual assigned may feel that he/she has already been working with the customer for a period of time and the customer feels comfortable with the individual assigned and does not want to have to start over with another person. It is my feeling that this is not entirely true. Customers that are experiencing a problem or have a question have one thing in mind..getting it resolved. And, despite what we may think, they really don't care *who* resolves it but *when* it will be resolved. And, having someone dedicated to the resolution is more desirable than having the same individual working on it in his/her spare time. This is where documentation and a formal call-turnover can be critical to the success of the next individual in resolving the call.

Closing the Call

As I mentioned previously, it is important that both the customer and the individual working the call are in agreement as to when the call is "closed" and as to the resolution. This sign-off must be a verbal or written agreement. How do we get this agreement? Just ask..."Do you feel the call is resolved and closed?".

Automating a Help Desk

Should a help desk be automated? Certainly a manual implementation can be successful. There is a lot to be learned by implementing a help desk manually. We can gain a better understanding of the personality of our customers and the kinds of information we need to record. But, as with any manual system, there are some limitations.

There are some very definite benefits to be gained from the automating of a help desk. Lets review these.

Perhaps the two most significant benefits include the *development and building of a knowledge base* and the *provision of management statistics*.

One of the most difficult things about a help desk is the documentation of how a problem was worked and its resolution. Admittedly, unless any real benefit is derived from documenting the problem and its resolution, individuals assigned to assist the help desk will not be inclined to do the documentation. However, if he/she sees some benefit from documenting his/her work, he/she will be more inclined to continue to do so. The documentation of a problem and resolution provides information to other individuals assigned to the help desk aiding them in more quickly resolving a future problem that may be similar to one previously resolved.

The documentation of all calls and problems also provides management statistics which can be used to our benefit. Statistics such as number of calls taken, number of calls resolved, and average time calls were open can all be derived from the help desk data. This information can be extremely valuable in conveying to other departments what our department is working on, justify existing staff, or possibly justify additional staff.

Automation of a help desk also lends itself to the information systems department being more *proactive in their support*. Now the department can make recommendations as to training requirements, documentation that needs to be rewritten, and applications that need work. It can also provide feedback to the customers documenting what has been accomplished for them during the past week, month, year, etc.

Finally, help desk automation can aid in *providing direction for the information systems department and help drive a steering committee*. By logging all "to-do's" into the help desk data base, it can become clear as to what is being worked on vs. what should be worked on. Other departments can more clearly see what the global picture is and how the requirements of their department fits into the requirements of other departments.

But if we are going to automate, where do we automate the help desk? On the HP3000, a personal computer, a mainframe? Ideally, the help desk information should be accessible and available to everyone. In this way, its automation benefits not only our department but others as well. With its information accessible by all, customers can potentially check on the status of their questions and problems, query the help desk data for potential resolutions, etc. without interrupting the help desk staff. Yet another benefit from help desk automation.

Other areas that can benefit from a Help Desk

Are there any other areas in a company's organization that could benefit from the implementation of a help desk? Actually, any department that experiences questions or problems can benefit from its implementation. Areas such as customer service and the personnel department handling grievances are just a few.

Can they utilize the same Help Desk automation software? More than likely, the answer to this question is yes. These other departments still have the same requirements..collection of data and generation of statistics.

Does it really work?

You may be thinking this all sounds great, but can it really work? Yes, it can but there are some key areas which can either make or break the help desk.

- Must be responsive

  Once a customer has requested assistance from the help desk, it is important that the assigned individual contact the customer in a timely fashion to, if nothing else, confirm that his/her call has been assigned

- Must show genuine concern

  A good portion of the task of the individual assigned is to make the customer feel confident that his/her question or problem is important and worthy of our attention

- Must make all customers feel as though their problem is being worked on and is of utmost importance

  Constant follow-up with the customer is important to assure him/her that their question or problem is being worked on and has not been shelved

- Must not be a first-in/first-out approach

  Though the individual assigned to the help desk will undoubtedly be working on a number of calls, it is critical that some progress be made on all calls

- Must involve a customer sign-off

  Again, a checkpoint that both parties agree that the question or problem is resolved

- Must discipline help desk personnel to document problems and resolutions and stay with the documented/requested priorities

  There can be a tendency to work on less critical questions or problems because they are less difficult

- Must provide feedback

  The maintenance of documentation through the life cycle of a question or problem must provide feedback to the customer and to other individuals working the help desk

Is a help desk the answer for you? Even if you have no funding for additional staff or software? I hope this paper has identified some areas which suggest that it can be just the answer you are looking for and turn those Mondays into just another day of the week!

Paper No. 3037

Attributes of a First-Rate Software Support Application

Mike Butler
Dynamic Information Systems Corporation
5733 Central Avenue
Boulder, Colorado 80301

Introduction

In most Information Systems (IS) departments a large percentage of resources are focused on supporting existing applications, performing help desk services, or staffing information centers. But most IS departments purchase software from various third-party vendors. These products range anywhere from manufacturing or accounting applications to specialized software for networking, database management, or even data center management. Depending on the technical nature of these third party products, the IS support people may have to rely on the support services of the vendor. Each layer of support organizations translates into additional time required to resolve the problem. Depending on the nature of the problem, user productivity could suffer while the problem is being worked on. Support people in both IS departments and in vendor organizations, need to have access to technical information about the products for which they provide services in order to reduce or eliminate the likelihood of users becoming unproductive. The support function is a vital part of an IS organization's and a vendor's overall reputation. For some companies, the support function is a competitive advantage in the marketplace. For instance, Dell Computers is a top contender in the IBM PC clone market because of their award winning support. Having an application with the right features can allow the IS department and vendor organizations to reach their customer service objectives.

The Support Organization

In most vendor organizations, the software support function consists of one or more support center coordinators, many software support engineers, and generally, a manager or supervisor. When you place a call, you are greeted by a support center coordinator whose function is to collect various information about you. This would include your name, the company you represent, your phone number, the product you are having difficulty with, and possibly a brief description of the symptoms of your problem or question. This information is generally logged in a software support application, and a tracking number is assigned for both the customer and the support people. With some vendors, your call may be directed to a software support engineer right away instead of having to wait an hour or two for someone to call you back.

The software support engineer uses the brief description to get an understanding of the general problem the customer is experiencing, and can do some advanced research before taking the call directly, or calling the customer back. Using the support application, documentation and other engineers as a resource, the engineer has a large base of information at his or her fingertips to resolve the problem. A much more technically involved call may require escalation to a more technical R&D resource or supervisor. The support application can be thought of as a knowledge base, where engineers can enter search criteria about the nature of the customer problem and quickly get a potential resolution.

In addition to being an escalation person for technically challenging problems, the support supervisor or manager is responsible for making sure customers are satisfied and given the best possible service. They need measurements and other such reporting mechanisms to ensure this is happening.

In the IS departments, the structure is generally similar, although few are afforded the luxury of a support center coordinator. The individuals providing support could enter the necessary information.

Support Center Coordinator's Application Requirements

The support application must allow the coordinator to be mobile. Being mobile means being able to move between various subsystems in the application easily, and giving access to other applications if required. When a call comes in, the coordinator must be able to respond quickly by perhaps moving to an add call screen.

In most vendor support organizations, the customer purchases a support contract, which can include software and materials support (the customer receives updated software periodically), or phone in consulting support (PICS) which includes the later plus the ability to call the support center at any time. The coordinator needs to be able to determine what type of support contract the customer has purchased for the specific product for which help is requested. This is going to require an interface to the licensee system or subsystem.

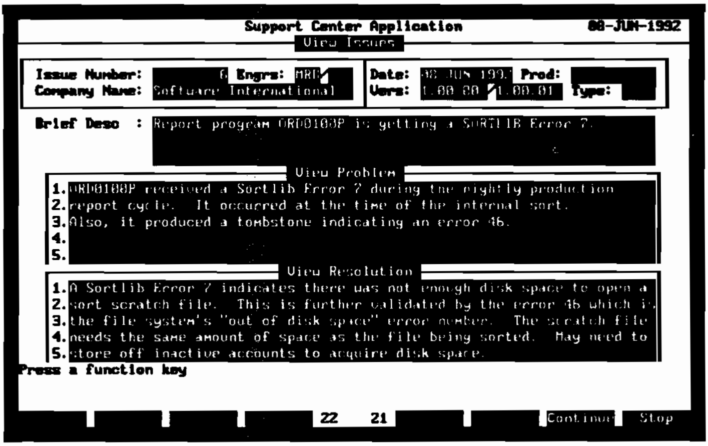
The coordinator must also be able to review the day's calls to ensure that all customers have received a call back within a given time-frame as set up by the support center management team.

In most cases, the customer's time with the coordinator should be limited to 90 seconds. The application can save time when adding a call by taking information from the last time this customer called and using it to fill in the fields on the add screen. The dates and times are defaulted to the current date and time, and the status is defaulted to "open" since this is a new call. The only fields that may change are the phone number, the product the customer is having trouble with, and the brief description.

To illustrate how this would work, the coordinator should be able to retrieve on the customer's name, company, or phone number. A screen would display the customer's open calls along with information about each call, such as responding engineer, call date, and status. Navigational function keys would then be available for the coordinator to move to the required screen. For instance, the customer may be getting back in touch with the engineer on a current open call in which case the application should route the coordinator to a call history screen to make a comment and optionally to the company electronic mail to send a message to the engineer. Similarly, a customer may have a new problem and another new call must be entered, and at the press of the "Add Call" function key, an add screen would appear. Most of the data entry fields would be defaulted to the person's previous call as described above.

Software Support Engineer Application Requirements

The engineer is going to be involved in a multitude of tasks including researching a problem, recreating a customer problem, testing a resolution, or sending a new tape to resolve the problem. To make the decision about what to do, the engineer has to have the ability to search the knowledge base on any word, error number, or error message the customer has experienced. To do this task efficiently, will require a database indexing product. For example, if the customer has experienced a specific error message like "Sortlib Error 7...", on the ORD0100P program, the engineer can do a search like the following: "SORTLIB 7 ORD0100P" on the brief description, problem or resolution text fields. This will retrieve all the records in your database that contain this same error. Since the engineer may be on the phone with the customer during the searching process, the support application needs the ability to instantly retrieve the records, and quickly review problems and associated resolutions. A sample screen that allows the engineer to view problems and resolutions is shown in Figure 1.



```
                         Support Center Application                  08-JUN-1992
                                 View Issues

 Issue Number:            6 Engrs: MRB   Date: 08 JUN 1992 Prod:
 Company Name: Software International    Vers: 1.00.00 1.00.01  Type:

 Brief Desc  : Report program ORD0100P is getting a SORTLIB Error 7.

                              View Problem
   1.ORD0100P received a Sortlib Error 7 during the nightly production
   2.report cycle.  It occurred at the time of the internal sort.
   3.Also, it produced a tombstone indicating an error 46.
   4.
   5.
                              View Resolution
   1.A Sortlib Error 7 indicates there was not enough disk space to open a
   2.sort scratch file.  This is further validated by the error 46 which is
   3.the file system's "out of disk space" error number.  The scratch file
   4.needs the same amount of space as the file being sorted.  May need to
   5.store off inactive accounts to acquire disk space.
Press a function key
```

Figure 1: Sample Screen, View Problems and Resolutions

Sometimes the information being requested isn't so much a problem, as it is just a question about the software that might be best answered by interrogating the manual. The application should allow the engineer to electronically access this information if the text resides in a file(s) on the HP3000. This would be a feature of the indexing product employed above.

As calls come into the support center, the engineer must be able to quickly retrieve them, and take responsibility for them if they are unassigned. The engineer's initials are used to signify ownership of the call. Call history is necessary to track phone calls to and from the customer. An entry in call history includes a call history code, the date and time, and a comment to describe what transpired. Perhaps the customer's phone was busy, there was no answer, or a message was left. When the customer returns a call, the coordinator can make a comment in the call history. The call history is necessary in case a supervisor needs to check the activity on a particular call.

There is generally a lot of communication between engineers and other technical R&D people. The application requires access to the company's electronic mail system. This is generally how the engineers will receive notice that a customer was returning a call if they happen to be on another call at that time. It is also the means for asking questions of R&D and receiving a reply.

The engineer should not have to leave and reenter the application to test and recreate problems. Fortunately for Spectrum machine users, access to the Command Interpreter is as easy as running CI.PUB.SYS. The engineer can do testing from his own account, or if the site has DS or NS/3000, a different account on potentially, a different machine. This is a really important feature since the engineer is involved in so many activities throughout the day. The objective of this is to make sure the engineer can respond to a call easily by exiting the command interpreter and getting to the appropriate place in the application.

A customer may report an enhancement or defect in addition to the call they have open with the engineer. To handle this situation, the application must allow an engineer to open another call to track the defect or enhancement in R&D. When the defect is corrected, or the enhancement is made, an individual in R&D will fill in the resolution and close the call.

A customer needs to know when a defect or enhancement has been completed, since this process can take several days or even weeks. The application needs to be able to manage mailing list information and be able to generate letters that include the problem and resolution information. This is also the means by which mailing lists for newsletters would be generated.

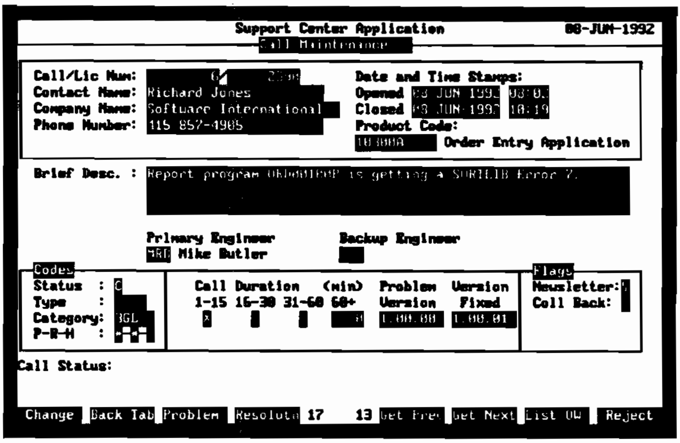
Reporting must be available on an ad-hoc basis. Using the same indexing product as mentioned above, a selection screen can be constructed that gives access to various fields in addition to search items. It is possible for the engineer to request a report of his open calls for the day or for the week. The engineer can also report all the calls with a particular customer problem, so that it becomes easier to piece together a resolution.

The application must allow the engineer to bring closure to a call. This entails:

- ❑ Entering a freeform textual description of the problem and resolution
- ❑ Changing the status from open to closed, entering the amount of time it took to resolve the problem (in 15 minute increments)
- ❑ Flagging the call as either a defect or an enhancement
- ❑ Flagging the call to be included in a technical newsletter

- ❑ Entering the version of software the problem was detected and the version the problem was fixed.

Figure 2 shows a sample screen that most of this information would fall into.



```
                     Support Center Application                 08-JUN-1992
                           Call Maintenance

 Call/Lic Num:           6/     23                Date and Time Stamps:
 Contact Name:  Richard Jones                     Opened  03 JUN 1992  08:0
 Company Name:  Software International            Closed  03 JUN 1992  18:19
 Phone Number:  115 857-4985                      Product Code:
                                                  10 000A   Order Entry Application

 Brief Desc. :  Report program [illegible] is getting a SURILIB Error 7.

                Primary Engineer            Backup Engineer
                MRB Mike Butler

Codes                                                                 Flags
 Status   : C      Call Duration   (min)   Problem   Version     Newsletter:
 Type     :        1-15 16-30 31-60 60+    Version   Fixed       Call Back:
 Category: BGL      X                      1.00.00   1.00.01
 P-R-H    :

Call Status:
```

Figure 2: Sample Screen, Closing a Call.

Support Supervisor/Manager Application Requirements

The application needs to be able to generate metrics and reports to ensure the support center is meeting expectations. In fact, in his book Managing Software Projects, Tom DeMarco said "You can't manage what you can't measure" when talking about software development projects. The same is true of the support center.

In terms of reporting, it is necessary to ensure that accurate information is getting out to the customers. A daily report showing each call along with problems and resolutions, allows the supervisor to review the calls for accuracy, and take action if inaccuracies exist. This serves two purposes: 1), it allows the support engineer to get back in touch with the customer to correct any misunderstandings, and 2), it ensures that accurate information is available the next time someone has the same problem. Another important reporting feature required is a report that gives the supervisor a list of calls that are over a certain number of days old (this number is based on the organization's guidelines). This report is helpful to make sure follow-up is being done on the calls and that none are being forgotten. This is where the call history information might come in handy to review activity against the calls on this list. On a regular basis, the R&D product managers need to receive a list of open defects and enhancements related to their products. This helps with scheduling releases and determines what will be included.

In addition to reports, there are important metrics that help measure the performance of the support center. The support application needs to be able to track information used to formulate metrics. The information needs to be downloaded to the PC and be able to graph the results. Most spreadsheet packages can handle the graphics which will be discussed below. Some metrics make use of 3-D graphics tools.

Weekly and Monthly Call Volume

This represents the volume of calls taken through the support center. It can be the gauge used for supporting staffing decisions. It helps spot trends to determine if the rise or decline in call volume is seasonal, or just the mere fact that perhaps more products are being supported. If you have more than one year of information, you can look at the same week or month last year for comparisons. The weekly information is good to determine staffing levels for holiday weeks. Using last year's numbers as a guide, the supervisor can produce accurate estimates of the call volume for the current year. Figure 3 shows a sample graph on a monthly basis.

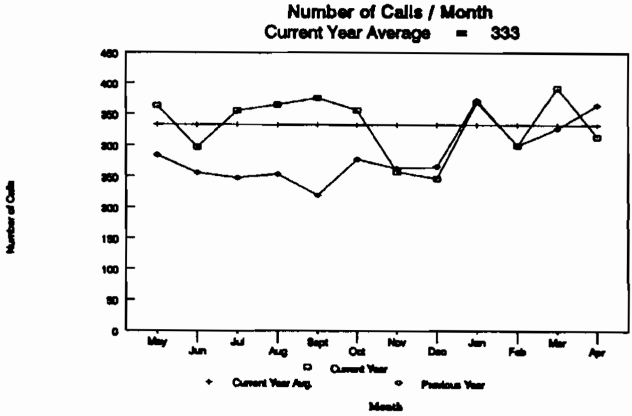


Figure 3: Simple Graph on a Monthly Basis

Monthly Call Responsiveness

When a customer places a new support call, some support organizations will collect the information from the customer, and then try to route the call to an available engineer. If an engineer is not available, a call back is required. If this is a policy, then the application must be able to support it. On calls that could not be routed to an engineer right away and required a call back, the application can flag the record so that collecting the data and calculating the percentages is easy. You should establish an objective of perhaps 70 percent of the calls being routed to an engineer without requiring a call back to the customers. This percentage should be established with the support team's participation. Figure 4 shows a sample graph.

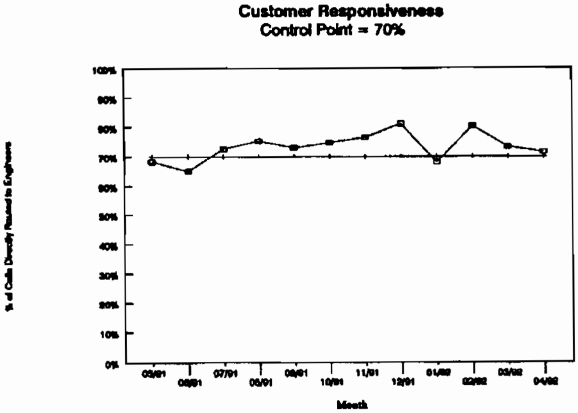


Figure 4: Simple Graph Monthly Call Responsiveness

Monthly Call Distribution

On a regular basis most support teams will convene for an hour to discuss the latest support calls and problems they are having difficulties with. Sometimes, these meetings are scheduled inadvertently during peak call times, and depending on the season, the peak call times can change dramatically. For instance, in the fall and winter months, calls may arrive on a fairly even distribution throughout the day. But it is possible to see calls arriving on a binomial distribution where peak times might be at 10am and 2pm. But, in the spring and summer months, calls might peak in the morning hours, and drop off towards mid to late afternoon. Scheduling meetings and time off the phones during the low demand hours can help engineers be available when customers need them the most. This graph is one that would use the 3-D graphics add-in capability. The X axis contains the call times in hour increments, the Y axis contains the number of calls, and the Z axis contains the days of the week. A sample is shown in Figure 5.

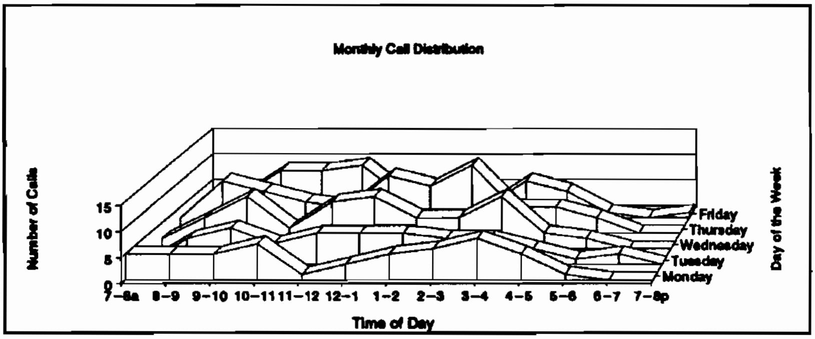


Figure 5: Sample Graph, Monthly Call Distribution

Calls Open 5 Days

This is really a graphical representation of the report discussed above. It shows over a period of time, how the support center is managing its backlog of open calls. The control point is determined by the supervisor on how tightly this should be managed. Figure 6 shows a sample of this graph. The numbers are taken from the report each day.

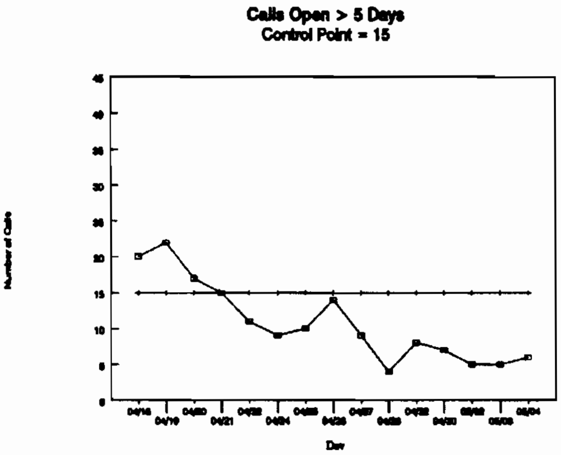


Figure 6: Sample Graph, Calls Open > 5 Days

Monthly Call Duration

This measurement tracks how much time is being spent handling support calls. It's one of those measurements where management has a gut feeling about how long the calls are taking, but needs some data to back it up. This is just one of the tools used to investigate if calls are starting to take more time, and if they are, it allows the supervisor to ask some questions like "Are we spending the right amount of time training our people to handle the new products being introduced?". Figure 7 shows a sample of call durations. Note that they are shown in 15 minute increments up to 60 minutes, and anything over this is lumped into one category.

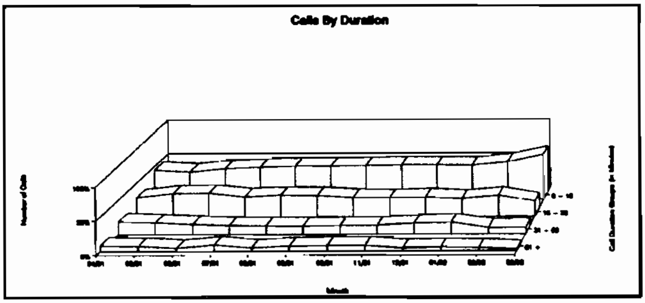


Figure 7: Sample Graph, Monthly Call Duration

Support Application Design Decisions

The application by no means has to start out large. It can start out small with some basic functionality, and grow in future development phases. The sample diagram shown in Figure 8 is a basic database representation to get the designer started. It consists of a master and three detail datasets and would enable the support person to effectively track support calls. Other features like letter writing, and licensee checking, could be added in future development phases.

The application could be coded in any language including COBOL or even C. But, if time is of the essence, you should consider using a fourth generation language. This will save valuable time, and will advance the project to the prototype phase much quicker.

The application should not require the user to go to various functions by traversing long menu paths. The menu paths should be as short as possible. Also, when a user has retrieved a call, and is filling in the information, he might want to fill in a problem, resolution, or call history. The application should provide navigational function keys to "jump" to these entry screens instead of having to go back to the menu and go to the problem via another path.

When designing for the IS department where the equivalent of a support center coordinator position is not available, be sure to minimize the amount of input the engineer must do. If the engineer finds that there is not enough time to enter calls, then it won't get done, and consequently, this will limit the amount of good information available to other team members.

Summary

Every year IS organizations and vendor organizations alike spend thousands of dollars performing support functions. By combining the use of a 4GL and a database indexing tool, an application can be produced that supports the work a support engineer needs to get done. The ideas described above, can put the organization well on the way to meeting the quality and support expectations customers have.

A list of tools recommended in the above discussion is shown in Figure 9. The key to making it work, is to get lots of input from the people who will use the software, and make sure it supports the day to day activities of these people.

- □ A 4GL for developing the application
- □ A database indexing tool for multiple key access
- □ A data retrieval/download tool to place data into a spreadsheet
- □ A spreadsheet add-in product that supports 3-D graphs

Figure 9: List of Recommended Tools

TELECOMMUTING - THE FUTURE HAS ARRIVED

by
Pamela Herbert
A.H. Custom Software, Inc.
El Cerrito, CA.
(510)-525-5070

ABSTRACT

Wave of the future or present day necessity? Telecommuting as a legitimate modus operandi for many American workers has arrived but is sorely under utilized. In this paper I will describe the current state of affairs concerning telecommuting and explore some of the key aspects of telecommuting from the perspectives of the manager, the employer and the employee.

INTRODUCTION

My first telecommuting experience was 8 years ago when the company I worked for allowed me to try working from home after the birth of my first child. Prior to beginning this arrangement the following memo from the MIS director was distributed to the entire MIS staff:

> "We've all probably read articles recently which predict a time when employees will be able to do at least part of their work from a terminal at home. Pamela's planned maternity leave and recent advancements in our telecommunications capabilities have given us an opportunity to learn about the advantages and disadvantages of allowing an employee to work primarily at home with only occasional visits to the office. The arrangements with Pamela are for a limited time and with the understanding that this is experimental, and there are no plans to repeat it."

Note the word 'terminal' in the memo. I left on maternity leave armed with a TeleVideo (dumb) terminal and a 1200 baud modem with a rubber coupling device on it used to plug in the hand set of the phone. I worked out of my living room with the baby sleeping or playing nearby or in my arms, depending on what the situation called for. I always felt that the experiment was a success as I found the baby considerably less distracting than my co-workers. None the less, the experiment lasted about 3 months and then was terminated because a new project came up that involved a lot of on-site training and much team development effort. When I was pregnant with my second child I was informed that the experiment would not be repeated because it would involve too much effort by my manager to monitor my productivity. I never returned to work at that office.

I have now been a telecommuter for the last 5 years, working as a software consultant for a consulting company. The desk in my office sports an AT clone with an in-board 2400 baud modem, a laser printer, a FAX machine, a 2 line telephone and an answering machine. I have 3 phone lines running in to my office, one for business, one for personal use and one for the computer and fax machine. I also have a full-time nanny who watches my 3 children while I work. Incidentally, I was back at work full time 3 weeks after the third child was born. Clearly, something changed between the first telecommuting experiment and my current situation. In this paper I will discuss the evolution of telecommuting and the influences that both hinder and promote its advancement as a standard throughout corporate America.

TELECOMMUTING TODAY

Throughout the literature the estimates of the number of people telecommuting all seem to point to the same study done by LINK Resources Corp. in 1987. This study found that about 26 million people do some amount of work from home including pulling papers out of a briefcase at night to read over in preparation for a meeting the next day. This number drops dramatically to about 6.7 million people who work at home full time with just under 3 million working via dial up access to a computer. Another paper citing the same study states that 3.4 million people have formal arrangements with their employers to work at home. It would appear, then, that most of the people who have formal work at home arrangements use computers to do their work.

The term telecommuting is considered to have been coined in the early 1970s by a man named Jack Nilles who became frustrated with sitting in southern California gridlocked traffic and decided that there had to be a better way. True as this may be, I believe that the real impetus for telecommuting progressing beyond the fantasy stage has been the development of technologies that make it easy to do and that have changed the nature of what we do and how we do it. This wave of new technologies has allowed people to find ways to eliminate the peripheral and non-productive aspects of their work day - donning appropriate office clothing, getting to work, getting a room full of people on the same wave length, getting co-workers to leave him/her alone, etc. and to get on with the task at hand.

In his book, The Third Wave, futurist Alvin Toffler discusses telecommuting at a much higher level. He uses the term the 'electronic cottage' to signify a return to a home based culture where work is more fully integrated with family life.

"The electronic cottage raises once more on a mass scale the possibility of husbands and wives, and perhaps even children, working together as a unit. And when campaigners for family life discover the possibilities inherent in the transfer of work to the home we may well see a rising demand for political measures to speed up the process - tax incentives, for example, and new conceptions of workers' rights". He was somewhat right in that the Republican Party included telecommuting as part of its 1984 platform to encourage the ability of mothers to work at home, thereby preserving the standard American model of the perfect family. This overlooks the absurdity inherent in the idea that a person can simultaneously care for children and be a productive worker, but more on that later. In any case, perhaps for a family run business Mr. Toffler's utopian 'electronic cottage' is feasible but for those who work for corporate America telecommuting will probably be a solo act, or perhaps multiple solo acts, one for each adult member of a household. By telecommuting at least part time, a person can spend more time actually doing productive work while enjoying increased hours of family/leisure time. In other words, telecommuting can vastly improve a person's quality of life.

BARRIERS TO TELECOMMUTING

There are certain barriers and misconceptions that need to be broken down before telecommuting can flourish as a generally accepted method for working. On the side of upper management, the notion that people won't perform unless watched over is frequently cited as a roadblock to sanctioning a corporate telecommuting policy. There is also a sense of loss of control over workers when they can't be monitored and accounted for at any and every moment of the work day. After all, how can you justify paying someone for a 40 hour work week when you don't know if s/he is working or out on the golf course? Lastly, there is a sort of cultural barrier to telecommuting that arises out of the sense that any one who isn't willing to suit up and march in to battle from 8 to 5 every Monday through Friday just isn't much of a person let alone a corporate asset.

Oddly enough, this same fear of loss of control can be instrumental in encouraging corporations to formalize work at home arrangements when they discover that people are doing it anyhow. There is some sense that if guidelines and procedures are implemented and followed, there will be a greater degree of success in allowing people to work from home. There don't seem to be any statistics on the benefits of 'fly by night' telecommuting arrangements verses structured ones but it seems fair to assume that both manager and employee would appreciate and benefit from guidelines. The existence of a formal policy probably makes telecommuting available to a larger percentage of the work force as well since it won't be utilized exclusively by those brave individuals who are willing to coerce the boss in to letting them be absent from the office for a substantial part of the 40 hour work week.

THE REALITY OF TELECOMMUTING

Fortunately, all of the fears noted above have been proven wrong over and over again. In fact, several studies indicate that existing telecommuting programs have generated an estimated 20% increase in worker productivity. Having control over a quiet and peaceful work environment is often cited as a reason for increased productivity. John Scully of Apple is quoted as saying "After nine weeks away, I was ready to ban business meetings. I think I got far more done in Maine with my link to Apple through electronic mail, facsimile, and Federal Express than I would have sitting in meetings. If we could ban meetings as a form of management, American productivity would probably go up"[8]. Extreme though his view may be, when you consider how much time is spent having meetings to figure out what the next meeting should be about you can't help but see his point. Another Apple employee, Blake White finds that the lack of general distractions at the office increase his productivity. "Here at home there are no distractions. I did a revision to a business plan in one day at home. In the office, with all the interruptions, the very same thing may have taken three days or more to accomplish" [4].

Escape from the stress of commuting is another often noted reason for increased productivity from the home environment. Fighting traffic (or surrendering to it) drains the psyche and saps energy. Those with cellular phones can be productive during commute times if making phone calls is an important function in his or her job but most people spend commuting time developing an ill humor. All of the energy wasted on staying calm, watching the road, inhaling exhaust fumes, and switching gears when the commuter finally reaches the office can be channeled directly in to doing productive work by the telecommuter.

Needless expenditure of energy in the form of gasoline is a major force in the trend toward telecommuting. In California there are ordinances mandating that employers get workers to make fewer car trips to work. Saving the environment is the most concrete reason corporations use in justifying telecommuting and certainly a very compelling one.

Most telecommuters go in to the office at one time or another, many at least one day a week. The quality of time spent in the office is greatly enhanced by its brevity. Given only a short period of time in which to interact with co-workers inspires people to make the most of it by getting down to business. Meetings become much more focused with an emphasis being placed on reaching objectives now, not at the next meeting or the next. Interacting with colleagues takes on a new vitality and becomes more synergistic because it is rare and therefore, somewhat precious.

MANAGING TELECOMMUTERS

Not everyone is suited to telecommuting. One of the most important characteristics to consider is how well a person values his or her own work. I had one colleague who was unsuccessful as a telecommuter because of an inability to determine the value of the time he spent on work. This individual felt okay about however much time it took to get something done at the office, but left to himself he often felt that he was taking too long and therefore ended up working very long hours in attempt to give the company what he perceived to be good value for their dollar. He made the intelligent decision and went back to work in the office. In evaluating an employee for telecommuting try considering how well the individual estimates completion times on projects or how well s/he meets your time estimates.

The obvious thing to consider is how well a person functions independently, without constant reminders to turn work in or to make deadlines. A conscientious person who consistently gets to work on time and who doesn't take two hour lunches is clearly a better candidate for working at home than a person who falls down in these areas.

The question remains as to how to manage projects and employees from a distance. With a person's physical presence no longer a yardstick on his or her contribution to the work environment how does a manager know that the employee is earning her salary? Clearly, all of the criteria used to evaluate an employee's performance aside from hours spent at work become much more important. This means that management must be by objective where goals and the time estimates to reach those goals are very clearly defined.

Managers must also be very clear on what projects or jobs lend themselves to working from home. In software development, the early phases of a project generally involve a tremendous amount of team work requiring all parties be on-site much of the time. Once the project is in the programming stage, much of the work is handled by individuals responsible for particular programs or modules. In the system testing phase it may be necessary, once again to bring workers together often. Both the manager and the project member need to be flexible in determining how much time is spent in or out of the office.

Good communication is a cornerstone of successful management under any circumstances but becomes particularly critical when the communications are less frequent. Communication methods are different with telecommuters. The phone (often the answering machine) and e-mail replace stopping by a person's desk several times a day. Communicating goals, objectives and expectations fully and clearly from the start can save a tremendous amount of time lost due to misunderstanding or simply spent in the act of communicating.

In short, the methods that make telecommuting successful are excellent management techniques and probably account for a measurable percentage of the productivity increases seen in telecommuting programs.

BEING A TELECOMMUTER

Peace and quiet, yea, the elimination of all inter-office distractions is the most frequently cited benefit of telecommuting by those who do. Just being able to sit down and do a project from start to stop without having to deal with meetings, co-workers stopping by to chat, managers needing information, users requesting the world, etc., is the soap box whence all telecommuters sing the praises of their working conditions. Exchanging inter-office hubbub for children running around needing constant attention or another adult who constantly interrupts will sabotage the efforts of even the most conscientious worker. The working environment must either be an empty house or a separate room with a door that can be closed, in order for the peace and quiet benefit of telecommuting to be enjoyed.

Proper equipment is a necessity. Who provides and maintains work at home equipment is generally worked out as part of each telecommuting program. Regardless of where the equipment comes from, it must be set up so that the telecommuter is comfortable using it. Ergonomics isn't just a buzz word, it is the very thing that prevents stiff necks, back aches and repetitive motion injuries.

Working at home offers total flexibility in terms of what hours are worked - sort of. Telecommuters still need to be accessible to the rest of the 8 to 5 world during at least some portion of those hours. They also need to be conscientious about tracking how many hours are spent working to keep from feeling either over worked or guilty because not enough hours have been put in. One of the biggest traps telecommuters fall in to is working most of the day and then some more at night making 10 to 15 hour days. Learn to turn off the light and close the office door at an appropriate time.

BENEFITS TO THE EMPLOYER

Several benefits to the employer have already been alluded to. Providing a means for reducing traffic congestion, air pollution and energy consumption is one. Improved management styles required for successful telcommuting can lead to greater productivity. Companies are able to draw on a larger pool of skilled employees because geographical proximity to the office becomes less of an issue. Keeping key employees when relocating a business is another. Pacific Bell was able to keep certain key people when it moved 2000 workers from San Francisco to a suburb 40 miles away by allowing them to telecommute for part of the work week. As telecommuting becomes more accepted and more common, corporations will be able to maintain smaller offices thereby saving money. The logistics of what kind of space to provide people who use it only occasionally have yet to be worked out, but surely the space requirements are lower.

DISADVANTAGES OF TELECOMMUTING

As Robert Heinlein once wrote "there is no such thing as a free lunch" and telecommuting is no exception. There is a certain extra effort that needs to be made to blend an office where some employees telecommute and others don't. Ad Hoc meetings when the telecommuter is out can leave that person excluded from making important decisions. It is up to a manager to see that it doesn't happen often which contributes to good planning but detracts from spontaneity. Giving one of several people the freedom to come and go at will can also generate inter-office jealousies. Lastly, loneliness and isolation can be a problem for a person who is geographically distant from an office and can't choose to be around people more when needed.

SUMMARY

The revolution in communications technology has brought new ways of working to the world. The results of employing these new technologies are overwhelmingly positive - decreased traffic, air pollution, and energy consumption, higher worker morale, improved management styles, better quality of life, larger pool of skilled employees, and savings on office space chief amongst them. My own, personal experience with telecommuting has been very positive and I can't imagine ever going back to the old 8 to 5 at the office routine. The trend toward telecommuting is clear and if your company doesn't have a telcommuting program yet, chances are it will soon.

REFERENCES

1. Dordick, Herbert S. and Williams, Frederick
   Innovative Management Using Telecommunications

2. Farmanfarmian, Roxanne
   "A Manager's Guide to Making Telecommuting Succeed"
   Working Woman; Feb. 1989

3. Flemming, David
   "A Design for Telecommuting"
   Personal Computing; Oct. 1988

4. Gite, Lloyd
   "The Home Based Executive"
   Black Enterprise; Jan. 1991

5. Gross, Neil and Verity, John W.
   "The Portable Executive"
   Business Week; Oct. 10, 1988

6. Katz, Donald R.
   "Hard Facts about the Home Office"
   Esquire; Apr. 1990

7. Kelly, Marcia
   "The Work-at-Home Revolution"
   The Futurist; Nov/Dec 1988

8. Toffler, Alvin
   The Third Wave New York, Bantam Books 1981

PAPER NO.: 3039

TITLE: RF/DC and the HP 3000

AUTHOR: Joe Howell
Professional Products
P.O. Box 589
De Funiak Springs, FL 32433
(904) 892-3021 ext 306

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

Paper # 3040

Risk Taking... In Your Career and Personal Life

Kathy S. McKittrick
McKittrick Associates, Inc.
5547 South Yampa Street
Aurora, Colorado 80015
(303) 690-1550

Risk taking. We've all done it. Sometimes we've won... sometimes we've lost. But we've almost always learned from it, and its more a part of our lives than we might think.

Risk is defined as "the possibility of harm or loss", but the term is more often used to describe actions that result in unpredictable consequences; positive or negative. And that's what we're talking about in the context of this paper.

For example, when we change jobs we take a risk. But before taking that risk, we calculate that there is a high probability that our new job will be more satisfying than our current job. This doesn't necessarily mean that the new job will meet our expectations. It simply means that we enter into it believing that the odds are in our favor, based on the data available.

How do we learn to evaluate risk? When we begin life, the world is full of risks. We can't avoid them, so we move forward with innocent gusto, forging ahead even after the bumps and bruises that come with new discoveries.

Consider the toddler who wanders away from his mother in a department store. All those bright colors, noises and possibilities draw him away before his small brain becomes aware of the possible consequences; losing his mother. When he realizes that she's nowhere to be seen, he panics, and the experience becomes a very unpleasant one. Even so, he's gained something; the knowledge that a world beyond his mother exists, the exhilaration of freedom and the first seeds of a self-confidence that will someday allow him to leave home.

As we continue to grow, the risks we take grow and change. By the time we reach our teen years, we begin to grow cautious in certain areas. We avoid the risk of feeling foolish in front of our peers, and yet at the same time, we risk the wrath of our parents almost daily (or at least every weekend) by indulging in behaviors that we know they won't approve of.

As young adults, we are bombarded with risky situations. In some we are at emotional risk. The teenage boy who calls a girl to ask for a date is at severe emotional risk, as evidenced by the knot in his stomach and the cracking of his voice as he asks. But he calculates his risk, ultimately deciding that the possible benefits outweigh the possible rejection. If he chooses never to take such a risk, he might end up alone.

When we go through the process of interviewing for our first job, we are called upon to behave in ways which are foreign and risky to us. Dressed in clothes that we wouldn't have been caught dead in on a college campus, we smile and shake hands and second guess what it might be that the interviewer wants to hear. The reward? A job. The alternative? Unemployment.

We've all heard the story about the aspiring actor who risks it all by barging into the producers office and demanding an audition. Or the one about the young athlete who crashes a major league practice to show his stuff. These young men have nothing to lose, and everything to gain, and so the risk becomes palatable.

That's part of the reason that we so willingly face risk in our early lives. Without risk there can be no movement or growth. We've got little to lose and everything to gain. Also, during our youth we don't calculate our risks very well, if at all. Our lack of experience allows us to take risks that later we wouldn't consider because we know what most of the variables are.

Somewhere along the line, as we mature, we begin to avoid risk. Risk is often uncomfortable, and the more comfortable our lives become, the less we want to "rock the boat".

There are a number of reasons for this. First and foremost is the "if-its-not-broken-don't-fix-it" mentality. If things are going along smoothly, why take a chance that an alternative will cause hardship?

Another important reason that we risk less as we age is that we become accustomed to the self-satisfying feeling of being experienced and knowledgeable. By the age of thirty or forty, we realize that we've spent much of our energy trying to eliminate risk from our lives. We marry, settle down in a specific job or career, order the same entree when we go to our favorite restaurant, and return to the same vacation spot year after year. Why introduce unknowns into our lives when we're comfortable and secure? Because the potential rewards can be great.

Consider Jimmy Conners. Last August, this thirty-nine year old world champion tennis player took an enormous emotional and professional risk when he played in the U.S. Open Tennis Tournament. He certainly didn't need the money. He could have made that much and more by playing the role of the aging athlete announcing tennis matches and making commercials. Instead, he trained and played all year, and gave it a try. What was the worst that could have happened? A humiliating defeat (such as the one John McInroe suffered) in an early round. This was no small risk for a man who had been at the top of his field.

He didn't win the U.S. Open. But he did make it to the semi-finals, and won the praise and admiration of the public and his peers. You could feel his exhilaration and heightened sense of self-esteem through the T.V. screen.

Another example is a retiring airline pilot who, at the age of sixty, decided to learn to ski. He considered the benefits; exercise, an activity that he could share with his children and grandchildren, and the self-satisfaction associated with accomplishing such a feat. He considered the downside; the possibility of broken bones, the grueling hard work involved in learning to ski, and the ridicule of other skiers watching an older man inching down the mountain. Today this man is seventy years old and has enjoyed ten years of skiing. During that time he's taken ski trips each year with his children and grandchildren, maintained his health, and gone on to enjoy other sports for which he never had time during his career.

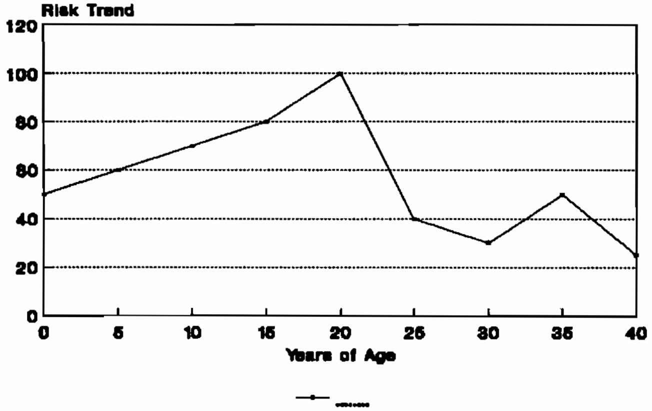
These are two examples of people who have risked and won; both when they were well past what many of us may consider to be our "risk taking years". There are many reasons that they were successful; luck among them. But in addition, they had two things in common. They believed that they would be successful, and they calculated their risk and decided that the possibility of a positive outcome outweighed the possibility of a negative outcome. They also happened to be people who were more accustomed to taking risks most of us.

We know that there can be benefits to taking a chance. But how do we become more comfortable with doing so?

First, we can look at our history. We've all taken risks at one time or another, and its helpful to look back and review their outcomes. Figure I shows the risk-history of a forty year old woman. Its not an unusual history.

FIGURE I

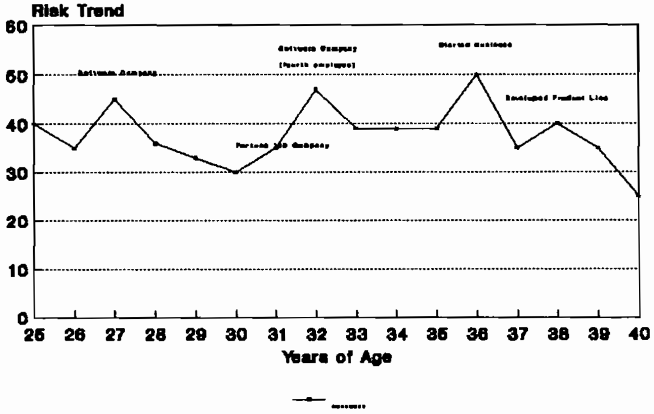
Charting Risk History



Early in life, her risks were controlled by her parents. As she grew older and became more and more responsible for her own actions, her risks increased, peaking in her early twenties. As she continued to age, and presumably realized some success in her life, she became increasingly self-satisfied and reduced the frequency and number of risks that she took.

Figure II, a more detailed chart of her career risks, which is quite different from her overall risk chart, shows that her risk taking increased into her mid-thirties. According to the chart, she's due for her next big risk, but may choose, as so many do, to avoid it.

FIGURE II
Career Risk Chart



The purpose of charting your risk history is so that you can review and examine those risks, and analyze their outcome. How many of your risks had positive outcomes? Do you remember how you felt and what elements caused you to go ahead and take the leap? Can you apply those factors to a current risk that you're considering? Chances are that you'll find that many had positive outcomes and that you feel slightly exhilarated after thinking about them. In addition, you might feel more comfortable and certain about the risk you're considering.

Another way in which you can encourage yourself to take risks is to carefully calculate the possible outcomes. This means more than casually considering what might happen. It means that you must determine the positives and negatives that might result in your taking the risk. For example, if you're considering a change in careers, first, list the factors that caused you to consider the risk in the first place. This might include things like additional income, job satisfaction, peer admiration, personal growth, and personal pride. Then, list the possible negative outcomes. What if you wind up making less money instead of more? How little is too little? What if you wind up feeling a lack of confidence because you are unfamiliar with your new role? How will you deal with that?

Listing the positives and negatives does two things for you. It allows you to evaluate whether or not you can deal with the worst possible outcomes, but more importantly, it allows you to anticipate some of them, and put strategies in place for dealing with them ahead of time. Finally, it allows you to measure the reward versus the risk, and thereby decide whether or not its worth taking the step.

Another important aspect becoming comfortable with risk, is practicing whatever it is your considering doing. Let me give you an example. About ten years into his career, a successful Systems Engineer considered a career switch to software sales. He had worked with customers extensively, so he was comfortable with dealing with their problems and concerns, but he wasn't at all comfortable with two aspects of sales; telephone prospecting and asking for an order. When he mentioned his fear to one of the Sales Reps he was working with, the Sales Rep suggested that he do some telephone prospecting before taking the job. The Sales Rep gave him a list of prospective customers in his territory who were the least likely to be interested in the company's software product. After coaching the Systems Engineer, the Sales Rep encouraged him to try any approach he liked that was comfortable for him.

"I don't think I'll sell anything to these companies anyway, so don't worry about losing a sale. And who knows. Maybe you'll interest them."

By taking the pressure of "success" off of the Systems Engineer, the Sales Rep gave him a non-threatening environment in which to practice his skills and determine if this was the right job for him. What was the worst thing that could happen? The Systems Engineer wouldn't lose his job. He couldn't even lose a sale since the Sales Rep had already written the companies off. The key here is that the risk was removed for a short period of time, allowing the Systems Engineer to make a more objective decision.

If you're getting ready to go into the boss and ask for a raise or a corner office with a window, practice first with your spouse or a friend. Be sure to give them a profile of your boss first, so they can simulate his or her responses. Encourage them to make it tough on you. Then you'll be ready for anything.

Before taking the risk you're considering, it could benefit you to seek out and spend some time talking to other people in your life who have taken similar risks. This might be a family member, a friend, or a co-worker. Ask them what steps they went through in their decision making process. Talk to them about why they think they were successful or unsuccessful. It always helps to know that others have walked in your shoes.

Finally, one of the most important aspects of risk taking is your belief in yourself. The competitive diver who allows herself to feel a momentary lack of self-confidence as she takes that last bounce on the diving board will not execute a perfect dive, and might even injure herself. She's got to go out to edge of the board thinking that she's a championship diver; better than any other diver competing. That's how you must approach risk, once you make your decision to take the plunge. Tell yourself that your an intelligent capable person. Remind yourself of your past successes. Before you take the risk, imagine yourself in the position of success. Imagine walking into the boss's office and approaching him or her with your request. Imagine that boss offering objections, and imagine yourself meeting those objections. Finally, imagine the boss agreeing with your point of view. Go through this process several times. You'll find that you'll be much more comfortable when you actually make your move. Imagine a scenario enough times, and it can seem almost as if its happened, giving you the feeling that you already have experience with the risk you're taking.

If things go sour, remind yourself of your past failures, and how you were able to successfully recover from them. You're still around and even though you may have had some setbacks, you've learned and grown through trying new things.

If you still find that you're not ready to take a risk that some part of you wants to, look at the reasons why you're avoiding the risk. There are a number of possibilities. You might have a realistic fear of the possible negative outcomes. If so, you might want to put the thought of the risk aside for awhile and re-examine it after some time has passed.

You might lack the self-confidence required to take the risk. See if you can find some formal training that can help you to feel more confident. Depending on the area you live in, you can often find courses on assertiveness, self-confidence, or, more specifically, courses that will give you the skills you feel you need before you take the risk.

If you're avoiding a risk due to financial or other responsibilities that you have to your family or loved ones, the time might not be right for the risk. Determine what it is that you can do that will put you in a position to meet those responsibilities while you take the risk. Perhaps you'll need to delay for a year or two while you set aside some money for the possible down side to your risk. Maybe you need to wait until your children are older. In any case, don't turn away from the risk, but rather, put together a plan that will allow you to do what you want to do.

If you've decided to forego the risk because the possible negatives outweigh the possible positives, you've probably made a good decision; as long as you've actually gone through the process of evaluating the risk in depth.
If, on the other hand, you're avoiding the risk because you're comfortable with where you are, take a second look. Most of us consider risks because we have a basic belief that we can improve our lot in life, and because we believe that to truly live you must risk. Don't pass up what might be an exhilarating experience, and what at worst will be a learning experience, because you want to avoid the unknown. Whatever your age, you're too young to avoid risks at the expense of growth.

MORE Secrets of MPE V Tables:
A Closer Look at Code Management Structures

Craig Nickerson
United Electric Controls Co.
180 Dexter Ave.
P.O. Box 9143
Watertown, MA 02272-9143
U.S.A.
(617) 926-1000

Introduction

At the 1985 INTEREX Conference in Washington, D.C., Eugene Volokh of VESOFT, Inc. presented an excellent paper entitled "Secrets of System Tables...Revealed!". This was subsequently reprinted in the third edition of his collection of essays, *Thoughts and Discourses on HP3000 Software* (it has been dropped with the fourth edition).

A highly informative and readable overview, "Secrets..." made the MPE internal data structures comprehensible. It was a godsend to responsible systems programmers, and has certainly been of immeasurable help to me in a variety of security-, performance-, and system-management-related applications.

In a recent (1991) HP database survey, two thirds of the respondent shops were either exclusively MPE V (or "Classic" MPE) like ours, or some combination of MPE V and MPE XL. In view that Classic machines are still so widely in productive use, I thought it would be worthwhile to produce a sequel to Eugene's paper, focussing on the internal management of code segments, and the interrelationships among the various tables involved.

In the area of code segments, the *MPE V Tables Manual for MPE V/E*--particularly the first (September 1984) edition to which I have access--presents a picture which is almost, but maddeningly not quite, complete; like a jigsaw puzzle, with a few missing pieces critical to one's full understanding and buried in various HPIUG and INTEREX papers.

I here present what I have gathered into a more-or-less coherent whole. To use this material requires (as one might expect) Privileged Mode (PM) capability, in addition to the Tables manual itself and Eugene's writings on its content--not to mention a fair amount of programming skill, and a high degree of responsibility. The full tale has been told elsewhere about the powers and perils of PM, and how to use it without destroying the integrity of your system.

I have provided four application examples and two diagrams to illustrate how everything fits together, plus two appendices, including notes on the original "Secrets...". Also, I have identified the locations of data items in the tables by such field names as are given in the manual first edition, rather than by their absolute offsets (which are, after all, subject to change without notice)--this to help ensure that use of this material will be competent, being virtually impossible without the Tables manual as a resource.

As you read this paper, keep the following two sentences in the back of your mind:

> **A relational database allows you to link or join a table to any other table that shares a common item. It is not required that this common item be a key.**
> -- Eric Savage, "Performance in Relational Databases" (1990 INTEREX Proceedings, Boston, MA)

The De-Homogenizing of Memory

In the formative years of data processing--back in the days when I was learning to write assembly code on the CDC 3600 and the IBM 1130--a computer program in memory was a single contiguous bunch of bits, either executable as machine instructions, or manipulable as data. Whatever boundary existed between instructions and data was defined by the program itself, and could be changed at any time, in any way, at will. This made programs potentially self-modifying, and assembler languages naturally lent themselves to this capability. Batch-oriented multiprogramming mainframes like the CDC 3300 required as much as 160K words of memory to accommodate a mere handful of concurrent jobs, largely because the impossibility of sharing code that can rewrite itself made it necessary for each job to have its own private copy of whatever program it was running.

With the advent of more sophisticated and powerful, albeit more physically compact, systems, a program's machine instructions and internal data became segregated into different sections of memory; in the HP3000 under Multi-Programming Executive (MPE), these sections are called *segments*. Under this architectural arrangement, all instructions are contained within one or more *code segments*, and the only primary memory the program can fully access is a *data segment* for its working storage, called a *stack*, and optionally one or more additional data segments, while only reading from its currently-executing code segment is allowed for the purpose of retrieving the values of constants embedded in the object code at compile time. This makes it impossible for a program to modify itself; nor can any data in the stack be executed, except for instructions dynamically built and executed one-at-a-time at the "top-of-stack". With the boundary between instructions and data thus fixed by hardware and operating system, program code becomes *sharable* and *re-entrant*, so that:

■ A single copy of the program in memory can be executed by several *processes* contemporaneously, independently of each other and each with its own copy of the stack, since the code itself is impervious to change at run time.

- A procedure can call itself recursively, without jeopardizing its local working storage, and without having to juggle return addresses.

The Classic HP3000's memory segments are mapped in the memory-resident Segment Table (ST), which is in turn subdivided into three contiguous and individually accessible tables, in this order:

Data Segment Table (DST):
For tracking the locations of all segments of data, including the ST subdivisions themselves, where the operating system and its constituent processes, and the constituent processes of user jobs and sessions, keep their "stuff".

Code Segment Table (CST):
For tracking the locations of all segments of instruction code loaded from the system library (SL.PUB.SYS) and other Segmented Libraries (SLs).

Code Segment Table eXtension (CSTX):
For tracking the segment-by-segment locations of loaded programs (except for the Command Interpreter and certain other system programs).

The memory-mapping entry for a data or code segment is four (4) words long and indicates whether the segment is "present" (in main memory) or "absent" (swapped out to *virtual memory* on disc). In the former case, the location is given as a memory BANK and BASE; in the latter case, as an L-type disc address (logical device number (LDEV) in the high-order byte).

Hereinafter, the term "memory" unqualified refers to either main or virtual memory.

The Numbers Game

Code segments, like data segments, have numbers attached to them, and we will approach the topic from that perspective.

Physically, code segments come in two varieties, corresponding to the two varieties of *prepared object code* files in which they are stored on disc, namely SLs and programs.

An SL code segment, being a collection of one or more procedures and not dedicated to any one particular program, is sharable as an individual unit. It has a *logical segment number* within the SL--the number displayed by the Segmenter -LISTSL command and shown on PMAPs; putting it metaphorically, this is its number on the "street" where it "lives". When it is *loaded* (into memory), it acquires a *physical segment number*, the key to its memory-mapping entry in the CST; this is the global identifier that it has "on the job".

Understand clearly that sharability is based on identity rather than content. SL code segments with different "home addresses" are handled as separate and distinct objects, regardless of their sources in User Subprogram Libraries (USLs).

A program code segment, on the other hand, is sharable by processes only as part of the family of code segments that constitute the program; in other words, it is private to the program and not sharable by other programs, and is thus, unlike SL segments, permitted to contain data references bound at preparation time to the global stack region between DB and Qi. A program code segment has a logical number within the program file, but not its own unique physical number "on the job", because all "residents" on the "street" are loaded and unloaded together, and their memory locations are mapped in a contiguous block of entries in the CSTX--one 4-word header followed by up to 255 mapping entries similar in format to those in the CST. Now, with these mapping blocks potentially varying in size from 8 to 1024 words, imagine how limiting this would be of the number of programs that could be loaded at one time, and how much table space would be wasted by single-segment programs, if these blocks were organized into stationary slots of uniform size, and the CSTX remained confined to a single data segment! Obviously, MPE must pack the CSTX to prevent fragmentation and maximize capacity, and as a consequence, a program's mapping block migrates when a previously-loaded program is unloaded. This in turn necessitates an auxiliary structure, the CSTXMAP (or CSTBLK), for tracking the current locations (relative to the base of the entire ST) of all CSTX blocks. A loaded program's unique key to this array, called a *block map index* (CSTXEIX), is its global "on the job" identifier.

Sharability being based on identity, code segment families from different program files are also separate and distinct to the Loader--even if they happen to be identical copies of the same program, prepared from the same *relocatable object code*.

When we shift our perspective to that of the process, we need to change our method of classification. From the sharer's point of view, all code segments break down into two categories:

1. **Physically-mapped segments**, which "live" in the system library, are part of the operating system and are mapped only in the first 255 CST entries (excluding entry 0, the table header), all of which are reserved for MPE.

2. **Logically-mapped segments**, which are non-MPE and further subdivided into:

   a. User SL code segments, including those serving subsystems like the EDITOR and 3GL compilers, which are mapped in CST entries 256 up to the highest configured entry (maximum 2047).

b. Program code segments, loaded from PROG files and mapped in the CSTX.

In the next section, we will learn how the two categories are differentiated, and why segments in the second require a third type of segment number.

The 8-Plus-1 Solution

The lower byte of the status register holds a number indicating the code segment in which the current process is executing. Now, the largest number that can fit in 8 bits is 255; but, as we have seen, the first 255 physical segment numbers are all reserved for MPE, and somehow, these code segments, and the system intrinsics they contain, must be available to a process in addition to its own constituency. Yet, how is it possible with only an 8-bit segment number?

Part of the answer is an internal 9th bit called a *mapping flag*, used simply to distinguish between physically-mapped (MPE) and logically-mapped (non-MPE) code segments. If the currently-executing segment is physically mapped, this bit is on (1), and the segment number in the status register is a physical number (CST index); if the current segment is logically mapped, the mapping flag is off (0)--and this is where the third type of segment number comes in. It is usually documented as a "logical" segment number, but I prefer to call it a *local segment number*, to avoid confusion with the segment's "street address", and to signify that this number may, for all practical purposes, be considered local to the process sharing the code segment.

What this local number is and how it works will become clearer as we go along, but first, I will summarize my definitions of all three segment numbers and establish the abbreviations I will be using from this point on:

Logical segment number (LOG#):
The number, assigned by the Segmenter, uniquely identifying a code segment within a prepared object file, either program or SL; associated therein with a 1- to 15-character ASCII name. Its value ranges from %0 to %376 in a program file, from %0 to %775 in the system library, and from %0 to %375 in a user SL.

Physical segment number (CST#):
The number, assigned by the Loader, uniquely identifying a loaded SL code segment system-wide; also called "CST number" or "CST index". Its value ranges from 1 to 255 for MPE segments, and from 256 to 2047 for user segments.

Local segment number (LCL#):
The number, assigned by the Loader, uniquely identifying a non-MPE code segment, either program or SL, within a process sharing the segment. Its value ranges from 1 to 255.

Thus, with the aid of the mapping flag, a process can share a maximum of 255 MPE code segments, plus a maximum of 255 program and user SL segments.

When a process calls a procedure, the mapping flag for the calling code segment is saved in the stack marker in bit 1 of the return address cell (located at Q-2). This explains why, on trace-back dumps, some return addresses appear to be larger than %40000; in those cases, a procedure was called from an MPE code segment. In fact, code segments are limited in size to 16K words (half the maximum size of data segments) for both machine code and embedded Segment Transfer Table (STT).

When a program is shared, LCL#s are assigned first to the program code segments beginning with 1, so that for any given program segment,

LCL# = LOG# + 1

The LOG#s of program segments are always consecutive from 0, with no "holes", since they are prepared (:PREPped) in the same way that they are loaded and shared--as a group. Thus, a LCL# which is less than or equal to the number of code segments in the program always points to a program segment--specifically, the one whose LOG# ("at home") is less by 1--and is the number used by all processes sharing the segment (by virtue of sharing the program). (It would appear that the "CST REMAPPING ARRAY" in the program file is merely a vestige of earlier versions of MPE and no longer serves any useful purpose.)

A process sharing the program is linked to its code segment family in memory by the afore-mentioned CSTXEIX, held in the Process Control Block (PCB) of the process (field PBX). The LCL# of a program segment also serves as the key to its mapping entry from the base of the program's CSTX block.

Things get more interesting when user SL code segments are shared, either through load-time binding, or dynamically through the LOADPROC intrinsic. To point to these logically-mapped segments, LCL#s are assigned beginning with <no. of program segments> + 1, and to connect these to their 11-bit CST#s, the Loader creates for the process a Logical Segment Transform Table (LSTT), in a data segment whose DST index is held in the PCB (field MAPDST). At the beginning of this table is an array of 2-word entries, keyed by LCL# and covering all code segments shared by the process. These are: one overhead entry (entry 0); dummy entries for the program segments; and entries for the user SL segments. Each of these latter contains, for the corresponding code segment, the CST# and, if needed for binding to other user SL segments (for reasons I'll be discussing just ahead), the location, in the LSTT, of its External Label List (ELL).

To further clarify the concept of sharability: "All code segments are sharable, but some are more sharable than others." What I'm referring to is the fact that SL code segments are sharable among different programs, as well as among different processes. What does this mean in terms of the LCL#s assigned to those that are non-MPE? It means that, while all processes running a given program sharing a given user SL code segment refer to said SL segment by the same LCL#, processes running a different program sharing the same SL segment may refer to it by a same or different LCL#, depending in part on whether the number of program segments is the same or different. Furthermore, a process running yet a third program that is not bound at load time to the SL segment, but LOADPROCs a procedure within it, will use a LCL# at least 1 greater than the highest LCL# assigned to that program's full load-time set of private and shared code segments. With all of these different LCL#s pointing at them, transfers at run time from one user SL segment to another would be problematical without some outside help, because, like the status register and stack markers, external *plabels* in STTs employ the 8-bit-segment-number-plus-mapping-flag construct (see Appendix I). Because they are embedded in the code segments themselves, they clearly cannot refer to their target segments by LCL#. MPE V solves this problem by making these external *plabels* point to local segment 0, which says to the CPU, "Use the LCL# in the status register as key to the ELL in the current process' LSTT to find the operative *plabel*." David Holinstat, in his 1983 paper "Architectural Changes for MPE V", gives a detailed explanation of how this whole scheme works.

The "Personnel Manager"

We have seen how the association is maintained between the LCL#s and CST#s of user SL code segments shared by a process--the question now is, how are those numbers associated with their LOG#s "back home"? This is part of the broader question, "How does the system keep track of where all these program and SL code segments come from and who is sharing them?"

The time is ripe to make our acquaintance with a magnificent structure, the Loader Segment Table (LST). This object comprises two data segments: the LST proper, whose DST entry assignment is fixed, for tracking loaded program files, any processes that are sharing them, and SL files from which at least one segment is loaded; and the eXtended Loader Segment Table (XLST), whose DST index is held in the SYStem GLOBal Area (SYSGLOB), for tracking SL procedures that are either dynamically loaded or allocated. The two data segments are identically structured. For the present, we will focus on the LST proper.

The LST contains directory entries of various types and sizes, of which three are useful: Type 1 entries for SL files (including the system library); Type 2 entries for program files; and Type 6 entries for processes sharing programs for which Type 2 entries exist. The rest seem to be purely administrative in function, some existent only while the Loader is actively doing something. All entries are organized into doubly-linked lists by entry type, with chain heads in the global section of the table.

There is one Type 1 entry for each SL from which at least one code segment is loaded. This entry contains two things of primary interest: the file disc address of the SL; and a SEGLIST array, consisting of one 3-word subentry for each loaded code segment and containing both its LOG# and its CST#--and here we find the connection between "home" and "workplace".

There is one Type 2 entry for each loaded user program. In addition to a wealth of information about the library search and other load options, the program's allocation status, and any user SL code segments it is sharing, this entry contains the file disc address of the program and its CSTXEIX, an item shared in common with the PCB of any process running that program. Here, there is no need for a SEGLIST array, since a program's segments are loaded and unloaded as a block, not separately--their LCL#s (constant for all sharing processes) under the umbrella of the CSTXEIX serving in lieu of CST#s, and related arithmetically to the LOG#s.

There is one Type 6 entry for each process sharing a user program (for which a Type 2 entry exists). This entry contains the disc address of the program file and the Process Identification Number (PIN) of the sharing process--and so the program is connected directly to the people who are running it.

I have taken care to say "<u>user</u> program", because some system processes are not loaded from program files as such. The Command Interpreter program is a case in point; <u>all</u> of its code segments are mapped in the CST.

All program and SL file disc addresses are L-types.

The two diagrams following "Conclusion" should help to clarify the interrelationships I have described so far, but I haven't yet told you everything there is to know; some material I considered best presented in the context of the application examples that follow.

Application Example 1: Restricting File Access by Program File (Security)

It is often desirable to provide access to a file to a specific application program, or set of such programs, without allowing global access to the file.

Suppose that you have an application program that reads and updates certain run-time parameters kept in a "control" file. READ and WRITE access to a file is granted by the security matrix structure according to the capabilities of <u>users</u>, not programs, so to prevent them from tweaking the control parameters "on the sly" with MPE utilities or their own programs, you protect the control file with a lockword, and apply Gandalf's advice to Frodo about Bilbo's magic ring to said lockword: "Keep it secret, and keep it safe."

"Secret" and "safe" usually means known only to the MIS staff and the application program itself, and it usually translates in practice to embedding (i.e. hard-coding) the lockword in the source code. But according to Adager's Alfredo Rego, hard-coding anything subject to change from outside the program is not really sound methodology:

> **A very important programming standard is that you should postpone *binding* as much as possible. This means that you should not burden your programs, at compilation time, with hard-wired stuff. You should wait until run time to adjust to the prevailing conditions of the day.**
>
> -- F. Alfredo Rego, "Database Indexing: The Key to Performance" (1991 INTEREX Proceedings, San Diego, CA)

Hard-coding lockwords in application programs makes it necessary to dig out, modify and recompile their source code every time a lockword change is desired. Nor would they be either secret or safe even in the program files unless READ access to them, as well as to the source code, is secured.

Eliminating compile-time binding to your control file's lockword requires that a part of your application code have the PM capability needed to interface with the control file in one of two ways:

1. Retrieve its entire file label (lockword and all), using the privileged CSL procedure DIR'READ, from which non-privileged code can build a complete file designator string for FOPEN.

2. Carry out all control file access on behalf of the application. Here, you may circumnavigate the lockword altogether by using the alternate file-opener MUSTOPEN (see Eugene Volokh's paper "Burn Before Reading--HP3000 Security and You"). Setting privileged access type READ IGNORING SECURITY (*aoptions* (12:4) = %17) would enable you to fully secure the control file through the standard matrix, making a lockword unnecessary; but then, you wouldn't be able to update it.

You then face the problem of preventing just anyone from running your privileged code.

VESOFT's SECURITY/3000 system already provides a means of keeping IMAGE database passwords out of application source code. In so doing, it introduces the concept of basing run-time security enforcement on <u>the identity of the program being run</u>--and that is the concept we will apply here.

Your privileged code could be confined to a separate "slave" program that allows itself to be run as a "son" by a process running an eligible program. Breaking out privileged operations into separate and temporarily-privileged processes relieves application programs and accounts of the burden of having to possess PM themselves, and provides the relative safety of running in PM only when absolutely necessary; you can even safely employ the technique, set forth in "Secrets...", of acquiring process-local System Manager (SM) capability. The downside of it is, it tends to increase the proliferation of PM programs, which must be monitored carefully.

The first thing your slave must do is to positively identify the "father". The quickest way is to pass the PIN returned by the FATHER intrinsic to a Privileged Mode call to PROCINFO requesting item 10, and compare the returned "name.group.account" string against your eligibility list (which could be maintained in a separate, and tightly secured, file).

Sounds easy enough, doesn't it? So why do you need to know where and how PROCINFO gets this information? Because there is something else you need to know that PROCINFO doesn't tell you--whether the program file is permanent or temporary!

Why is this a security issue? Well, if Joe Q. Sinister doesn't already know, I don't want to be the one to tell him! I will only say that the other half of the problem is enforcing the requirement that the father program reside in a permanent file.

A fully reliable way to obtain all pertinent information is to get the father's Type 6 entry in the LST, for this points to the actual program's file label--your "one-stop shop" for its name, group, account AND *foptions*.

Using the privileged CSL procedure SEG'READ, you must serially traverse the Type 6 chain until you find the one entry containing the father's PIN, and since the LST is a highly dynamic linked-list structure, it would be wise to single-thread with the Loader by obtaining and releasing the LST SIR around your search.

Now, my Tables manual is fairly clear about where the chain heads are, and the Type 6 layout is almost self-explanatory; nowhere does it say, however, that all non-zero chain heads and links in the (X)LST point, not to word 0 of the next entry, as one would normally expect, but to word 3! (As to why this idiosyncracy, my guess is that two-way chaining and entry sizing were added after the original table design in such a way as to avoid having to change a zillion array indices in the Loader code.)

When reading a file label given its L-type disc address, you should always call the system procedures CHECKDISC and DISCSIZE to validate the LDEV and sector number, respectively, as a safety measure, before passing them (as separate parameters) to the system procedure FLABIO. In the unlikely event that the address is corrupt on the LST, the system probably should have crashed already, but such possible corruption isn't the issue; unless this simple precaution is taken, Heaven forbid your program should inadvertently read the file disc address from the wrong location, for FLABIO is as catastrophically unforgiving of invalid parameters as any MPE internal can well be!

Now that you've secured your control file, you need to think about maintaining the general security of your system. I would suggest the following:

1. Secure your slave program and all application programs eligible to run it as you would programs with PM capability, i.e. by securing WRITE access to the program files and SAVE access to their respective groups, and NEVER :RELEASEing them. This applies to the eligibility list, also.

2. Restrict the installation of eligible application programs, and maintenance of the eligibility list, to MIS personnel whom you can trust as a System Manager.

ASIDE -- 4GL APPLICATIONS

While we're on the subject of screening programs attempting access, we may as well touch all the bases.

The widely-used PowerHouse products of COGNOS, Inc. present a special difficulty not encountered with programs written in conventional languages like FORTRAN and COBOL; for, instead of a program dedicated at compile time to specific operations on specific data, you are now dealing with a 4GL *driver* that does whatever the user tells it to do with whatever data the user tells it to access, through a "program" of descriptive statements or "compiled" specifications input at run time, either from a disc file or interactively. Thus, as far as the Loader is concerned, all QUIZ processes, however much they may differ in what they are actually doing, are running the same program loaded from the same PROG file; and this holds true analogously for QUICK and QTP processes.

To allow PowerHouse access to the control file through your slave program, the eligibility list would have to include the drivers; yet, this by itself would obviously grant access to anyone who simply knows how to write and execute a program that runs the slave. Deciding which PowerHouse programs may or may not gain access seems least problematical in the case of QUICK screens, for both the QSCHEMA and screen files are opened by the QUICK driver on channels whose MPE file numbers work out to be the same every time it is run, subject to change only when QUICK is upgraded to a newer version. Carolian Systems' SYSVIEW utility can help you find out from a QUICK driver process which file numbers those are.

How to ascertain the files opened by another process, requiring as it does a working knowledge of the File System tables, is beyond the scope of this paper. Yet, I considered it important to include 4GLs in the discussion, since at many installations, much, if not all, file and database access is carried out through 4GLs like PowerHouse, rather than through programs readily distinguishable one from another at Loader level.

Application Example 2:
A More Down-to-Earth Stack Dump, Using FPMAPs
(Performance)

Like many of you, I have found the Stack Dump Facility an effective debugging tool, but very awkward to work with; its output is cryptical, to say the least, and not sufficient unto itself, requiring the availability of other hardcopy, including (and especially) PMAPs.

The Segmenter's FPMAP feature, for embedding a machine-readable PMAP in a program file or SL segment, offers the potential, in the event of a trappable error condition, to programmatically identify by name the procedure wherein the problem occurred and report it in a much more readable form of stack marker trace-back than is provided by STACKDUMP, without requiring a separate PMAP listing (except for the stack locations of COMMON blocks). The FINDPMAPADDR intrinsic returns an FPMAP record containing the name of a procedure and its *start address* (or *code offset*), given the file number of an opened program or SL file, a LOG# and some offset, arbitrary or otherwise, within the code segment. The problem is knowing what LOG# to pass, as well as where to look for an FPMAP record.

The key to where your process is in your program's hierarchical structure is an elegantly simple 4-word system table, the *stack marker*. It is too well-documented elsewhere to need any introduction here, so let's get right to the stack marker data relevant to this application:

ADDRESS	BITS	DATA
Q-0	(0:16)	Delta-Q: relative backward link to the previous stack marker (except in the "initial" one).
Q-1	(8:8)	Segment number: LCL# if mapping flag is 0; CST# if mapping flag is 1.
Q-2	(1:1)	Mapping flag.
	(2:14)	PB-relative return address within the indicated segment; points to instruction following the PCAL that generated this stack marker.

To determine your trace-back limit, read "INITIAL Q" (Qi) from PXFIXED (in your stack below DL) and add 4; the sum is the location of the *initial stack marker*, which remains on the stack until the process terminates. When you get this same value after subtracting a delta-Q, you have reached what is analogous to an "end-of-file" (the initial stack marker itself contains no useful information).

Before we walk through the stack marker analysis, I must call your attention to three things that my Tables manual does not indicate in the schematic for the LST Type 1 entry:

1. Under MPE V/E, the system library can have as many as 510 code segments. For this reason, the "SEG ARRAY" bit map in the system library's entry needs to be 32 words long, but for all other SLs, it retains its original 16-word size; consequently, the offset of the SEGLIST within the Type 1 entry is variable! My explorations with privileged DEBUG have further revealed the presence of an extra word, the purpose of which is obscure to me, sandwiched between the "SEG ARRAY" and the SEGLIST. Therefore: If the Type 1 entry belongs to the system library--which you may quickly determine by comparing the file disc address to that of the system library held in SYSGLOB--then the SEGLIST is at offset 41 from the base of the entry; otherwise, the offset is 25.

2. Since more than 256 code segments are allowed for the system library, the "LOG SEG NUMBER" field in the SEGLIST subentry is expanded to 9 bits in all Type 1 entries; and it would make sense for this also to be the case with "# SEGLIST ENTRIES". What happens then with "# ALLOCATED SEG"? From experiment and observation with :ALLOCATEing a (system library) procedure, this latter field appears to be now obsolete, since, absent any definite limit on the number of system library segments out of the possible 510 that may contain an allocated procedure, it would also have to be 9 bits in size, and there obviously aren't even 8 bits left in that word. The field is irrelevant for user SLs, anyway, so I would feel safe in ignoring it altogether.

3. A SEGLIST in any Type 1 entry may, within the range of "# SEGLIST ENTRIES", contain a subentry for LOG# %777--which is, of course, an illegal value, the highest valid LOG# anywhere being %775. I believe this indicates a subentry vacated when a segment was unloaded. However, the rest of the subentry is zeroed out, so as long as you're targeting on CST#, you needn't check the LOG# for validity.

Nor must I forget to mention what the Intrinsics and Segmenter manuals both neglect to, about using FPMAP intrinsics. When servicing a :RUN command or Process Handler request, the Loader uses the LOCK *aoption* (or some equivalent thereof). Since LOCK and NOLOCK are mutually exclusionary, this means that anyone FOPENing or :FCOPYing a program or SL file should specify optional locking, so as not to cause a problem for another user trying, at the same time, to run that code. It is not necessary, however, to FLOCK the file before reading FPMAP information.

Here is the logic for translating the contents of each stack marker in the trace-back:

1. HOUSEKEEPING.

   You are going to be opening some files, so to reduce the possibility of running short of stack space, call the ZSIZE intrinsic with a parameter of 0 (see Eugene Volokh, *TAD*-3, p. 135).

   At least one of your code segments is in the program file. From the LST, get the Type 6 (sharer) entry for your PIN, FLABIO the file label, and MUSTOPEN the file, using the fully-qualified name and file domain *foption* copied from the label, and *aoptions* %377 (SHR, LOCK, MR, READ IGNORING SECURITY). I recommend MUSTOPEN, because a program can have a lockword, and this saves you the work of having to test it for non-blank and insert it in your file designator string. More strongly do I recommend including "No :FILE" in your *foptions*, to ensure that the file you specify is the file that will be opened.

   Get from the PCB of your process the DST index of your LSTT. This number should not be 0, since your current "ego", so to speak, is a logically-mapped SL segment.

   Get the program segment count (PSC) from SYSGLOB, where it is placed by the Dispatcher for the currently-executing process, in a cell curiously mislabeled "TOTAL SEGMENT NUMBER OF CURRENT PROCESS". For this purpose, you may avail yourself of the system procedure SYSGLOB (see Eugene Volokh, *TAD*-4, p. 153).

   Also get from SYSGLOB the file disc address of the system library (SLA).

   Get the DB-relative location of the initial stack marker:

   ```
   Qi = contents of DL-'DL-2'+3
   QOB = Qi + 4
   ```

   Get the Q (DB-relative location) of the above (current) stack marker:

   ```
   Qx = Q
   ```

2. TOP OF LOOP.

   Read the mapping flag from Qx-2 (1:1) in the stack marker you are now pointing to. If it is 0, go to step 6.

3. The mapping flag is on. Immediately, you know that you're dealing with an MPE segment in the system library, and that its CST# is right there in the lower byte of Qx-1. Your target Type 1 entry, therefore, is the one for the system library, where the SEGLIST array is at offset 41.

Obtain the SIR for the LST and get from its "PRIMARY DB AREA" the forward chain head for Type 1.

4. (NOTE: Remember to subtract 3 from the next-entry pointer to get the correct table-relative offset. You should never encounter a 0 link before the end of your search.)

   SEG'READ the first six (6) words of the next entry. If the file disc address ≠ SLA, read the next-entry pointer from word 0 and repeat this step.

5. You have found the system library's Type 1 entry. Using the entry size in word 2, SEG'READ the full entry and release the LST SIR.

   Serially read the SEGLIST until you find the subentry with the target CST# (unless something is seriously amiss, it will be there), and read the LOG# from the upper 9 bits of the first word of that subentry.

   Read the return address from Qx-2 (2:14) in the stack marker, and subtract 1 to point to the PCAL or trapped instruction.

   Since this is an MPE code segment, there seems little point in looking for an FPMAP record, so just issue your stack marker display on that basis and go on to step 11.

6. The mapping flag is off. This means that you have, in Qx-1 (8:8), a LCL# pointing to either a program segment or a user SL segment.

   If LCL# ≤ PSC, then the LOG# (in the program file) = LCL# - 1 AND you must go to step 10.

7. LCL# > PSC and is, therefore, pointing to a user SL segment. Get the CST# from the LSTT at offset LCL# * 2.

   Obtain the SIR for the LST and get from its "PRIMARY DB AREA" the forward chain head for Type 1.

8. (NOTE: Remember to subtract 3 from the next-entry pointer to get the correct table-relative offset. You should never encounter a 0 link before the end of your search.)

   SEG'READ only word 2 of the next entry for the entry size, then SEG'READ the whole thing.

   If the file disc address = SLA, then the SEGLIST begins at word 41; otherwise, it begins at word 25. Serially scan the SEGLIST for a subentry containing the target CST# (from the LSTT). If there is no hit, read the next-entry pointer from word 0 and repeat this step.

9. You have found the Type 1 entry with the SEGLIST subentry that contains the target CST#. Release the LST SIR. Read the LOG# from the upper 9 bits of the first word of the subentry.

   If this is the system library's entry, then you already know exactly where the segment came from, and where to look for an FPMAP record; otherwise, using the file disc address, FLABIO the file label for the group, account and file domain. MUSTOPEN the SL using *aoptions* %377 and *foptions* including "No :FILE". (Not that an SL would ever have a lockword--it's just less confusing to use the same file-opener as for the program file.)

10. You are now ready to look for an FPMAP record. If you got here from step 6, you are looking for it in the program file, unless you've already looked for one there and failed. If you got here from step 9, seek it in the SL that you have just MUSTOPENed.

    Read the return address from Qx-2 (2:14) in the stack marker, and subtract 1 to point to the PCAL or trapped instruction, to ensure that at display time, it will be pointing into the program source statement in process. Pass the adjusted address to the FINDPMAPADDR intrinsic along with the LOG# (from step 6 or 9) and the file number (from step 1 or 9).

    If you were looking in an SL file for an FPMAP record, you may FCLOSE it at this time.

    If you were looking in the program file for an FPMAP record, and your FINDPMAPADDR call is unsuccessful, you may FCLOSE it at this time and not bother to look there again; either all program segments have FPMAPs, or none of them do.

    If your FINDPMAPADDR call is successful, you have a wealth of information for your stack marker display, including the name of the primary entry point and the name of the code segment, to go with the exact name and location of the file it was loaded from. Subtracting the start address in the FPMAP record from the PCAL or trap address gives you the offset within the subprogram unit, compatible with the location numbers in the compile listing; these numbers are generated by $CONTROL directive options like these:

FORTRAN 66:	LOCATION
FORTRAN 77:	CODE_OFFSETS
COBOL II:	VERBS

    By the way, if the delta-Q links back directly to the initial stack marker, then the one currently being analyzed points into your program's *outer block* (the main program unit where the process began execution--which, incidentally, does not necessarily reside in local segment 1); you may want to note that in your stack marker display.

11. BOTTOM OF LOOP.

    Set the pointer to the next previous stack marker:

    ```
    Qx = Qx - delta-Q
    ```

    If Qx > QOB, go back to step 2.

12. CLEANUP.

    You have completed the trace-back analysis; proceed with any other error-reporting business.

It should be noted that a process is capable of dynamically loading and calling a procedure from <u>any SL on the system</u> with the aid of a "son" process running the CSL program XLOGON5 ("Cross-Logon")--a powerful utility with frightening potential as a system penetration tool if not carefully managed and secured! However, the beauty of this trace-back design is that no confusion results if an SL is not in either the program's or the current logon's library search path.

For global callability, independent of user and program capabilities, your custom procedure must be "OPTION PRIVILEGED" in the system library. The chief danger in this--especially if you have any parameters passed by reference--is that, being thus *permanently privileged*, your procedure code is deprived of the safeguard of automatic bounds-checking. But there is one advantage that I haven't seen mentioned anywhere:

It would be nice if you could display the stack marker data without having to open a new file on $STDLIST or count on coöperation from the caller in that respect. The PRINT intrinsic, of course, would allow you to do this with or without privilege; the good news is that the system opens $STDIN and $STDLIST on MPE file numbers 1 and 2, respectively, on behalf of every user process, and that these channels can be accessed <u>directly</u> in PM. This gives you greater flexibility in terms of how you want to handle your display output; you can even write segment-internal display routines in FORTRAN 66 using formatted WRITE statements (the Segmenter would not object to a variable set to -2 in place of a logical unit number).

Application Example 3:
Who's Using That Code Segment?
(System Management)

You have just finished testing a modification to a library procedure shared among several applications. It's after hours, and all users should be logged off and in rush-hour traffic. You are grabbing these few precious minutes before the backups start to install the new version of the code segment in the account SL where it resides; but when you try to -PURGESL the old version, you get Segmenter error 110: "SEGMENT CURRENTLY LOADED"; which may be roughly translated, "I'm sorry, Dave--I'm afraid I can't do that."

How on earth could your code segment be loaded at this late hour? Possibly, one of your people forgot to exit an application and log off before leaving the office (not very good for system security); possibly, someone is working overtime to catch up on a backlog of work; or possibly, everybody has logged off, but a sharing program is still allocated. Rather than have your valuable time consumed with making hit-or-miss inquiries, wouldn't it be nice to have a utility program that can tell you in a matter of seconds who is using your code segment, and where, given its LOG# and SL group and account?

I have hardly more than mentioned the Type 2 (program file) LST entry; I must now describe it in more detail, and also introduce you to Type 7 (extension).

As I have said, there is one Type 2 entry for each loaded user program, and that it contains information about user SL code segments shared thereby. This information resides in an SLINFO area which may contain as many as three (3) subentries, one for each SL actually containing code segments required at load time, pursuant to the library search option specified when the program was first loaded from an unloaded state (thus, the subentry for the system library is always the last in sequence).

(From this, you should be able to guess the reason why if, for example, a program is currently being run ";LIB=S", and you wish to run it ";LIB=P" or ";LIB=G", you'll just have to wait until nobody else is running the program, and then if necessary, :DEALLOCATE it.)

The "LIB SEG ARRAY" part of the SLINFO subentry is a bit map which is 16 words long (254 used plus 2 unused bits) for all SLs, except, once again, for the system library which requires 32 words to cover all 510 possible code segments. Each bit corresponds to a LOG# and is on if the segment is shared by the program (at least, this is always the case for a user segment; a shared MPE segment is flagged or not depending on other conditions that I haven't quite figured out, but which really don't concern us here). Once again, the SL where the code segment "lives" is identified in the subentry by an L-type disc address.

A Type 7 entry is generated in the XLST whenever a process dynamically loads a procedure, and remains until that process either terminates or calls UNLOADPROC passing the same identity number returned by the creating LOADPROC call. There is one Type 7 entry per unique procedure per process, and it contains: the 1- to 15-character name of the procedure; the caller's PIN; the *plabel* returned to the caller; and an SLINFO block, just like the one in a Type 2 entry, and serving a similar purpose. The first subentry in the SLINFO is for the SL in the specified search path to which the procedure reference is actually bound.

So, you see, your code segment could be loaded either because some program requires it at load time, or some process requires it at run time, or for both reasons; hence, you need to locate any and all SLINFO subentries pointing to your SL file among both Type 2 and Type 7 entries. These subentries are marked by the L-type address of your SL's file label, which may be obtained by opening it in the manner prescribed in Application Example 2, then calling either FGETINFO or FFILEINFO.

Whenever you find an SLINFO subentry with the matching file disc address, you must check the "LIB SEG ARRAY" to see if the bit for your LOG# is set to 1. To help you locate it, imagine your LOG# to be subdivided as follows:

Bits (7:5)	Word offset within the bit map.
Bits (12:4)	Bit offset within the word.

In the case where a qualifying SLINFO subentry is found in a Type 2 entry:

You have identified a program bound to your code segment at load time. Before looking for sharing processes, however, you should check not only the sharer count ("# PROCESS SHARING"), but also the load option bits. Through experiment and observation (again, necessitated by deficiency of detail in my Tables manual), I have found that if and only if a program is permanently allocated (through the :ALLOCATE command), indicated by bits 1 and 3 of word 3 set to 1 and 0, respectively, the sharer count is greater by 1 than the true number of sharing processes. In any event, if a program is either :ALLOCATEd or AUTOALLOCATEd, the net sharer count could be 0.

Sharing processes may be found either by chasing the Type 6 chain for entries pointing to the same program file, or by serially scanning the PCB table for assigned entries pointing to the same CSTX block; this latter method seems preferable, since you wouldn't have to tie up the LST SIR for quite as long. With a sharer's PCB in hand, it is a relatively simple matter to key into all process- and job-related information.

In the case where a qualifying SLINFO subentry is found in a Type 7 entry:

You have identified a process bound to your code segment at run time. How would your utility program do the following?

- Find out what program the process is running, and its file domain.

- Find out if the procedure was LOADPROCed from the code segment in question. (HINT: Use the *plabel*; it is layed out as described in Appendix I.)

Bear in mind that the same PIN may occur in two or more Type 7 entries, and that more than one of these may conceivably contain a qualifying SLINFO subentry.

Let's change the problem a bit--suppose the code segment in question had been installed in the system library, rather than localized to an application account. Your program finds a qualifying SLINFO subentry in a Type 7 entry--but lo! and behold, the PIN is zero!

I'm sorry, I should have warned you: The XLST Type 7 entry is also used for tracking system library procedures that have been allocated through the ":ALLOCATE PROCEDURE" command. The entries for these contain a zero PIN, and the procedure's LOG# instead of a LOADPROC count. Your sweep of SLINFO blocks would pick up these entries as well as any generated by LOADPROCs, and whenever you find a qualifying subentry in a Type 7 with a zero PIN, you have the name of a procedure that must be :DEALLOCATEd in order to free up your code segment. And, of course, if the procedure just happens to reside in the code segment in question, then your target LOG# is explicitly matched in the main part of the entry.

ASIDE -- A POSSIBLE SHORT-CUT TO JOB-RELATED INFORMATION

You may want to save yourself some tables legwork by calling the JOBINFO intrinsic, once you have obtained the job/session number from a sharer's Process Control Block eXtension (PCBX). You may also want your utility program to be available for use by operations personnel who possess System Supervisor (OP) capability, but not SM--in which case, beware of this pitfall:

When the request is for information about a job or session other than the current one, JOBINFO requires that you the user have SM capability and, furthermore, doesn't care what mode you're in when you call it. No problem--the program has PM, so all you have to do is set the SM capability bit in your own PCBX, and you're in business. Right?

No such luck! This technique works well enough for FOPEN and some other intrinsics--not for JOBINFO, for it checks the user capability mask in the Job Information Table (JIT) for the current job/session. Your utility would have to inspect the SM bit there (see my note on the WHO intrinsic in Appendix II) and, if necessary, set and clear it around the JOBINFO call, taking whatever measures are necessary to ensure that the interactive user does not end up with SM capability for the rest of the session.

Application Example 4: Securing Usage of Sensitive Procedures

Let's suppose that, rather than in a separate program, the privileged operation described in Application Example 1 is handled by an "OPTION PRIVILEGED" procedure in an SL dedicated to the application.

We're still presuming, of course, that none of the users and application programs possess such powerful capabilities as SM or PM; only the group and account containing the application SL must now have PM capability, and only one or two people entrusted with SM and PM have WRITE access to the group and to the SL.

There is a simpler and more efficient way to secure the operations performed by an SL procedure, and the information it returns to its caller, than by checking the identity of the program where the calling process is running; and that is to design it to enforce the requirement that it be called only from within its SL of residence, not from the program or any other SL. In fact, any "application internal" procedure that retrieves and returns sensitive information that ought not to be directly available to just any user program that calls it, or that performs some "primitive" operation on behalf of a more sophisticated module--but, due to design constraints, is stuck with being "OPTION EXTERNAL" and callable--such a procedure may dynamically make itself internal to the SL with the help of a general utility procedure added to the system library.

This utility procedure could be structured like this:

```
LOGICAL PROCEDURE SAMESL;
OPTION PRIVILEGED;
```

SAMESL finds, in its above stack marker (at Q), the LCL# of the application internal (Callee); the LCL# of the Caller is in the next previous stack marker (at Q'). As in Application Example 2, let

PSC = program segment count
QOB = location of initial stack marker

The Callee's LCL# > PSC, and Q - delta-Q > QOB, provided that SAMESL is being used properly. If the Caller's LCL# = the Callee's LCL#, then immediately we know that the call to the internal is legal, and must return TRUE; obviously, Caller and Callee reside in the same SL segment. If the Caller's LCL# ≤ PSC, or Q' - delta-Q' = QOB, then immediately we know that the Caller is not in the same or any other SL, and must return FALSE.

If the Caller's LCL# is pointing to a different SL segment than that of the Callee, then we have to interrogate the LST. Our security requirement really boils down to both CST#s being present in the same SEGLIST array, so all we need to do is use either the Caller's or the Callee's CST# to find the Type 1 entry of its "home" SL, then look for the other one's CST# in the SEGLIST; we don't have to read any file labels or compare any file names.

OH! BY THE WAY--if you're still concerned about the identity of the accessing program--in this case, the current one--and your access filter recognizes it as a QUICK driver, finding out which screen is being run is much simplified, with no sophisticated tables expertise or even PM necessary, because the QSCHEMA and screen files have been opened by the current process! So, just call FGETINFO or FFILEINFO against the appropriate file numbers.

A Word to the Wise

The complexity of the foregoing application examples--especially the second and third--compels me to remind you of several cautionary points that Eugene made in "Secrets..." about SIRs; these may be found in *TAD*-3, pages 242-245.

> * **Never do any terminal I/O while you have a SIR locked. In fact, do not perform any task that you suspect might take a long time to finish...**
> -- Eugene Volokh, "Secrets of System Tables...Revealed!" (1985 INTEREX Proceedings, Washington, DC)

Quite frankly, where the LST SIR is involved, I would advise against doing <u>any</u> I/O, PERIOD! For one thing, a FORTRAN I/O statement, or call to FNUM or UNITCONTROL, using a previously unreferenced logical unit number generates a call to FOPEN, which obtains the File Multi-Access Vector Table (FMAVT) SIR, which has <u>the highest priority of all SIRs</u>! You have to realize that:

> **...there is a certain very fixed and unalterable SIR locking sequence that you MUST obey. ...Each SIR is given a priority** [or Rank] **number, and you must never lock a SIR whose priority number is lower than that of one you already have locked.**
> -- Ibid.

Generally, while traversing an LST entry chain, you should be careful of what system intrinsics you call deliberately, and be aware of what system intrinsics, if any, your home-grown procedures call or that your source language statements cause to be called "behind your back", and:

> **...you better know what SIRs they try to lock, and you better make sure that your SIR locking sequences are appropriate.**
> -- Ibid.

> * **When you need to get a SIR, release it as soon as possible. Every statement between a GETSIR and a RELSIR is just one more opportunity for a disastrous (system-crashing) abort to occur.**
> -- Ibid.

While holding the LST SIR in Application Example 3, you should collect only essential key information--PINs, CSTXEIXs, file disc addresses, etc.--in arrays of appropriate size on your stack, and defer reading file labels or chasing process and job/session information until after you have gathered all you need from the (X)LST and let go of the SIR.

- **Always disable break before locking a SIR. ...If you let a user hit break while you've got a SIR, the user might try to execute an MPE command that tries to get the same SIR, and you'll get a deadlock...**
  -- Ibid.

Definitely do this when obtaining a SIR within a program code segment. I question whether you really need to disable break within a privileged system library segment, but it wouldn't hurt. When executing anywhere in PM, you may call the FCONTROL intrinsic directly on MPE file number 1.

MOST CRITICALLY IMPORTANT OF ALL:

- **If your process terminates itself for any reason (TERMINATE, QUIT, stack overflow, or whatever) while you have a SIR locked, you get a System Failure 314. You best be *very* careful.**
  -- Ibid.

Conclusion

MPE V's code management structures rank with those of the File System in complexity, and the documentation generally is the most fragmented and confusing in this area; yet, here is revealed the most remarkable of all secrets of system tables!

Perhaps, you have noted with interest the dual rôles played by the CST indices of SL segments, and the CSTXMAP indices of program segment blocks. Apart from being keys to the code segments themselves in memory--a dead end for most practical application purposes--they are <u>common</u> <u>items</u> whereby one table containing information that really matters is <u>joined</u> to another!

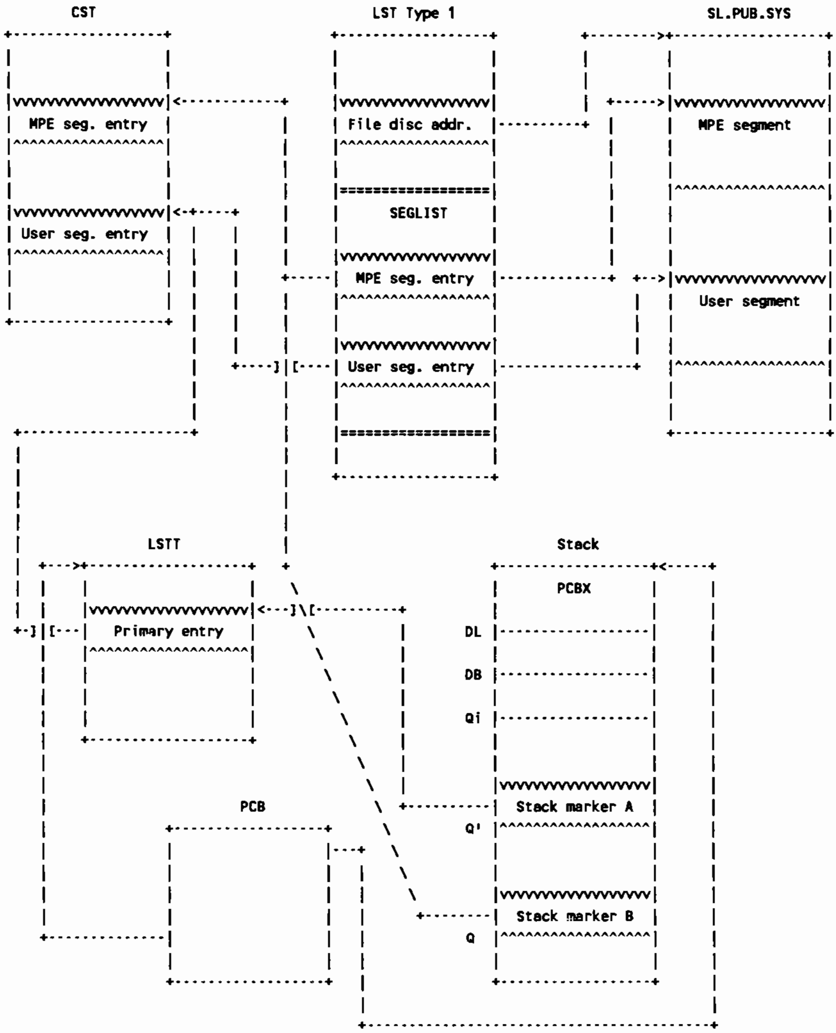
This is why I bade you keep in the back of your mind that seemingly irrelevant quote in the "Introduction". By sharing the benefit of this kind of interrelationship in common with relational databases, the MPE data structures need not provide indexing or chaining for every conceivable table cross-reference or detail linkage.

So, as Paul Harvey would say: "And now, you know--the *rrrest* of the story!"

- * -

LOADER SEGMENT TABLE

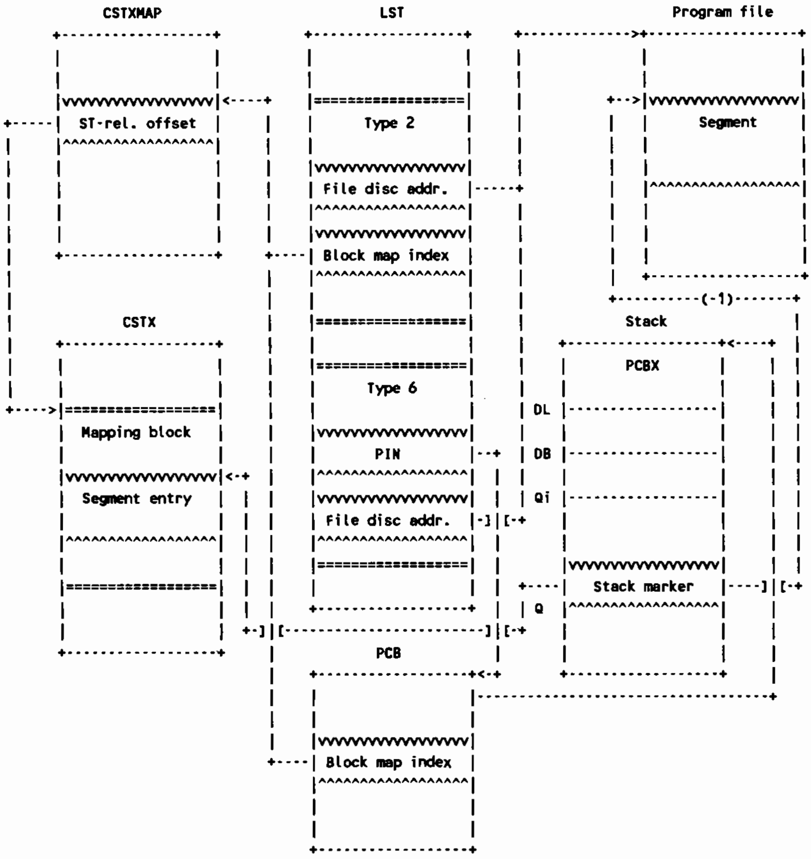
Type 1 (SL File) Entry



Stack marker A: Transfer from user segment.
Stack marker B: Transfer from MPE segment.

LOADER SEGMENT TABLE

Type 2 (Program File) and Type 6 (Sharer) Entries



Stack marker: Transfer from program segment.

- - - - -

The CSTX-relative location of a program block is given by:

'CSTXMAP(CSTXEIX)' - 'SYSGLOB(DFS)'

Appendix I:
A Note on *Plabels*,
or How Two Bytes Do the Work of Fifteen

The *plabel* returned by the LOADPROC intrinsic is in "external" format and layed out as follows:

Bit (0:1)	Mapping flag.
Bits (1:7)	STT index.
Bits (8:8)	Segment number (local or physical).

The STT index is an implied-negative offset from the physical end of the target code segment, pointing (in the STT) to the procedure's "internal" *plabel*, which consists of a segment-relative entry point address and a flag bit which is set if the procedure is "uncallable [in User Mode]".

LOADPROC does not itself, of course, call the procedure after loading it; in SPL code, this is done by first stacking whatever parameters, if any, are expected by the procedure, then stacking the *plabel* itself, then ASSEMBLE-ing a "PCAL 0" (a special form of PCAL that bypasses the STT lookup in the code segment of origin).

FORTRAN programmers who have despaired of calling dynamically-loaded procedures from their favorite language will be overjoyed to learn that this is exactly what FORTRAN object code does in a subroutine or function that receives a procedure reference as a passed parameter, for the EXTERNAL type declaration in the caller generates a *plabel* in the external format given above. Where a procedure name is specified among the actual arguments of a subroutine or function, what the object code does depends on which incarnation of FORTRAN you're using: FORTRAN 66 (aka FORTRAN/3000) passes the procedure's *plabel* by value; FORTRAN 77 stores it on the stack and passes the address. (Have you ever tried, in FORTRAN 77, to pass an *errorproc* subroutine name to the SORTINIT intrinsic?)

A *plabel* returned by LOADPROC is a fundamentally similar object and works the same way; it doesn't matter that it was generated at run time. So, the trick is to use the FORTRAN compiler itself to generate the "push-and-PCAL" code you need in a "gateway" procedure--as illustrated in the FORTRAN 66 example below, using the ASCII intrinsic as a "guinea pig":

```
 ...
C 'ASCII' HAS BEEN HOOKED; WE NEED TO CALL THE REAL THING.
      INTEGER ORIGASCII
      LOGICAL LVAL
      CHARACTER OSTR*6
      SYSTEM INTRINSIC LOADPROC
      ...
      IDENT=LOADPROC("ASCII ",0,IPL)
      ...
C GET DECIMAL STRING, RIGHT-JUSTIFIED.
      L=ORIGASCII(\IPL\,\LVAL\,\-10\,OSTR)
      ...
      END
$CONTROL CHECK=2
C              ^ SUPPRESS PARAMETER TYPE CHECKING.
      INTEGER FUNCTION ORIGASCII(OASCII,IVAL,IBASE,OSTR)
C
C LINK TO ORIGINAL 'ASCII' INTRINSIC; SAME TYPE AND CALLING
C SEQUENCE, EXCEPT FOR EXTRA PARAMETER FOR 'PLABEL'.
C
      CHARACTER OSTR*6
      INTEGER OASCII
C
C ('IVAL' AND 'IBASE' ALREADY BY VALUE)
      ORIGASCII=OASCII(IVAL,IBASE,OSTR)
      RETURN
      END
```

Gateway procedures like ORIGASCII are also callable from COBOL. The only real disadvantage of this workaround is that one size does not fit all.

Binding of subjective external references is always lateral or unidirectional in the library search path from program to system library. In other words, there is no way for a hard-coded procedure reference in, say, an account SL to be bound at load time to an entry point in a group (non-PUB) SL or in the program. However, a process executing in any code domain can call a procedure in any code domain if it is in objective possession of that procedure's *plabel*. The trap intrinsics, and the trap-handling constructs in FORTRAN, work on this principle.

Appendix II: Some Notes on the Original "Secrets..."

Capability Reporting

> **Just as the :SHOWJOB command reads the JMAT** [Job MAster Table]**, the WHO intrinsic looks at your JIT; all the information it gives you--capabilities, user, account, group, home group, local attributes, etc.--all come from the JIT.**
> -- E. V.; *TAD*-3, p. 254.

Not quite all the information comes from the JIT directly. Under MPE V/E, the first word of user capabilities (*file access attributes*) returned by WHO is actually read from the calling process' PCBX! When I first experimented, in a "plain vanilla" logon, with Eugene's technique for acquiring process-local SM, I found that WHO dutifully reported that I had SM whereas before I did not--even though I hadn't done a thing with the JIT!

While we're on the subject of capability masks--another highly useful one, which is not returned by any standard non-privileged procedure that I know of, gives the *general resource capabilities* of the program and is copied from the program file to PXFIXED. The FOS up through version V-Delta-4 provides no non-privileged means of testing for PM capability at the program level, other than the GETPRIVMODE intrinsic, which aborts the process if PM does not obtain.

In Defense of JPCNT

> **I haven't the foggiest notion of what the JPCNT is actually useful for, except for a little-used (if at all) device called Job SIRs. This is one table you can afford to ignore.** ...[It contains] **A bunch of bits the reason for which I could never fathom.**
> -- E. V.; *TAD*-3, pp. 254-255.

Indeed, the Job Process CouNt Table itself contains nothing that any program outside of MPE would need to know. HOWEVER--if you're going to write a program to inquire about the file equations, JCWs, temporary files and intra-job-sharable eXtra Data Segments currently in effect/existence in someone else's session (something along the lines of SYSVIEW), then the JPCNT index of that session is one item of information you cannot afford to ignore; for that number plus "some base" gives the number of the SIR--or "JIR"--that you must obtain to prevent any PUTJCW or GETDSEG calls, or operations on temporary files or file equations, that might be going on in that session, from updating its Job Directory Table (JDT) out from under you while you're trying to read it!

Consider this: When you FCLOSE a new file saving it as a temporary file, a new entry must be added to the Job Temporary File Directory (JTFD) section of your friendly neighborhood JDT. Because the JDT is maintained in a tightly-packed state, with no "dead" space between the directories or their entries, other information in the table migrates to make room for the new JTFD entry. I would expect FCLOSE to take the precaution of obtaining the JIR of your job/session, in case another active process in your job/session is concurrently trying to do something affecting the JTFD or some other JDT directory. In that case, one can hardly consider the JIR "little-used (if at all)."

The JPCNT index is a unique number, independent of the job or session number displayed by :SHOWJOB, which is assigned to a job or session (more precisely, to its JDT) in the same way that a PIN is assigned to a process, and recycled in the same manner. It ranges in value from 0 to one less than the maximum job process (or C. I. process) count held in JPCNT. The bit map in the table is used simply to keep track of which indices are assigned and which are available.

A job/session's JPCNT index is held in the PXGLOB (section of the PCBX) of every process associated with that job/session. The "some base" which must be added to the index to calculate the right JIR to lock the JDT is actually found in Chapter 5 of the Tables manual as the SIR number assigned to "1st JOB". A JIR is obtained and released through the same MPE internal procedures (GETSIR and RELSIR) used for SIRs, and the same cautions and warnings apply.

In summary: **The JPCNT provides a mechanism for assigning a unique resource identifier to a JDT, whereby intra-job access thereto may be single-threaded, and whereby a system management utility running in another job/session may lock same JDT while examining its contents.**

Bibliography

David N. Holinstat: "Architectural Changes for MPE V"; 1983 HPIUG Proceedings, Edinburgh, U.K.

Eugene Volokh: "Burn Before Reading--HP3000 Security and You"; 1984 HPIUG Proceedings, Anaheim, CA.

Eugene Volokh: *Thoughts and Discourses on HP3000 Software* (3rd and 4th editions); VESOFT, Inc., Los Angeles, CA 1987/9.

Acknowledgments

I would like to express my gratitude to the following for their encouragement, feedback and assistance:

Daryl Rennie and Chris Limb of *Tellurian*, who were the first to publish anything I've written;

Judy Fisher of the 1991 INTEREX Conference Committee, who was the first to accept a paper of mine for inclusion in the annual symposium;

Eugene Volokh of VESOFT, whose comments on an early draft of "MORE Secrets..." were extremely helpful and led to the correction of several errors;

Brian Becker of this year's Host Committee, for accepting this paper, and then having to put up with my pestering him with inquiries because of persistent foul-ups in the Postal Service;

Phil Engle, Ed Raposo and Bruce Hamilton of UE, for their help in preparing the hardcopy and presentation materials.

Most of all, my heart-felt thanks to

My wife Caryl,
for her abiding love and support,
and to whom this paper is dedicated

* * * * *

PAPER NO.: 3052

TITLE: Reload (Install) Cookbook on MPE XL Systems

AUTHOR: Donald E. DeFreese
McDonell Aircraft Company
5695 Campus Parkway, Bldg. 274
Dept. 462C, M/C 274 1125
Hazelwood, MO 63042-2338
(314) 233-3332

HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

#3053: Quick Access to Archived Data for the HP 3000 with Optical Technology

Husni Sayed
Deborah Cobb
Joe Nemeth
IEM, Inc.
P.O. Box 1889
Fort Collins, CO 80522
(303) 221-3005

1 Introduction

As an installation grows, it fights a constant battle between the need to maintain all of the information it has accrued, and the need to reduce the costs of storing a constantly increasing amount of data. As time goes on, more and more space is devoted to storing information that is less and less frequently accessed. While the information is not needed on a constant basis, when it *is* needed, it is needed now! Most installations cannot afford to keep such data using premium system hard disk space, yet keeping it on tape is inconvenient and time-consuming to retrieve.

The development of optical disk technology is revolutionizing some of the traditional categories of computer data storage. The appropriate use of optical technology is *not* as a replacement for either hard disk or tape technologies, but in a niche which requires both the unlimited capacity and the random access capabilities of these drives. These capabilities are a perfect fit for storing archival data. With this technology, users can have access to archival information within seconds or minutes, instead of hours or days.

2 The Information Explosion

At one time or another, every computer installation that has been around for more than a few years begins to experience an information explosion. Early on, an installation encounters few problems finding places to store all of its information. The explosion occurs as an installation grows—more and more work is done, more and more projects are completed—producing massive amounts of information that must be maintained. Typically, there are three classes of information which must be maintained: current information, backup information, and archival information. These three classes of information are discussed below.

2.1 Current Information

Current information is information that is both stored and accessed on a regular basis, to which users must have virtually immediate access.

- Current information is constantly stored and updated as it is changed by the user.
- Current information in continually accessed by users, and must be instantly available upon request.
- Current information is always dynamically changing as users create, modify, and remove files.

Current information includes such things as system files needed for daily operation (the operating system, editors, etc.); databases; records on current employees, students, or patients; and files that are needed for current projects. The amount of current information on a system stays relatively proportional to the number of users: as new files are added to the store of current information, other files become obsolete. Therefore, a storage device for current data does not need to have unlimited capacity.

> **A device used for storing current information must give users the ability to quickly and easily create, edit, modify, and remove information. This must be a fast, random-access device.**

2.2 Backup Information

A backup is a "spare" copy of all of the information stored on the system, in its most up-to-date form (as of the time the backup was made).

- Backup information is stored on a regularly scheduled basis, so that the system can be restored to its normal operating condition in the case of a disaster (major or minor) such as a disk crash, or inadvertent deletion of a critical file.
- Backup information is retrieved rarely, most often when a system or device goes down. The smallest unit that might be retrieved is a single file.
- Backup information needs to be available, but does not need to be instantly accessible.

The size of a backup will depend upon the backup method used by an installation. If an installation performs only full backups (copying all of the information on the system), then the backup information is equal to the size of the system. Some installations perform a full backup at regular intervals, and incremental backups (storing only what has changed since the last backup) in between. In some cases, incremental backups are impractical—if a database is stored on the system and a single record is changed in the database, the entire database file will be included in the incremental backup anyway. With many such large files, an incremental backup will save little, if any, time or space.

Similarly, the frequency with which a backup is performed will be installation-dependent. When a disaster occurs on the system, all information which has changed since the last backup will be lost. So the "safest" solution is to perform backup on a continuous basis. This, however, ties up the system and degrades performance. Many installations perform a backup in the evening after business hours, when the number of users on the system is lowest. With this method, the worst case would entail losing an entire day's work.

Since access to backup information is rare, and often a large portion of the backup is being accessed, random access to backup information is not necessary. So, backups are generally stored on offline serial devices.

> **A device used for storing backup information must provide a cost-effective means of storing large amounts of redundant information. Generally, slower serial devices are adequate.**

2.3 Archival Information

Archival information is the part of the current information that is no longer needed on a regular basis but must be kept around, generally for historical reasons.

- Current information becomes archival information as it becomes outdated. The frequency of this will vary from one installation to the next.
- Archival information is accessed more frequently than backup information, but not on a continual basis—it must be more readily available than backup, but not as instantly accessible as current information.
- Retrieving archival information differs from backup in that it generally entails extracting only a single record or small piece of information from a large file. Therefore, random access to archival information is important.

Archival information includes such things as old versions of operating systems; records on past employees, students, or patients; outdated databases; accounting information from past years; and information on past projects.

The biggest problem with archival information is that when it is accessed, it is typically to restore a single record or piece of information from a large file. For example, a doctor's office may need to access an old patient record, which is stored in a large database file of old patient records. If the information is stored on a serial device, retrieving a single record is a time consuming task, as illustrated later.

Another problem with storing archival information is that the amount of information constantly increases. Unlike current and backup information, which stay relatively constant in size, archival information simply grows.

> **The ideal archival medium offers virtually unlimited capacity that is cost-effective, with moderately fast access to stored information.**

2.4 The Relationship Between Information Types

The relationship between current, backup, and archival information is a dynamic one, dependent upon the particular organization of an installation. The following three examples illustrate the different needs of different organizations:

1. **Storing Student Records**

   In a university setting, there is a need to maintain student records, including information about the student, and current grades and class schedules. During each semester, this information is current. At the end of each semester, the grades and class schedules become historical. The student information remains current until the student graduates, at which time that information also becomes historical.

   In this setting, current information lasts for a fixed, relatively short term. Every semester boundary will guarantee that at least some of the current information in the student records database will become historical. In this case, information could be stored to archives at fixed intervals. Backups may or may not need to be performed on a daily basis as student records, at least by mid-semester, should not change often.

2. **Storing Medical Records**

   Medical records have no fixed turnover of current information. Some conditions require only a single visit, while others may entail months or years of care.

   In this setting, current information changes to archival information every time a treatment ends. There is no way to predict how often this will happen. Some of these will be very short-term, and some will be longer term. Generally, though, some information becomes historical relatively frequently. In this case, information will probably be stored to archives on a periodic, frequent basis. Backups on a daily basis are essential in this case, as new information is added to current patient records.

3. **Storing Employee Records**

   Employee records have variable boundaries, as this information becomes historical as employees leave the company. Hopefully, this is not a frequent occurrence; but it usually cannot be predicted. In this case, information will probably be stored to archives on a periodic, infrequent basis. Again, backup of employee records may not be required on a daily basis.

3 Common Organization of a Data Processing Center

A typical data processing center in the HP 3000 environment is shown in Figure 1.

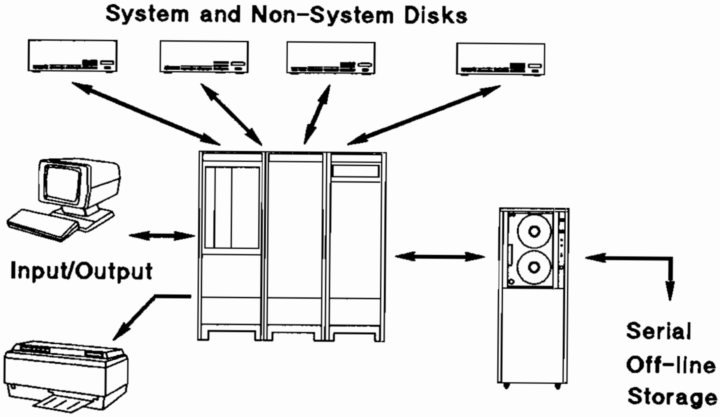


Figure 1: Typical Data Processing Center

3.1 Equipment Configuration

In this environment, the HP 3000 is connected to terminals and printers, and different storage devices. The types of storage devices that are typically configured into the system can be divided into two categories: fast online storage, and slow offline storage. Fast online storage is provided with system hard disk drives, which offer fast random access to files. The disk drives may be configured as system disks or non-system volume sets (private volumes). Slow offline storage is provided with tape drives. The attributes of each of these types of storage is shown in Table 1.

	Hard Disk	**Tape**
Capacity	fixed	unlimited
Storage Cost	high	low
Random Access Speed	fast	very slow
Serial Access Speed	fast	moderate

Table 1: Hard Disk and Tape Storage Device Attributes

Hard disks are fast random-access devices, their down side being high cost and fixed capacity—in order to increase storage space, you must add more devices to the system. Tapes, on the other hand, are low cost devices with unlimited capacity—to increase capacity, you simply need to buy additional storage media. Tape drives, however, are extremely slow and limited to serial access.

The drawback to this common data processing configuration is this: the system has two types of devices (fast online, and slow offline), but there are three types of information that need to be stored (current, backup, and archival).

3.2 Accessing Current Information

The essential feature of a device used for storing current information is the ability to quickly and easily access, create, edit, and remove information. The information must be completely and directly accessible to the user. In the common data processing center, access to current information is generally provided with hard disks attached as system or non-system volumes.

This is ideal for current information. The high cost of using hard disks for fast online storage is not a major consideration, as the amount of current information on a system remains fairly constant.

3.3 Storing and Retrieving Backup Information

Devices for backup must be cost-effective. Storage and retrieval is not an issue, as storage is generally done when users are not on the system, and retrieval is rare. In a typical data processing center, backup information is stored serially to tape drives, and the media stored on-site but offline. This is ideal for backup information.

The fact that tape drives are limited to serial access does not adversely affect backup information, since typical access involves restoring an entire system, device, or (less frequently) an entire file. One should never need to access a single record from one file on a backup tape. Reasonble restoration time is expected, but speed is not a major factor. And since a backup operation is replacing corrupt data, there is no need to worry about having sufficient disk space on which to restore the backup—the backup information is simply stored on top of the corrupt data.

3.4 The Culprit: Storing and Retrieving Archival Information

The culprit in this configuration is the archival information. The ideal device for storing archival information is a cost effective device that offers moderately fast random access to information, and connects just like a system disk (with information stored in the same format as the current information, but with unlimited capacity). Neither fast online (system hard disk) nor slow offline (serial) storage in this common data processing organization provides this combination of attributes. For most installations, the solution is to try to fit a square peg into a round hole by "misusing" one or the other of these two storage types.

In one case, archival information is stored along with the current information, on system hard disks. While fast and convenient for those who need access to the archival information, this is a very expensive solution. Since the body of archival information is constantly increasing, more and more system hard disks will need to be added as time goes on. Eventually the limits of the system or the limits of the budget will be pushed: it's a toss-up which one will be exceeded first.

In the other case, archival information is stored offline, like backup, in a serial medium format. While this solution is highly cost effective as far as storage costs, it is very costly in time and effort when information needs to be retrieved. Random access is not required with backup information, since it is typically an entire device or system that is being copied over corrupt current information. With archival information, however, the user typically needs to extract a single record from a large file.

For example, a university may need to access information on a student who attended many years ago. Extracting a single student record from a file containing perhaps thousands of records on students that graduated that same year would be extremely cumbersome. To retrieve this record from an offline serial device, the entire file containing the record would need to be restored before the record could be retrieved. To do this, a "staging" area must be cleared on the system disk, generally requiring the temporary offloading of current information to make room. After the necessary record is retrieved the archive file can be purged from the disk, and the offloaded current data restored onto the system. This time consuming activity can waste an entire day.

- Fast online storage, or hard disk, is ideal for storing current information.
- Slow offline storage, generally serial tape drive, is ideal for storing backup information.
- Neither fast online system storage nor slow offline serial storage has all of the features necessary for an ideal archival medium. This can present a major problem in any data processing environment.

4 Optical Disk Technology

Optical disk technology is quickly increasing in popularity as users discover all that is has to offer: namely, relatively fast random access to a remarkable amount of data, stored on a compact, removable cartridge. Optical disk devices, which use lasers to store and retrieve information from optical disks, were first developed as an alternative to the Video Cassette Recorder. In 1978, the first optical disk system—the Laser-Disk Video Player—appeared on the consumer market. Since then, optical recording technology has been further developed by the mass storage industry, and split into three distinct branches: CD-ROM (Compact Disk Read Only Memory), WORM (Write Once, Read Many), and Rewritable.

4.1 CD-ROM Technology

CD-ROM (Compact Disk Read Only Memory) disks are not generally useful for most mass storage applications, since they are "Read Only" memories: information can be written to the disk only during the manufacturing process, not by an end-user. A master disk is used to duplicate the information by "stamping" the information onto other disks. CD-ROMs are largely used to distribute and reference large amounts of relatively static data such as on-line encyclopedias, legal citations, and (of course) musical recordings.

4.2 WORM Technology

WORM (Write Once, Read Many) optical disk systems store information on a hard plastic cartridge. Unlike CD-ROMs, information can be written to the disk by the end-user, but only once—the writing process causes permanent alteration of the disk surface. WORM optical disk drives write information using a laser which burns pits into raised portions of a spiral track on the surface of the disk. Once a pit has been created, that area of the disk cannot be restored to its normal flat surface: thus information written to a WORM disk is permanent. Figure 2 shows a vertical cross section of a WORM disk.

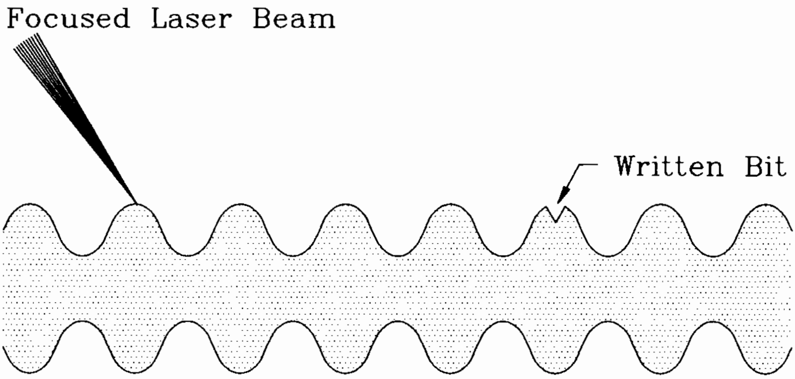


Figure 2: WORM Optical Recording

The same laser beam that is used for writing is used (at a much lower power) to read the information that has been recorded: each pit is interpreted as a digital 1, and each land ("no pit") is interpreted as a digital 0. Pits and lands are identified by the manner in which light is reflected off the surface of the disk.

WORM optical drives have one major "snag" that is not encountered with other mass storage technologies: most existing file systems are structured so that some space on the medium is reserved for a directory. This directory must be updated each time a file is added, edited, or deleted. Since WORM optical disks cannot be rewritten, such directory maintenance is nearly impossible. A number of solutions have been devised to overcome the WORM directory maintenance problem. The successful solutions treat the WORM disk as a serial device when writing, and as a random-access device when reading. These solutions make WORM drives ideal for storing archival information.

Although there are disadvantages imposed by the write-once limitation of WORM drives, they do have one unique advantage. Once data is written, it cannot be altered. This characteristic makes WORM drives excellent for storing information that must be kept for legal and audit considerations.

4.3 Rewritable Optical Technology

Rewritable optical disks are very similar to WORM systems, except that the manner in which information is stored makes it possible to erase and rewrite information. There are several forms of rewritable optical technology, only one of which has to date proven to have commercial value: magneto-optical disk (MOD) technology.

Magneto-optical technology, as the name implies, uses a combination of lasers and magnetic field effects to store and retrieve data. The disk is composed of a magnetic material, highly stable at room temperature, encased in a plastic cartridge. The value of a bit depends upon whether its magnetic orientation is "north-pole-up" (representing a value of 1) or "north-pole-down" (representing a 0). This is illustrated in Figure 3.

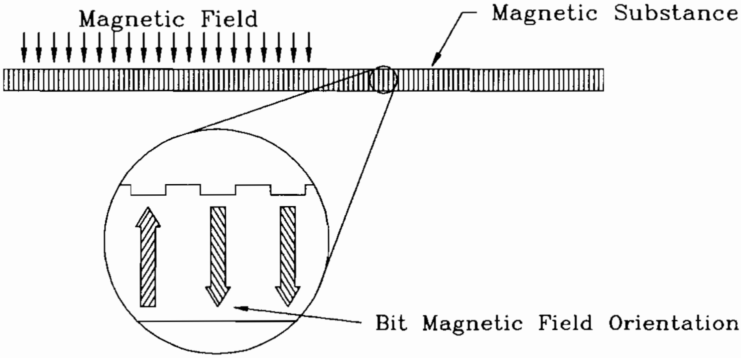


Figure 3: Magneto-Optical Recording

A blank MOD cartridge has all of its bits pointing north-pole-down. A coil in the drive produces a magnetic field that points north-pole-up. The strength of a magnetic field required to change the orientation of a bit varies with temperature: at room temperature, the magnetic coil is too weak to induce such a change. However, at temperatures above 150 degrees Celsius (300 degrees Fahrenheit), the force required to change the magnetic orientation of a bit falls to a manageable level, so bits are easily "flipped" by the magnetic coil.

To write to the disk, a laser heats a spot on the disk to the critical temperature, at which point the magnetic orientation at that spot (representing a bit) can easily be changed by the magnetic field generated by the magnetic coil. After the disk cools—only microseconds later—the magnetic orientation once again becomes nearly impervious to magnetic fields.

To read stored information, an MOD drive directs a low-power laser at the surface of the disk. the light reflected from the surface will rotate in a clockwise or counterclockwise direction depending upon the magnetic orientation of the material at the point of reflection (representing a bit), allowing the data to be interpreted.

4.4 Autochangers

Autochangers, also called jukeboxes, autofeeders, or library systems, use robotics of some sort to swap optical disk cartridges from a large library of disks to one of several online optical drives. Autochangers are used when massive storage capacities and an access time of less than a minute are desired. These systems typically have total storage capacities in the tens to hundreds of gigabytes, with enough internal optical drives to have 5 to 15% of the total capacity mounted and accessible to users at one time.

A home CD player that holds multiple disks and loads one as the previous disk finishes playing is a very simple example of an autochanger. A more pertinent example is Hewlett-Packard's C17*xx* line of Rewritable Optical Disk Library System products. These autochangers come in models that support one, two, or four internal optical drives.

Autochangers, being relatively new to the computing world, are truly useful only when their special needs are taken into account. Existing operating systems have never had to deal with autochangers, and therefore lack sufficient "intelligence" to deal with them effectively on their own. The biggest problem is that existing operating systems have no way of keeping track of what information is stored on which disk in the library—leaving it up to the user to remember where information is stored, and to ask the autochanger to load the correct disk. Specialized tools are needed to keep track of what information is stored where, so that users can simply request a file by name and trust that the autochanger will load and mount the correct disk.

5 Organization of a Data Processing Center Using Optical Technology

Now, look at another possible organization for a Data Processing Center, this time employing optical disk technology:

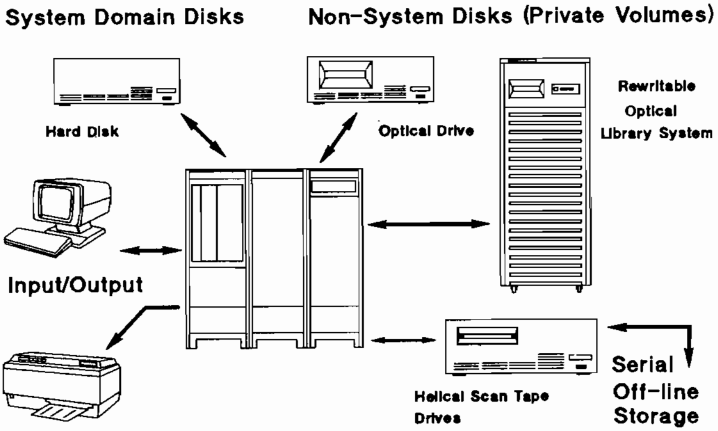


Figure 4: Proposed Data Processing Center

Once implemented, this organization solves the fundamental problem of dealing with archival information.

5.1 Equipment Configuration

As with the previous configuration, the HP 3000 is connected to terminals and printers, and different storage devices. However in the previous example, three types of information (current, backup, and archival) were being stored on two types of storage devices (fast system online and slow serial offline). With the addition of optical technology, a third type of storage device emerges: we'll call this slow online storage. The attributes of each of these types of storage are shown in Table 2.

	Hard Disk	Tape	Optical
Capacity	fixed	unlimited	unlimited
Storage Cost	high	low	moderate
Random Access Speed	fast	very slow	moderate ($\frac{1}{2}$ to $\frac{1}{4}$ the speed of hard disk)
Serial Access Speed	fast	moderate	moderate ($\frac{1}{2}$ to $\frac{1}{4}$ the speed of hard disk)

Table 2: Hard Disk, Tape, and Optical Storage Device Attributes

Optical drives fall in the middle, between fast online system storage and slow offline serial storage, having unlimited capacity and random access capabilities, but somewhat slower performance than hard disks and higher media costs than tape (but much lower media costs than hard disk). It is this third storage medium that offers the ideal solution to the problem of storing archival information.

5.2 Accessing Current Information

As with the earlier configuration example, access to current information is provided with hard disks attached as system disks or non-system volume sets.

5.3 Storing and Retrieving Backup Information

Backup information is still stored on the low-cost serial tape media which is ideal for this purpose. To optimize the backup process and cost effectiveness, reel-to-reel tape drives can be replaced with the new helical scan tape drives. Both 8mm and 4mm tape drives which employ helical scan technology are available, offering much higher capacities and greater reliability than other traditional tape storage mediums.

5.4 The Solution: Storing and Retrieving Archival Information

While it is the storage of archival information that creates problems, it is in precisely this situation that optical technology has its best fit.

Because an MOD is a random-access device, it can be mounted on the computer just like a system hard disk, allowing information to be stored just like on system hard disk. When the information is no longer needed, the optical disk can simply be removed from the drive and stored on a shelf, at an enormously lower cost per megabyte than hard disks. This tremendous cost savings makes the MOD a much better solution than hard disk for archiving information. It is in the retrieval process that the MOD has its edge over serial storage devices. The average amount of time it takes to access data archived on an MOD is less than 30 seconds, for unmounted disks. For an optical disk that is already mounted, access time is in the milliseconds.

Let's look back to our previous example of the university wanting to retrieve a single student record from the archives. When stored on a serial device, the entire database containing the student records from the appropriate year would need to be restored to disk. Restoring the database to disk may involve offloading current information, unless there is sufficient unused disk space to accommodate the file. Once the file is restored, the record can be retrieved—after which the database file can be removed from the system, and the offloaded current information returned to disk. With the MOD, retrieving the record is simple: just load the optical disk containing the appropriate database, and look up the record as if it were stored on the system hard disk as current information.

Once optical technology is introduced into the data processing environment, the solution can be taken one step further with the use of an autochanger, or library system. The use of an autochanger can increase the efficiency of a center by automating access to the information stored on optical disk cartridges. This reduces human intervention, making the process both faster and more reliable.

5.5 Optical Drives in the HP 3000 Environment

There are two basic requirements for attaching optical disk drives (or any device, for that matter) to any computing system: a compatible hardware connection (interface), and a device driver that can communicate with the drive. For WORM drives there is an additional requirement: some sort of solution to the directory maintenance problem described earlier.

In the HP 3000 environment, two types of interfaces are generally available: HP-IB (Hewlett-Packard Interface Bus) and SCSI (Small Computer Systems Interface). Several device drivers are also available. Table 3 below illustrates the current compatibility of MODs, WORM drives, and autochangers with HP 3000 computers.

	WORM Optical		MOD		Autochangers	
Interface:	HP-IB	SCSI	HP-IB	SCSI	HP-IB	SCSI*
Drivers used:	CS80 or tape	none available	CS80	none available	CS80 and tape	no random access

* HP 3000s can access the autochanger as a serial device for backup only, using Turbo-Store on the MPE-XL and MPE/iX systems.

Table 3: Connectivity of Optical Disks on the HP 3000

5.5.1 Attaching WORM Drives

WORM drives are available for HP 3000 computers using the HP-IB interface; although some 3000 computers have access to the SCSI interface, there are currently no device drivers available to support WORM drives. When the HP-IB interface is used to connect a WORM drive, standard HP drivers can be used to access the drive. However, special software utilities are needed to support directory maintenance on these drives.

5.5.2 Attaching MODs

MPE-V systems do not support the SCSI interface, so magneto-optical disk drives on these systems must be attached via HP-IB. Once attached, the device is accessed using HP's CS-80 disk driver. The MOD can then be installed as a Private Volume, Serial Disk, or Foreign Disk.

Under MPE-XL (MPE/iX), MODs can be attached via HP-IB or SCSI. With the HP-IB interface, the disk uses the CS-80 disk driver and can be attached as a Non-System Volume. While HP does support a SCSI interface under MPE-XL, there are currently no device drivers that support magneto-optical drives, though they may be available in the future.

5.5.3 Attaching Autochangers

HP's magneto-optical library systems (HP C17*xx* products) are equipped ONLY with a SCSI interface. This has caused particular difficulty on HP 3000 computers—the Classic (MPE-V) systems do not have SCSI interfaces or drivers; the Precision Architecture (MPE-XL or MPE/iX) systems, though equipped with SCSI interfaces, treat the library system as a strictly serial device, which severely curtails its effectiveness.

IEM's library system controllers address this issue by providing full random access to the library systems through the HP-IB, rather than SCSI, interface. The internal MOD drives in the library system are made to appear as HP 7935 removable disk drives, allowing them to be installed and operated using standard HP drivers and utilities. The library system picker mechanism itself is also made to appear as a standard HP device, allowing programs to be written for the library system using high-level system file access calls, such as `FOPEN`. Application programs can also be written to manipulate cartridges in the library system.

6 Media Management for Autochangers (Library Systems)

The final step in increasing the efficiency of a data processing center is to "fine-tune" the use of autochangers. While autochangers have provided a big step toward total automation of data processing centers, they have only provided a portion of the total solution. Library systems provide a huge storage capacity, and library system controllers provide the means of automatically loading and unloading disks. However, the entire burden of keeping track of where the information is stored is placed on the user. HP offers library systems that provide up to 93 gigabytes of storage on 144 optical disk—a tremendous amount of information for the user to keep track of. Library systems will reach optimum functionality only when a system is provided to keep track of where information resides on the disks in the library system.

So, the missing link is a media management system. Such a system would, ideally:

- Allow the autochanger to be treated as one gigantic disk;
- Manage (hide) all the details of where files are stored;
- Manage (hide) all the details of file retrieval;
- Manage (hide) all the details of file creation;
- Allow existing applications to run without changes;
- Support multiple users (multitasking).

Access would be expected to be somewhat slow when information is stored on an unmounted disk (15+ seconds), due to the need to swap disks in and out of drives, and the slower access speeds of MODs compared to hard disks. Access speeds would be less than one second for a mounted disk.

When the total media management solution is finally devised, the true value of library systems will be realized, and the idea of a totally automated data processing center will become a reality.

Paper Number: 3054

Electronic Draft Capture Interfacing for the HP3000

By Gary L. Biggs

Micro Consulting, Inc.
9708 Skillman Avenue, Suite #107
Dallas, Texas 75243
(214) 340-7794

The wide spread acceptance of credit and debit cards has revolutionized the way business accepts money. Businesses ranging from airlines to zoos, from the sole proprietor to giant corporate entities; now accept MasterCard, Visa, Discover, American Express, and other cards, as payment for merchandise or services. Even taxes can be paid by 'plastic'; and the ready availability of funds through the Automated Teller Machine has added convenience that few Americans thought possible a decade ago. It is only natural that the HP3000 be programmed to handle such cashless transactions. Business benefits from lower discount rates and costs, improved cash flow with the next day availability of funds for bank cards, and the reduced incidence of charge backs due to error or fraud; the consumer benefits from lower prices and improved service in ticketing, retail point-of-sale, and mail order applications.

In choosing an interface method, one is presented with three viable options: RJE, Polled Multi-drop Asynchronous Access, and Dialed Single and Multiple Transaction Access. RJE requires a large investment in specialized hardware, software, and data communications links and is generally suitable only in very large installations with high transaction volumes. Polled Multi-drop Asynchronous requires switched carrier modems which are not directly supported on the HP3000. This type access requires a data communications engine which serves as a buffer to the HP3000 and responds to a poll character issued by the service bureau, bringing up a transmit carrier and transmitting the transaction. Upon completion, carrier is dropped and polling continues. Since this is a shared link, total transaction throughput is dependant on the number of polled terminals on the link and the length of time required to poll each of those terminals. Expensive, specialized hardware is required to make the connection and performance varies with the number of terminals sharing the link. Finally, Dialed Single and Multiple Transaction Access offers the best option for interfacing the HP3000 to automated credit card and electronic draft capture networks. The hardware required consists of relatively inexpensive 'Hayes compatible' modems and an ATP or DTC port off the 3000. Standard, low-cost voice grade phone lines are used thus eliminating the need for dedicated data communications circuits. Network access times are dependent only on the amount of time required to dial a local or 800 WATS number, and a dedicated link is established between the 3000 and the EDC Service Bureau. This paper deals with the problems and methods used in implementing this type of interface.

Virtually everyone is familiar with the Point-of-Sale authorization devices found in most retail stores and restaurants. The clerk takes the charge card, swipes it through a magstripe reader, enters the amount of the sale, and presses a button to request approval of the card. In less than half a minute, a message appears in a window on the data terminal indicating the acceptance or rejection of the credit card transaction. Optionally, these devices will incorporate a receipt printer to give both the customer and merchant a written verification of the transaction. It is this functionality we wish to duplicate on the HP3000.

In general, POS authorization devices communicate with the host authorization computer using what is known as VISA II protocol. VISA II defines a 300 or 1200 baud, even parity message with the following format:

<STX> ...... *(DATA)* .... <ETX><LRC>

where :

<STX> = ASCII Start-of-Text (%2, 2H)
<ETX> = ASCII End-of-Text (%3, 3H)
<LRC> = XOR of all characters between
and including <STX> and <ETX>

<STX> and <ETX> serve as start and stop delimiter characters while the calculated <LRC> serves as a check sum to insure the integrity of the transaction. Since we are dealing with monetary amounts in the data portion of the transaction, some assurance that credit card number, merchant identifier, and transaction amount are correct is very desirable. The immunity to noise corruption is provided by the <LRC>. In the event that the <LRC> received does not match the <LRC> calculated by the recipient, the recipient responds with an ASCII <NAK> character and the message is retransmitted until it is correctly received.

Thus, an exchange between terminal and host looks like this:

```
<STX>MICROCONSULTING<FS>;4012000010000=9909101000000<FS>1234<ETX>2
```

and the return exchange from host to terminal:

```
<STX> CALL CENTER<ETX>*
```

where:

MICROCONSULTING is the Merchant ID;
<FS> = ASCII Field Separator Character (%34,1CH);
;4012.....=9909.... is the data off the Credit Card; and
1234 is the amount $12.34 to be authorized.

It is this message exchange that we must duplicate to implement the VISA II protocol on the HP3000.

One of the most important factors involved in implementing a dial link is the quality of the modem. Over the past 8 years, inexpensive external modems, intended primarily for the home PC market, have proven less durable than modems designed for the commercial market. Undoubtedly, this is a result of the fact that any given modem can be called upon to complete hundreds, if not several thousand, telephone calls in a single day. Modem failures seem to result mainly from EMP (lightening strikes), power supply problems, and trouble with dialing or carrier detection. Generally, modems with option switches by Hayes, U.S. Robotics (Courier), and Multi-Tech have proven to be the easiest to configure and the most reliable.

The HP3000 implementation of the RS232 interface does not provide for any direct control or sensing of the signals sent to or received from a modem. The modem type port requires the carrier detect line to be high prior to any reads or writes to the port, thus it is not possible to send an auto-dial command to a modem port since the carrier is not present when dialing. Additionally, no mechanism is available to the MPE programmer to directly sense whether or not the carrier is present when reading or writing a modem port. Any and all carrier detection must be implemented by exploiting the functionality of the 'Hayes' command set. By configuring a 'Hayes compatible' modem to hold the Data Terminal Ready line high (DTR), it is possible to connect this modem directly to a direct connect type of port on an ATP or DTC. On modems with configuration switches, this option is enabled by changing the position of one of the switches. Modems without configuration switches require a special cable in which a jumper is installed at the modem end to hold the DTR line high at all times. It is for this reason, that modems with configuration switches are easier to install and maintain in this application.

The first step in implementing a VISA II protocol is to open a modem port and issue a dial sequence to connect to the host. In order to increase system performance, more than one authorization device needs to be supported on a system. During times of heavy load, Christmas for example, it is possible that more than a single modem and phone line will be needed to keep up with transaction processing on the 3000. In programming this interface, we chose to pass a table of logical device numbers in a character array (ie. '100 101 102 000') that allows the interface to poll, in round robin fashion, until it finds a port that is not in use. By allowing this multiple device support, it is possible to multi-thread more than one authorization or EDC transaction through the 3000, something that is impossible with a dedicated data communications interface. Adding additional capacity or bypassing a broken modem merely involves changing the table to reflect the new port configuration. If all modems are busy, the interface waits 5 seconds and repolls until a port comes available.

Dialing 'Hayes compatible' modems tends to be somewhat of an art. Since the 'Hayes standard' is, in fact, a defacto standard, no real published specifications exist for this command set; so manufacturers are free to implement the command set as they see fit. This has been the primary source of modem incompatibility found over the past several years. Currently, the following command string sent to the modem offers the fastest and most consistent dialing performance:

```
ATQ0V0E0X4S6=2S12=10S9=1S2=2M0DT#######
```

where:

Q0	=Return Result Code
V0	=Numeric Result Codes
E0	=Echo Characters OFF
X4	=Extended Result Code Set
S6=2	=2 second wait for Dial Tone
S12=10	=Set escape code guard time to 1/5 second
S9=1	=Set Carrier Recognition time to 1/10 second
S2=2	=Set Escape Char to <STX>
M0	=Modem Speaker OFF
DT	=Dial using Touch Tone
#######	=7 digit phone number

By incorporating the modem setup parameters in all dial requests, we insure that the modem is initialized prior to each dial. Unfortunately, this command string requires 39 characters and the 'Hayes' command structure limits commands to a maximum of 40. To dial 800 WATS numbers or access codes thru a PBX (9+ dialing), the interface breaks the dial sequence into two commands which are sent as needed.

Upon reception of the dial string, the modem goes off hook and dials the requested number. In return, the modem issues a single status character which represents the success or failure of the call. If a carrier is detected, a modem status of CONNECTED (1 or 5) is returned and the call is considered successful. The real differences in modem compatibility are found in error detection and handling. Many recently designed modems are capable of detecting dial tones and busy signals which can greatly speed dialing and error processing. These modems often will return extended status codes which are specific for that particular brand of modem. In addition, it is necessary to time-out all reads for modem status, so that in the event of modem or phone line failure, the user is protected from waiting indefinitely for a status return that will never occur. Flexibility in handling these types of errors is essential for the fastest call processing and interface performance.

With carrier up, we now post another read to the modem port, this time terminated via FCONTROL #41 to the ASCII <ENQ> (%5, 5H) character. <ENQ> is the VISA II signal that the host is ready to receive the authorization transaction. Within 10 seconds of detecting carrier, the host should transmit the <ENQ> and we respond by transmitting the outbound message. If the 3000 fails to respond to the <ENQ> within 2 seconds, 2 additional <ENQ> characters are sent at 2 second intervals to attempt to synchronize the transmission. Conversely, if no <ENQ> is received during this period, our interface needs to assume the network is down and proceed with alternate processing.

VISA II specifies the communication protocol without detailing the content of the data area of the transaction. Always present are the merchant account number with the service bureau, the credit number and expiration date, the amount of the transaction, and a transaction code that details what the transaction should do. Debits, credits and voids are indicated by the transaction code. Optionally, some transaction types will require additional fields; debit cards need PIN numbers, transactions which received voice approval must be submitted with the approval code obtained when the clerk called the manual authorization center. The format and content of the data area varies from provider to provider; thus, switching providers often results in the necessity of changing the routines to format the outgoing transaction.

At the completion of transaction formatting, the <LRC> for the outgoing transaction must be calculated. Implemented as a small subroutine, the calculation requires that all characters in the protocol block be XORed. The result is a additional character to be appended to the end of the outgoing transaction. A write is posted to the modem port followed by a read with a 30 second time-out. At this point, the host will respond in one of two ways. Either the message will be accepted as correct, the <LRC> we calculated matches the <LRC> calculated by the host, or the message will be rejected. These two conditions can be handled by specifying an <ETX> as the primary line termination character (FCONTROL 41) and a <NAK> (%25, 19H) as the secondary line termination character (FCONTROL 25). Receipt of the <NAK> is indicated by a condition code of Less Than (<) from the FREAD intrinsic, and the outgoing transaction must be retransmitted.

A successful FREAD, as indicated by a condition code of EQUAL TO (=) , results in the reception of the resultant incoming message from the host, minus its terminating <ETX>. In the event that the 30 second timer expires, one must assume that either the network is down or carrier has dropped on the modem. In either case, alternate processing is required to notify the user to resubmit the transaction.

The trick, now, is to catch the <LRC> transmitted by the host. On MPE /ix, the programmer merely needs to enable the type-ahead buffer using the FDEVICECONTROL intrinsic prior to posting the read for the inbound transaction. Posting an additional 1 character read immediately after the successful reception of the inbound message will retrieve this character. It is then a simple matter to compare the received <LRC> with a calculated <LRC>; if they do not match, a <NAK> is transmitted to the host, and a retransmission will occur. Up to 3 retransmissions may be requested in an attempt to compensate for noise or other data corruption. MPE/V machines require a different tactic. Reception of the <LRC> character on these machines is a hit or miss proposition. The <LRC> is transmitted within a few character times of the terminating <ETX>. At speeds greater than 300 baud, the 3000 can not reliably post another read prior to the transmission of this character, thus, it is invariably missed. To compensate, we need to <NAK> the first inbound transmission every time. This will cause the host to retransmit, and the two transmissions can be compared for accuracy. If they match, the reception is good and processing can continue; otherwise we must continue <NAK>ing until 2 transmissions in a row match.

Finally, we must acknowledge the host that we correctly received the inbound transaction. By transmitting an <ACK> (%6,6H), we notify the host of our correct reception. It is at this point that the host considers the transaction complete and drops carrier. If the host does not receive the <ACK>, the transaction is considered incomplete and will be rolled back by the host, thereby preventing any monetary transaction from being posted.

The format of the inbound messages varies from host to host. What follows is a representative sample of some responses:

`APPROVED 32 – REF 12345678`	MC/V with EDC
`ACCEPTED – REF 98765432`	Credit with EDC
`GOOD 9876`	MC/V approval only
`DECLINED`	Over Limit
`CALL POLICE`	
`DESTROY CARD`	

Electronic draft capture messages return a unique Reference Number that identifies that individual transaction. Approval codes indicate that the card is valid for the amount of a transaction. In the event that a merchant is enabled for EDC and obtains an approval code with no reference number, the transaction must be resubmitted at a later time as a POSTAUTH transaction to perform the electronic monetary transfers. Some hosts require that all transactions resubmitted in a batch protocol perform the settlement function. In any case, the message formats are defined by the host and can be parsed by the interface to extract the approval codes and reference numbers. These should be permanently recorded as an integral part of the payment, for later reference and reconciliation.

To this point, each and every transaction requires a new phone call for processing, and there are some hosts that can handle only one transaction per call. Each transaction requires, in very general terms, 4 to 10 seconds to place a local phone call, one second for carrier detection and synchronization, one second for outbound message transmission at 1200 baud, 8 to 20 seconds for processing by the host and an additional two seconds for message reception and acknowledgement; thus typical transaction times for single, dialed transactions range from 16 to 28 seconds, depending on the load at the host. Maximum throughput in this environment can only be expected to be two transactions per minute per modem. Fortunately, many EDC vendors have developed protocols that allow the submission of more than one transaction per call. Batch type submissions are often suitable in mail order and other fulfillment applications where the customer is not always present. Late night processing often reduces transaction times and increases network reliability, especially during the heavy Christmas retail shopping season.

The batch mode capabilities of some EDC providers can be exploited to yield an on-line transaction processing system of much higher throughput than the single transaction per dial interface. As a part of the data portion of the outgoing transaction, various function code bits are often used to indicate whether the card number was read from a magstripe or manually input, if the transaction is a debit, a credit or a void, and whether or not more than one transaction is expected per phone call. When more than one transaction is specified, many vendors will continue to hold the carrier for a reasonable period of time. If the HP3000 can generate an additional request within, let's say, a minute, the modem carrier will be up and the host will be ready to receive and additional transaction. Often with a sufficient volume of transactions, a dial interface can meet and often exceed the response times of dedicated data links. Transaction rates of up to 12 per minute have been recorded as the host preferentially services those inbound lines with the highest volumes.

The lack of hardware carrier checking is a serious problem in implementing a batch mode protocol in which transactions come at irregular intervals. Without checking for carrier prior to each transaction, there is no way to determine whether the host has dropped the carrier due to inactivity. Fortunately, the 'Hayes' modem offers a reasonable solution that can be achieved in software. In the original modem setup string transmitted during the dial sequence, the interface sets the modem escape character to <STX> character. If the carrier is up, and three <STX> characters are received in rapid succession, the modem will go off-line without dropping the carrier, and enter the command state. At the same time, it issues a modem status of OK (0) which can be caught by issuing a read to the modem port immediately after transmitting the three <STX> characters. If the read does not complete within one second, the modem was off-line without a carrier, and a dial sequence should immediately be issued in order to reestablish the link. When the modem does respond with the 0 status, the carrier is present and the modem command of ATO will place the modem back in the on-line state. By selecting <STX> as the modem escape character, the host begins a new transaction which must now be completed. Thus:

<STX> <STX> <STX> <ETX> ?

will be <NAK>ed by the host as a bad message; ? is a bad <LRC> for this sequence of characters as a VISA II message. Having determined that the carrier is active and the host is responding, the interface can proceed to transmit the authorization request without resetting and redialing. This whole process of determining the presence of carrier and host activity takes little more than a second.

Implementation of the VISA II protocol for the HP3000 was undertaken in SPL for MPE/V and later translated to HP C/XL for native mode applications. Because of the differences between various hosts, each protocol requires a customized formatting routine for the outgoing message and a custom parsing routine to retrieve the authorization codes and reference numbers for successful transactions. Additionally, several protocols require log-ons to COMPUSERVE, GENIE, or BTTYMNET to serve as transparent pipelines to the authorization host. About 80% of the code required for logging, dialing, and transmissions is common to all protocols, so minimal time is required to program new protocols. The result is a single subroutine, AUTO_AUTHORIZE, which can be easily changed to use new protocols by relinking. Since the choice of EDC vendor is often influenced by the merchant's banking relationships and discount rate negotiations, the ability to switch protocols at minimal cost insures the long-term success and desirability of this type of interface.

Directly interfacing HP3000 with Electronic Deposit Capture Networks, greatly enhances the usefulness of the HP3000 in many applications. Innovative programming, an understanding of the RS232 interface, and careful error handling insure the timely and accurate handling of credit and debit card transactions. The improvement in cash flow and customer service more than justifies the inclusion of this functionality in any ticketing, mail order, and point-of-sale application.

#3055: Popular Mass Storage: Optical Disk and Helical Scan Tape

Husni Sayed
IEM, Inc.
P.O. Box 1889
Fort Collins, CO 80522
(303) 221-3005

1 Introduction

Optical disk drives and Helical-scan tape drives are relatively recent advances in the mass storage arena. These two technologies, each with its own specific advantages, are transforming mass storage technology.

Helical-scan tape is very inexpensive, and has a large enough capacity to allow unattended backup of on-line systems. It would take almost 13 reels of high density (6250 bpi) 9-track tape, or 12 IBM 3284 cartridges, to store as much information as a single 8mm tape cartridge. Furthermore, traditional tape systems would require the presence of an operator to change every one of those reels or cartridges: a process that may take hours. This additional labor cost makes helical-scan tape even more attractive. Decreased storage space is another incentive that favors helical-scan tape.

Optical disks have many of the advantages of helical-scan tape, such as large capacities in small volume, with faster access times. They are serving to fill a previously empty niche that existed between large capacity, low cost mass storage tape systems, and fast access, high cost Winchester disks. The removable nature of optical disks makes them ideal for storing large software systems and data bases that can be swapped in and out as needed.

2 Optical Disk Technology

Optical recording devices, which use lasers to store and retrieve data from optical disks, were first developed as an alternative to the Video Cassette Recorder. In 1978, the first optical disk system — the Laser-Disk Video Player — appeared on the consumer market. This read-only device used a 12″ platter and a laser read-head to play back digitally encoded video signals.

Since then, optical recording technology has been further developed by the mass storage industry, and split into three distinct branches: CD-ROM (Compact Disk-Read Only Memory), WORM (Write Once, Read Many) and Rewritable (Erasable).

The main expense involved in optical disk systems is the read/write head, which uses lasers, beam-splitters, lenses, and mirrors to access data. For this reason, most optical systems are single-sided: the disk must be removed from the drive and manually turned over to access the other side of the disk cartridge. An optical disk system that could access both sides of a disk cartridge without removing the disk would require two read/write heads, effectively doubling the cost of the drive.

2.1 Optical Disk Advantages

Optical disks have numerous advantages over magnetic media. First, since the density of an optical disk cartridge is limited only by the wavelength of light used by the laser writing the information, optical disks are capable of tremendous track capacities. Most current systems use a near-infrared laser with a wavelength of 8,000 – 10,000 angstroms, resulting in track densities on the order of 16,000 tpi. A single $5\frac{1}{4}''$ optical disk cartridge with such a track density can hold roughly 800 MBytes of data. Seco.d, the distance between the head and the surface of the disk is much greater than that used by traditional Winchester technologies. This increased separation between the disk and the head makes head crashes very rare. Finally, the optical disk cartridge itself is very durable. Encased in plastic, it is immune to fingerprints and resistant to heat and humidity. These factors combine to offer an archival life of more than 10 years, in a disk that is removable and easily transported from one machine to another.

However, optical disk drives have slower access times than Winchester disk systems (50–150 ms as opposed to 10–20 ms). This is because the head in an optical disk drive weighs much more than the head in a Winchester. Data transfer times, which are dependent solely upon rotational speed, are comparable between the two systems. However, in applications where speed is crucial, the optical disk drive may well be outperformed by magnetic hard disk.

2.2 CD-ROM (Compact Disk-Read Only Memory)

2.2.1 Characteristics

CD-ROM (Compact Disk Read Only Memory) disks are high capacity "Read Only" memories: information can be written to the disk only during the manufacturing process, not by an end-user. This characteristic limits their usefulness as a typical mass storage medium. The disks themselves are thin, flat disks made of polycarbonate covered by a thin reflective layer. Each disk is roughly 4.75 inches in diameter, and just over a millimeter thick. The medium used for computer applications is physically the same as that used for the music industry, so anyone who has been in a music store has seen a CD-ROM disk.

Information is stored on only a single side of the disk, with capacities of more than 500 MBytes per disk.

2.2.2 Implementation

The surface of a CD-ROM disk has a continuous spiral track, like a phonograph record, but with a much higher density. A "pit" on the raised portion of the track represents a digital 1, and a flat area (or "land") represents a digital 0. A master disk is used to duplicate the information by "stamping" the information onto other disks.

The stored data is read using a low-power laser: pits and lands on the surface of the disk reflect light differently. This difference in reflective quality is detected, and translated into readable data.

2.2.3 Applications

Because the process of preparing data and creating a master disk is so expensive, CD-ROM is not economical unless a great many copies are going to be made. For this reason, CD-ROMs are commonly used to distribute and reference large amounts of relatively static data such as on-line encyclopedias, legal citations, and (of course) musical recordings.

2.3 WORM (Write Once, Read Many)

2.3.1 Characteristics

WORM (Write Once, Read Many) optical disk drives are more useful as a mass storage medium for the typical end-user. Unlike CD-ROMs, information can be written to the disk by the end-user—however, information can only be written once, as the writing process causes permanent alteration of the disk surface.

WORM Optical disk cartridges typically come in sizes of 5.25 inches and 12 inches, with the 5.25-inch size more popular. A WORM optical cartridge, unlike a CD-ROM disk, is actually encased in a plastic cartridge. Also in contrast to CD-ROM disks, information can be written to both sides of the disk, though the disk must be physically turned over to access information on the second side. Capacities on a WORM disk are typically 600 to 800 MBytes (300–400 MBytes per side).

2.3.2 Implementation

Like CD-ROM disks, WORM optical disk drives write information using a laser which burns pits into raised portions of a spiral track on the surface of the disk. Once a pit has been created, that area of the disk cannot be restored to its normal flat surface: thus information written to a WORM disk is permanent. Figure 1 shows a vertical cross section (along tracks and sectors) of a WORM disk.

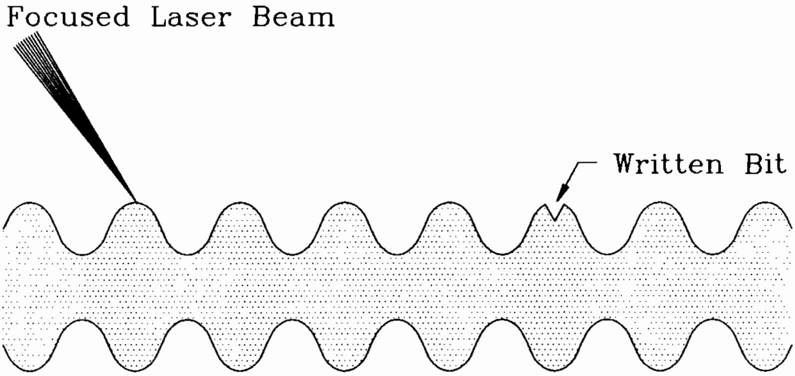


Figure 1: **WORM Optical Recording**

To read data the same laser is directed at the surface, but at a much lower power setting. The laser is reflected off the surface, and this reflected light is gathered into a photocell. The light reflected by a pit is easily distinguished from light reflected by a flat surface: this difference in light reflections is used to read bit patterns. Each pit is interpreted as a digital 1, and each land ("no pit") is read as a digital 0.

2.3.3 Applications

WORM optical drives have one major "snag" that is not encountered with other mass storage technologies. Most existing file systems are structured so that some space on the disk is reserved for a directory. This directory must be updated each time a file is added, edited, or deleted. Since WORM optical disks cannot be rewritten, such directory maintenance is impossible. One solution to this problem is to employ special software drivers that use an entirely different file structure involving linked directories. Other solutions involve using a flexible disk to store the directory entries, while the actual data is stored on the optical disk. Directory maintenance limitations, combined with the write-once nature of the media, make WORM optical disk drives useful primarily for backup and archival tasks.

The write-once "limitation" of the medium, however, gives the WORM optical disk drive one unique advantage: once data is written, it cannot be altered. This characteristic makes WORM drives excellent for storing information that must be maintained for legal and audit considerations.

2.4 Rewritable

2.4.1 Characteristics

Rewritable (also referred to as "Erasable") optical disks have been introduced as a viable mass storage technology relatively recently. Unlike WORM disks, information stored on a Rewritable Optical Disk can be erased and re-written. This increased flexibility makes them a more attractive option for many users. The typical erasable optical disk is 5.25 inches in diameter and encased in plastic, like a WORM disk. Capacities are similar to those of a WORM disk—600 to 800 MBytes (300–400 MBytes per side).

There are several forms of rewritable optical technology, but only one to date has proven to have commercial value: magneto-optical disk (MOD) technology.

2.4.2 Implementation: Magneto-Optical

Magneto-optical technology, as the name implies, uses a combination of lasers and magnetic field effects to store and retrieve data. The disk is composed of a magnetic material, highly stable at room temperature, encased in a plastic cartridge. The value of a bit depends upon whether its magnetic orientation is "north-pole-up" (representing a value of 1) or "north-pole-down" (representing a 0). This is illustrated in Figure 2.

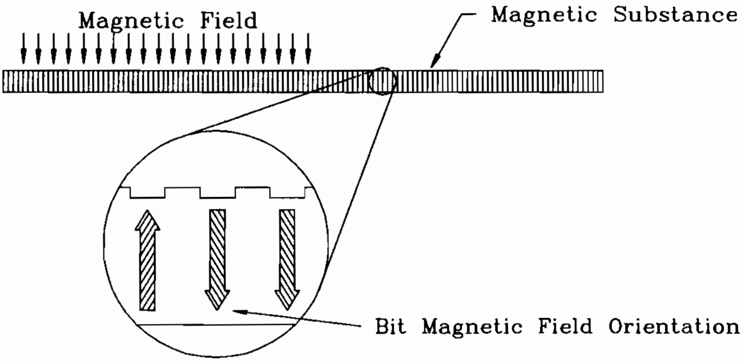


Figure 2: **Magneto-Optical Recording**

A blank Magneto-Optical Disk (MOD) cartridge has all of its bits pointing north-pole-down. A coil in the drive produces a magnetic field that points north-pole-up. The strength of a magnetic field required to change the orientation of a bit varies with temperature: at room temperature, the magnetic coil is too weak to induce such a change. However, at temperatures above 150 degrees Celsius (300 degrees Fahrenheit), the force required to change the magnetic orientation of a bit falls to almost zero, so bits are easily "flipped" by the magnetic coil. To write to the disk, a laser heats a spot on the disk to above 150 degrees Celsius, at which point the magnetic orientation at that spot (representing a bit) can easily be changed by the magnetic field generated by the coil. After the disk cools—only microseconds later—the magnetic orientation once again becomes nearly impervious to magnetic fields.

The properties of the Kerr effect are used to read data stored on an MOD cartridge. The Kerr effect states that light will rotate in a particular direction if influenced by a magnetic field. An MOD drive uses this effect by directing a low-power laser at the surface of the disk. The light reflected from the surface will rotate in a clockwise or counterclockwise direction, depending upon the magnetic orientation of the material at the point of reflection. The read head detects the rotation direction, and sends a corresponding value of "0" or"1" to the computer.

One disadvantage of MOD cartridges stems from the fact that MOD systems must write zeros to the surface before data may be written to that spot. This means that the disk must rotate twice to complete a write operation: once to write zeros, and once to write the desired information. This requirement effectively increases the write-access time of an MOD drive by 40% over read access times.

2.4.3 Applications

Rewritable optical disk systems have roughly the same capabilities as WORM disks; capacities, access times, and media life are comparable. Therefore, they can be used just as effectively in most of the same applications as WORM drives, except for audit-trail (which requires the unalterable nature of WORM media). The rewritable nature of these drives, however, makes them much more flexible. Since a rewritable-optical disk drive "looks" like a Winchester hard disk, files can be stored and accessed without using a special archival format. Access to these files is extremely easy; the disk need only be inserted into the drive and mounted.

With rewritable technology, disks containing obsolete backups and archives can be reused, standard system software can be used with read/write access, and file changes can be stored much more efficiently. Since the disks are rewritable, they may be used in other applications for which a WORM disk would be unsuitable. For such applications, the rewritable-optical system is a replacement or supplement to on-line storage such as a Winchester hard disk or flexible disk drive.

2.5 Autochangers

Autochangers, often referred to as "jukeboxes", offer users automated access to a vast amount of information. Autochanger systems typically contain one or more optical disk drives, a storage area for a number of optical disks, and a mechanical arm used to select and load the disks that are stored in the jukebox.

Most jukebox systems use at least two optical drives, to increase the speed at which cartridges can be changed. In a system containing only one drive, a new cartridge cannot be selected and accessed until after the current cartridge is removed from the drive and stored in its proper location. In a system with two drives, a new cartridge can be loaded into drive "B" and access can begin immediately. While the new cartridge is in use, the old cartridge can be removed from drive "A" and stored in its proper location.

Autochanger systems usually have enough internal optical drives to have 5%–15% of their total capacity on-line at any one time.

3 Helical Scan Recording

Helical-scan tape stores data using technology that was originally developed by the video recorder and digital audio tape industry. The name is derived from the method by which the tape travels over the head. Traditional tape technologies use a fixed head, with the tape passing (relatively slowly) over the head. Helical scan, however, uses a head that is mounted on a rapidly spinning drum aligned diagonally to the track. As the tape passes over the drum, the head writes tracks of data in a diagonal pattern corresponding to the pitch of the head. This method produces track densities on the order of 1,000–2,000 tracks per inch.

In contrast to traditional computer tape devices, which record longitudinally (along the length of the tape), helical scan tape drives record in diagonal tracks across the tape. This diagonal recording allows data to be recorded more densely and read with greater accuracy. It also results in gentler tape handling, because the higher recording density permits lower tape speeds for the same data transfer rates.

Although helical-scan drives used in the computer industry are very similar to a VCR (Video Cassette Recorder) or a DAT (Digital Audio Tape) player, they require a much higher reliability. VCRs and DAT players typically have an error rate of 1 in 10^6. When an error occurs during a VCR recording, a small extraneous spot may appear on the screen. Similarly, in the case of DATs, a timeout may occur for less than a millisecond. These types of errors are virtually undetectable to human senses. An error rate of 1 in 10^6, however, is unacceptable for mass storage applications. Because of this, error checking and redundancy must be implemented to produce an error rate more in the neighborhood of 1 in 10^{13}.

3.1 Helical Scan Advantages

Helical scan tape technology has a number of advantages over traditional tape media. Tapes are very compact (smaller than a deck of playing cards), and offer extremely high capacities—1.3 GBytes for 4mm tapes, and 2.5–5 GBytes for 8mm tapes. The use of data compression can increase these capacities up to fourfold. The media itself is relatively inexpensive, resulting in a cost per MByte of storage well below one cent.

Helical scan tapes also have a longer archival life (10 years, according to the United States National Bureau of Standards) than magnetic tape, which typically needs to be replaced at least every 3 years. This increases data integrity, saves money by requiring less frequent replacement of media, and saves time and personnel in the maintenance of archived information.

3.2 8mm video tape

3.2.1 Characteristics

8mm helical-scan tape systems were derived from commercial Camcorder technology. With a maximum storage capacity of 5.0 GBytes per tape (without data compression), 8mm media has the single highest storage capacity-to-volume ratio of any mass storage device currently in use. Access time, as with all tape systems, is relatively slow: in the tens of seconds. Burst transfer rates are on the order of 10 MBytes per minute.

The media used for 8mm tape in the mass storage industries is the same lightweight, plastic cartridge used in the entertainment field. Any high quality metal tape from a Camcorder can be used in an 8mm helical-scan tape drive (although data grade tapes are recommended). The 6″ by 4″ by $\frac{1}{2}$″ cartridge fits easily into a shirt pocket. The volume of sales for the entertainment industry has drastically lowered the price of these cartridges to under $10.00 each. This factor, plus the huge storage capacity of these tapes, has driven the cost of storage to less than one penny per MByte.

3.2.2 Implementation

On 8mm tape drives, three heads (two read/write heads, and a servo head to position the tape) are mounted on the rotating drum, which is tilted at a 5° angle from vertical. The drive also has a stationary erase head positioned 1.56 inches from the read/write position in the tape path. In each rotation, one track is written, and the second head read/write head provides a read-after-write check.

To eliminate crosstalk, fully erased guard bands are left between tracks: this protects against reading data from adjacent tracks, and reduces the need for exacting tape positioning. During write operations the erase head is always active, "erasing" the portion of tape about to be written. This ensures that there are no residual servo or data signals from previous writes. This is shown in Figure 3.

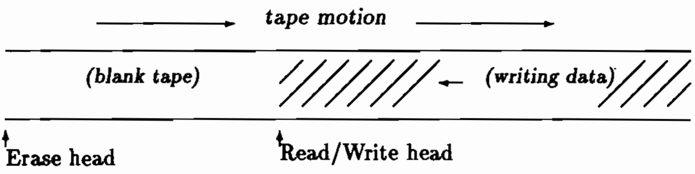


Figure 3: **8mm Tape Writing Procedure**

3.3 4mm DAT

3.3.1 Characteristics

DATs, or Digital Audio Tapes, are just now being marketed in the computer industry. They are very similar to 8mm tapes, the main difference being that the medium is 4mm wide instead of 8mm (resulting in a maximum capacity that is half that of 8mm tapes). In addition to being half as thick, DATs are also smaller than 8mm tapes (about 3" by 2").

3.3.2 Implementation

The audio version of DAT has two combination read/write heads, where DAT drives for computer applications use two read heads and two write heads (for a total of four heads). DAT drives with only two heads do not have a read-after-write capability, so data reliability is reduced.

On four-head DAT drives, two diametrically opposite write heads are mounted on the rotating drum, which is tilted at a 6° angle from vertical. In each rotation, each head writes one track at a different angle from the other head, resulting in two tracks written per rotation. Since the heads are wider than the tracks, the tracks overlap each other slightly, with no guard bands between them.

To eliminate crosstalk, each write head is set at a slightly different angle—±20° from the *azimuth* (the axis around which the drum rotates)—so data in adjacent tracks is written at different angles. Each read head reads only the tracks which were written at the same angle as itself, and uses the weaker signals from surrounding tracks to center itself. These read heads are mounted on the drum at 90° angles from the write heads, and provide a read-after-write check.

Because DAT drives were originally developed for audio recording, a special format designed to "overlay" the basic audio format is implemented to allow DAT drives to be used for computer applications. There are currently two competing formats—DDS (Digital Data Storage) and Data/DAT.

The DDS format, originally developed by Hewlett-Packard and Sony, is a "traditional" tape format: files are stored sequentially on the tape, and stored files cannot be modified or updated. The Data/DAT format supports both sequential and random access modes. However, tape drives are by nature sequential devices—using Data/DAT formatted drives in random-access mode requires excessive tape preformatting (which takes about 2 hours). This pre-formatting not only takes time, but wastes a significant amount of tape capacity.

3.4 Autofeeders

As with optical disk drives, systems that handle multiple tapes are also available. These autofeeders differ somewhat from optical disk autochangers, because of the way in which they are typically used. An optical disk autochanger is generally used for random access to on-line storage. Since speed is important in this applications, optical disk autochangers often contain two drives. Helical scan tape drives, however, are not used for on-line storage, and do not offer random access to files.

Since their main purpose is for backup, autofeeders were designed to facilitate the automated backup of systems that are too large to fit on a single tape. These systems typically contain a single drive, and a mechanical arm which is used to sequentially load and unload tapes. The operative word here sequential: tapes cannot be randomly loaded and unloaded; rather, they are loaded into the drive in the order in which they were placed in the autofeeder storage area. As one tape is filled during backup operations, the autofeeder will automatically load the next tape and continue the backup operation.

3.5 Helical Scan Applications

High capacity helical scan tape drives are becoming the solution for backup and archival operations. Their extremely high capacities make it possible to perform unattended overnight backup of systems too large for other mediums. With Autofeeders, systems up to 200 GBytes can be backed up without operator intervention.

Helical scan tapes offer another advantage for archival—a shelf life of 10 years, compared to 3 years for other types of magnetic tape media. Due to the method in which information is stored on helical scan tapes, information is not as susceptible to bit migration and print-through. No other medium offers as high a capacity at as low a cost as helical scan tape.

3.6 Summary Comparison of Helical Scan Technologies

	2GB 8mm	5GB 8mm	4mm DAT
Recording Density (KB per inch)	45	45	61
Track Density (Tracks per inch)	819	1638	1869
Areal Density (MB per inch^2)	35	74	114
Single Tape Capacity (GBytes)	2.5	5.0	2.0
Read/Write Speed (KB per second)	246	500	183
File Search Speed (MB per second)	2.4	37.5	36.0
ECC Overhead	28%	28%	29%
Correction Block (Bytes)	1024	1024	5756

4 Conclusion

As optical and helical scan systems are refined, their cost, storage capacities, reliability, and speed will all improve. With helical scan technology, capacities will continue to go up. In the optical arena, products that incorporate a combination of optical technologies are looming on the horizon. Whatever the future holds, users can be sure that these technologies will continue to offer new solutions to old problems, and gain widespread acceptance as their advantages become more apparent.

4.1 REFERENCES:

Freese, Robert P.
"*Optical Disks Become Erasable*"
IEEE Spectrum, February 1988.

Levine, Ron
"*Optical Storage Comes of Age*"
DEC Professional, November 1988.

Parker, Chas
"*Digital Audio Tape*"
HP WORLD, April 1989.

Small, Steven F.
"*8mm and DAT Formats Compete for Tape Technology of Choice*"
Computer Technology Review, Spring 1990.

Vermeulen, Bert
"*Helical Scan and DAT—a Revolution in Computer Tape Technology*"
Hewlett-Packard Company.