INTEREX

HP BUSINESS USERS
CONFERENCE PROCEEDINGS
VOLUME 1

LAS VEGAS

SEPTEMBER 20-25, 1987

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

INTEREX

the International Association of
Hewlett-Packard Computer Users

Proceedings

of the

1987 North American Conference
Of Hewlett-Packard Business
Computer Users

at

Las Vegas, Nevada
September 20-25, 1987

VOLUME 1

Index By Author

PETER R. ACKERMANN, Attorney-at-Law,
ORBIT INTERNATIONAL Ltd., London

THE LEGAL PROTECTION OF SOFTWARE IN THE US AND INTERNATIONAL LAW

1. Introduction

Ever since software had become a marketable commodity in its own right and consequently was copied by others for profit, the protection of software has been an important issue. The last decennium has seen new specific copyright protection laws in at least ten countries, including the US, Japan and Germany. With the growing development of more and more sophisticated software in virtually every country and its mounting importance vis-a-vis hardware, no national legal system can allow itself to abstain from studying the question whether and how computer software should be protected.

The UNESCO and the World Intellectual Property Organization (WIPO) have jointly considered all aspects of the protection of computer software at national and international levels in order to issue guidelines to national legislators who have yet to implement rules for this new field. WIPO has in fact in 1984 established the first technical classification of computer programs.

In the United States, a Congressional Commission on New Technological Uses of Copyrighted Works (CONTU, 1976) laid the groundwork and subsequent developments have clearly established that software products qualify for protection under several aspects of intellectual property law. Nevertheless, under current law, while there are few doubts about whether or not software receives legal protection, the nature and scope of that protection is uncertain and depends on the source of law from which the software proprietor derives its rights. Clarification of this requires consideration of laws relating to copyright, patent and trade secrets.

Most users of computer software have signed comprehensive licence contracts, containing non-disclosure clauses and regulating every aspect of their relation to the licensor. The drafting of software contracts is a paper for itself. We are however not touching upon contractual relationships but are solely concerned with the situation between parties who have not signed their names to one paper.

2. Definition of "Computer Software"

What exactly is the object of legislation and jurisprudence, enacted and handed down for the purpose of our very protection ? International consensus on a definition has not been reached but a brief definition is given in sect. 101 of the 1980 US Computer Software Copyright Act, which describes a computer program as

> "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result".

This definition however does not result in an easy deduction, that everything falling within these terms will enjoy protection under the Act.

3. Mode of protection

There are no less than eight different forms of protection that one can distinguish in the various countries that have already dealt with the subject matter:

- patents
- copyright
- trade secrets/confidentiality rules
- criminal law
- catalogue protection
- civil law
- protection against "slavish imitation"
- protection against "enrichment without cause"

3.1 At first sight, patents are often considered to be the natural way of protection for computer software, since patents and computers both are of a very technical nature. However, in most countries (e.g. West Germany, France, Italy and in the European Patent Act) the patentability of software is totally excluded. Several factors are responsible for this situation. Most countries limit patent protection to processes which are both novel and inventive. On one hand however, computer programs are not intended to solve problems in a novel way but computerize processes which more or less existed before. On the other hand, patent law is traditionally limited to processes of a physical nature and to physical products. Many computer programs concern methods of organization and administration which do not bring about any physical changes.

Under relatively severe restrictions, software can be patented in the US. The restrictions, defined in a host of

court cases, revolve around the underlying concept that a process may be patentable, natural laws are not. The Supreme Court held in Diamond v. Diehr (450 US 175 -1981) that if a patent claim seeks to preclude the general use of a mathematical equation, it is not patentable. If however the claimants seek only to foreclose from others the use of that equation in conjunction with all of the other steps in their claimed process, a patent may be granted. Consequently, the discovery of a natural law or an algorithm does not convey a right to exclusive use thereof divorced of particular useful applications. It has repeatedly been held that "patent claims that seek to preempt the natural laws on which science and technology are built are not patentable since they inhibit rather than promote development of science and technology".

Japan, Holland and the UK also accept the patentability of software on the condition that it must serve a material process, or that an appartus to which the program has been applied has become novel by it. Apparently, the possibility of industrial application is the key to a patent as may be the case with software-driven robots.

3.2 Copyright is the main form of protection for computer software. It is admitted in most countries. The US, the UK, West Germany and several others have already enacted specific computer copyright acts and legislation to that effect is pending in various others.

All countries require that works, in order to be protected by copyright, must meet a certain standard with is usually called "originality". Software must be an "individual creation of the mind", it must have a certain uniqueness, i.e.it must be expected that another programmer performing the same programming task would inevitably come to a different result. That of course is bland theory, because in any court proceeding, this "inevitability" will be hard to establish.

Protection under copyright is not only given for the program itself but also for the steps leading up to it. Program design and flow charts are part of the copyright, underlying algorithms again are not. (Exception: in the UK, where algorithms seem to be protectable under copyright provided that they have been laid down in material form.) Protectable is source code and object code, applications and operations software. Chips qualify for copyright protection in the UK, because the masks which are used are considered drawings, engravings or photographs. The US have enacted special legislation for the protection of chips which has a main feature of interest: If the chips are

produced in a country that does not meet the reciprocity demands of the american legislation and are then imported, they do not enjoy protection under this particular act.

Most countries do not prescribe any formalities for copyright protection; exceptions are the US and Spain. In fact, the Berne Convention on Copyright does not permit the requiring of formalities with respect to foreign works. If we mark our products with the famous c in a circle, that consequently means that we think to enjoy a copyright, not that we have one. It will always be the courts to decide upon the originality of the program.

The rules with respect to ownership vary from country to country. In principle, copyright is vested in the author and can either not at all (Germany) or only in a limited way be transferred to another person. Consequently, only exploitation rights of software are exclusively or non exclusively assigned to a third party. Almost all countries know a system by which copyright is vested in the employer if the program has been written by an employee or under contract.

The owner of copyright has the monopoly on producing, offering for sale and copying the software. Exceptions are found in the permissible copying for back-up purposes, laid down in law by the US and Australia and seen as "fair use" by other countries. Making single copies for private purposes my also be allowed if done for non commercial reasons.

The adaptation of a computer program into another language or for use on another make of machine will be considered an infringement of copyright if made for commercial reasons. US law allows to adapt a purchased copy to one's own computer. If however, the modification goes further and the program is rebuilt with a modified structure, it may not be an infringement but a new copyrightable product. The reverse-engineered product will under these circumstances probably also not be considered infringing.

4. Trade Secrets and Confidentiality

In principle, the law of trade secrets and confidentiality seems to be mainly available with respect to contractual parties, it cannot really be considered to be a form of third-party-protection. Trade secret law is not well suited to a mass market where confidentiality restraints are counterproductive to the author's endeavour to sell as many copies of his software as possible. Also, in an environment where employee mobility and entrepreneurship are common, trade secrets can hardly be protected.

5. "Computer Crime"

is the keyword which is much quoted but there are next to no specific measures in the general penal codes and more often than not we read, that as software "thief" was acquitted of theft, embezzlement or other "general crimes" which were defined long before the computer age.

Criminal sanctions of various forms have been introduced in most national Copyright acts. One can get up to three years in Germany for infringing copyright but I have never come across such sentences. Once caught, the civil law provides plenty of punishment because in general, the software owner will get damages amounting to the profits he would have made, had he sold all the copies that were made by the infringing party. That looks sometimes as if everyone would be better off if he waited for a strong marketeer to nick his software and then reap the profits with the help of the courts later.

Computer software may or may not be protected for the original author. The fact that it says so on the console as soon as the program is installed may just be wishful thinking. As a ground rule, users should live on the proposition that the software they are finding on the market is protected; vendors should live on the proposition that it is not, hence some technical means of protection are called for.

DIALING OUT FROM THE HP3000

by
John D. Alleyn-Day
A H Computers
8210 Terrace Drive,
El Cerrito, CA 94530-3059
415-525-5070

Introduction

The divestiture of AT&T and other changes in the telecommunication industry have caused difficulties for many of us both as private consumers and data communication professionals. On the other hand, these changes have forced us to come up with creative solutions leading to improved and more cost-effective service. An example of this is a project which was brought to my attention by Underground Service Alert. In solving their problem, I learned some interesting things about the HP3000 and discovered methods of expanding its versatility.

Underground Service Alert has a central office which sends messages to several hundred locations in two states. They had been using private line facilities for several years with reasonable satisfaction, but two major problems had arisen. First, since the AT&T divestiture, the cost of private line facilities was rising astronomically. Secondly, the equipment USA used, a Telecommunication Processor and a protocol which dates from the old teletype days, was out of date. The Processor is actually a microprocessor with firmware on ROM that took care of all the protocol and timing needed on the private lines. The manufacturer of the Telecummunication Processor had gone out of business, making repair or upgrading of the system almost impossible.

The cost of dial-up facilities was an attractive alternative to the excessive cost of a private line system. To illustrate this, a cost analysis showed that, on one line which was a particularly long line with very little traffic, total telephone company charges could be reduced to one-tenth the existing charges by converting to a dial-up facility.

USA was impressed with the reliability and service available from HP. They wanted a new system that would save money on telephone service while avoiding non-HP equipment at their central site as much as possible. I took on the task of devising a way to put this new system together.

A brief description of the Underground Service Alert

operation is essential to understanding the trade-offs necessary in the decisions I made. USA is a co-operative organization of most of the companies that own underground facilities in the region. Included are gas, electric, telephone, cable TV, water and sewage facilities, and most cities and similar government bodies. USA takes calls from contractors and homeowners in Northern California and Nevada who are planning to dig. This data is entered into an IMAGE database online. The software then determines, from the map-references and from tables stored in the database, exactly which members have underground cables or pipes in the area and need to be notified. A program running as a batch job continually checks the database and transmits the messages as soon as they are ready by sending the appropriate data to the Telecommunication Processor.

The Problems

The new system I developed requires three crucial functions that are difficult to do with an HP3000 terminal port. The first of these is a way of diconnecting the modem when a call is finished. The simplest way of doing this is to use a modem that responds to the DTR (Data Terminal Ready) signal and to lower this signal to disconnect. The HP3000 gives no method of doing this, except by closing and opening the port, which is clumsy and time-consuming.

The second requirement is to know whether a call has been disconnected before it is complete. This is usually done by checking the DCD (Data Carrier Detect) signal from the modem. Again the HP3000 gives no reasonable way of doing this.

The third problem associated with the HP is that the ports are essentially half-duplex -- that is, they cannot read and write at the same time. Most devices such as modems assume that the other device can read and write simultaneously. This problem must be solved to make consistent communication a reality.

Additional problems which arose during the development of the new system were associated with the printers and indirectly with the HP3000. USA wanted to use inexpensive printers, but these generally use DC1/DC3 flow control protocol. The Series III has a notorious problem handling this protocol consistently, so it was important for me to overcome this limitation.

My final problem was to ensure that the message delivered to the printer was exactly the same as the one that left the computer center. Phone lines often introduce random errors in messages which can have two effects. If special characters are generated by the error, particularly the "escape" character, the printer may do peculiar things such

as flipping into graphics mode, which turns the message into strange-looking hieroglyphics. Even more serious is the possibility of distorting the message into something that looks correct, but which is, in fact, incorrect. Imagine a message that reads "Digging 2000 feet north of intersection" garbled so that one of the "0"s was converted to a non-printing character. The message would read "Digging 200 feet north of the intersection". Such an alteration could result in the contractor hitting an unsuspected facility with the possibility of property damage, injury, and loss of life; certainly with a major legal liability for USA.

The Solution

Although the HP has difficulty communicating with any outgoing modem, the first decision USA and I had to make was which modem to use. We chose Hayes-compatible modems because the Hayes protocol seems to have become the de facto standard. The operating system can handle incoming calls without difficulty, partly because it can be set up to handle the special signals to and from the modem (mainly Data Terminal Ready and Data Carrier Detect). The HP does not provide any mechanism for accessing these signals from a user program nor is there a method of examining the condition of the DCD signal without delving deeply into the operating system. Such a course would involve tricky experimentation and would make the software dependent on the particular version of MPE being used. For these reasons, I discarded this option at a very early stage.

Another option to render the modem compatible with the HP was to buy a considerable number of Telamon Network Engines and to write my software using these. This would have worked well, but, for ease of maintenance, USA wanted HP hardware only at the central site, to the extent this was achievable.

Alternatively, I could have used a microprocessor -- probably a Z80 CPU with many serial ports -- then written a program for this processor to do the actual telecommunication. This would be similar to the existing private line arrangement, but could be set up using standard parts on a standard buss, with spare parts available from several different manufacturers, protecting USA in case their principal supplier went out of business. In addition USA would own the source code and could make adjustments to it as necessary. On the other hand, although I can program both the HP3000 and the Z80, it could be difficult to find this particular combination of skills at some future date. This tentative solution also had the same disadvantage as the Telamon approach -- that of introducing additional equipment from third-party vendors.

Because we wanted to eliminate these possibilities, I looked closely at the feasibility of direct communication from the HP3000 to the modem. It is important to recognize that we had a set of circumstances that make such a possibility within reach. All the communication is under program control and the situation is very simple. USA's requirement is to dial a number, transmit a message in one direction to a printer and then disconnect. The system I designed to do this would not work in a situation requiring human interaction or the ability to dial to any modem on any computer and communicate satisfacorily. Such sophistication requires the use of Telamon's Network Engine or some similar device.

First I addressed the HP's problem with talking to a modem. The fact that HP terminal ports cannot read and write at the same time works well if the other device is co-operative. The HP3000 transmits a DC1 at the start of each read to indicate that it is ready to receive. However very few non-HP devices recognize this type of protocol. Most of them, the Hayes-compatible modems included, send back their data as soon as the carriage return is sent at the end of a record. A write followed by a read would almost certainly miss some or all of the reply.

One way of handling this is to connect two HP3000 ports to a single modem, performing the writes on one port and the reads on the other. By using no-wait I/O on the read port, the read is set up before the write and completed using the "nowait" intrinsic. However, this requires special cables, privileged mode and twice as many HP3000 ports (which are not inexpensive). USA was, however, quite prepared to go this way, if necessary.

Finally, I found a way to accomplish our objective using one port, a standard cable, and without privileged mode. The secret lies in the comment above about the modem obeying the command sent to it as soon as it receives the carriage return. In fact, for the modem, the carriage return is the "trigger character" that causes the command to be obeyed. The question I asked was, "How can I use this to achieve the results we want?".

Most people who use modems never explore much further than the dial command and possibly the interrupt sequence (+++) and the "on-hook" command. These three commands suffice for most purposes. However, tucked away in the instruction manual is a set of commands to alter various registers, such as timing parameters and the codes for the carriage return, backspace, and other characters. This is the key to making the modem compatible with the HP3000. I simply sent a command to the modem to change the carriage return character to a DC1!

I also altered the HP3000 port to make it resemble what the modem wants to see. In particular, I opened the port with "CCTL" and every command I sent had the control parameter in the "FWRITE" intrinsic set to "%320". This had the effect of sending the modem command line without any terminating carriage returns or line feeds. Now, when the "FREAD" command is executed, the HP3000 sends the DC1, which the modem recognizes as the trigger to execute the command and sends back its result to the HP3000 which is waiting for it.

I had to make two other adjustments. First, the standard reply from the modem is a carriage return, linefeed followed by the result in words and then another carriage return, linefeed. By using numeric result codes, I made the modem send back just a single digit followed by a single carriage return. Actually the record is terminated by a DC1 (because this is what the modem considers to be a carriage return), but since my program executes a single character read, the HP3000 never actually sees it. The second adjustment is to turn off the echo on the modem and the HP3000 port so that extraneous characters are not transmitted.

The manipulations I have described allow the HP3000 to do all the parameter setting and dialing necessary. Once a connection is made, transmitting the data should be no problem, unless, as we are, you happen to be working with a Series III. If we write multiple records to the port, and the remote terminal sends us a DC3, the HP will usually stop sending as expected. However, if we are unfortunate enough to receive the DC3 in between one record and the next, the control code will be ignored completely, reulting in a significant number of characters being lost entirely.

I was able to overcome this problem quite easily, although in a way that is not generally applicable. I sent one single record consisting of all the lines of the message with the appropriate carriage returns, linefeeds, etc. specifically included. Our messages were short enough that we could do this easily. The ultimate limit on this technique is, of course, the size of the stack, usually 32K. In fact, this is a technique similar to that used by the "PTAPE" command and intrinsic, which will not handle files longer than 32K because it reads them in as a single record.

Having transmitted the message, I still had some further problems. First of all, how could I know that the call was not disconnected in the middle? This is usually handled by checking the status of DCD (Data Carrier Detect). Since this signal cannot be checked via MPE, I had to find another solution. The answer lies in the use of the "interrupt" function on the modem. As some of you may know, when the modem is in the transmitting mode, the operator may enter three "+" signs in quick succession, and the modem will switch back to command mode and send an "OK" or, in short

code mode, a "0". If the modem was disconnected, it will already be in command mode. The escape sequence is then not recognized and no reply will be given. This serves as a method of detecting the mode of the modem and hence whether the modem is still connected.

There is still the problem of the half-duplex operation of the HP3000 port. If three "+" signs are sent, the reply will be missed or the DC1 sent by the read will spoil the escape sequence and the modem will not go into command mode at all. I solved this problem by using the same strategy as before. There is a register in the modem that defines the "escape" character, and I made this a DC1 instead of "+". Two DC1s can now be sent to the modem, followed by a read which sends the third. If the response "0" is given, then the program knows that the line was still connected when we finished and thus the message was sent completely. Any other response, and particularly a time-out on the read, indicates that the phone call was disconnected in the middle.

As a further refinement, I set the escape timer to zero, so that the timing of the escape characters is not critical. This avoids possible problems from a loaded machine that could cause a considerable delay between the write and the read.

All the reads from the modem are always timed reads. In some cases this is crucial to the operation, but in other cases it is only an assurance that, if anything goes wrong as it does from time to time, the port and the program will not be hung up.

Summary of Adjustments:

to modem:

1) change "carriage return" to "DC1"
2) use numeric result codes to make modem send back a single digit and carriage return
3) turn off echo
4) change modem so that escape character is "DC1" rather than "+"
5) set escape timer to zero

to HP3000:

1) open port with CCTL
2) send commands with control parameter in FWRITE intrinsic set to %320
3) turn off echo on port

Now that I had the modem working appropriately, I turned my attention to the printer at the far end. USA required an

inexpensive printer with a serial interface; one capable of using DC1/DC3 protocol. The simpler versions of many printers do not necessarily have these facilities. An additional requirement was a function which would lower the Data Terminal Ready when the printer is off-line or when the paper or ribbon runs out. This function can be used to prevent the modem from answering or to disconnect when these circumstances arise. We chose to use the Okidata 182 because we felt it would be reliable, but several other printers would probably work just as well.

As discussed previously, errors on the line can have a devastating effect on the remote printer. One possible solution, offered by some vendors, is a specially programmed modem that filters out objectionable characters. Although this eliminates the more gross aberrations, it does not protect against the textual errors so dangerous to USA's operation. I felt that this solution was inadequate and therefore searched for an alternative. There are now modems available with error-correcting firmware built into them. The list price of these modems is not much higher than that of a Hayes Smartmodem, and many of them use the Smartmodem command set with additions. We are using such modems made by Microcom using an error-correcting protocol called MNP, invented by them but employed by several other modem manufacturers. These modems allow us to transmit messages with the certainty that they will print out exactly the way that we intend them to.

There are some additional problems introduced by the use of these modems. The modems have buffers in them, so that the error-correction can occur. This means that they cannot disconnect immediately after the end of the message, because, if the line is noisy, some of the message may still be left in the buffer. Fortunately, the manufacturer has recognized this, and there are features to overcome this problem. Although not entirely satisfactory from certain points of view, they do allow us to operate well with our remote terminals.

Another rather nasty problem resulted from the combination of our special method of operation and the error-correcting modems. Because the modem will need to retransmit when an error occurs, it has to use DC1/DC3 protocol in order to control the output of data from the HP3000. When I turned on this protocol for the modem port, the whole system hung up completely, and it took quite an effort to discover the problem. Because the DC1 is both a protocol character and a character that is a part of the message, the DC1 was checked and thrown away before it could get through to the command portion of the modem firmware. I corrected this by using a special feature that permitted me to turn on the protocol in one direction only, so that the modem would send DC1/DC3 to the HP3000 but would not recognize it coming in the other

direction.

This system which I developed for Underground Service Alert is working well in actual production. As of May, 1987, USA has about 35 remote terminals up and running and they are adding more terminals with the ultimate goal of 400-500 terminals in locations over the states of Nevada and California. They are also realising the considerable cost-savings that I projected.

The Effective Use of a Professional Recruiter

Diane Amos, C.P.C.
Amos & Associates
633-B Chapel Hill Rd.
Burlington, NC 27215

Good Morning! I'd like to begin this presentation with a story.

There was this monastery where the monks had to obey a vow of silence, but after every 10 years they could say 2 words. George, our monk in question, lives his 10 years in total silence and then comes the day when he is allowed his 2 words. The priest asks him what he wants to say, and George states "BED HARD". The priest says,"Thank you very much" and ushers him out. Another 10 years go by and George comes in again. The priest says "My son, what would you like to say?" George says, "FOOD LOUSY". The priest says, "Thank you very much" and ushers him out. After 30 long years, an old George comes in once more for his precious 2 words. The priest again says, "My son, what would you like to say?" George says, "I QUIT!" The priest then says, "I'm not a bit surprised. All you've done for the last 30 years is bitch and complain!"

Wouldn't we all wish that we had employees that would stay with us through thick and thin for 30 years. But what we're here to discuss today is how to find those employees in the first place and I'm going to present one option to you. Some of you do use recruiters and some of you may not. If you do, I'm going to guide you through the process, giving you some DO's and DON'TS that will help you to maximize the return on your investment--that investment of hiring a qualified candidate.

Figure 1

Is this what most of you envision a Head Hunter to be? Some feel that we are necessary evils and charge outrageous fees. Well, I'm not here to defend that statement or speak to that issue. But if we are necessary, hopefully today's presentation will give you the tools that you need to make working with a recruiter a pleasurable and cost effective experience.

Why do companies even use Head Hunters? Corporations usually go the recruiter route today because of qualitative and quantitative reasons. They are finding increasing difficulty

locating and wooing the qualified job candidate. The balance of supply and demand is simply not in sync.

Figure 2

"Don't flush all of your dollars down the toilet!" Corporations may ultimately find themselves spending a large amount of money on ineffective advertising that brings in 20 unqualified candidates to every 1 qualified. Weeding through those candidates will also waste the managers time and money. Simply the cost of the unfilled vacancy on production may drain DP's budget.

In a recent 1984 survey of hiring managers, it was discovered that the size of the headhunting firm, the proximity to the client's location or the cost of the services was not as important in the selection of a recruiter as the measurement of professional experience, reputation and level of rapport.

Figure 3

So the first issue I'd like to address is how to select a reputable recruiter keeping in mind experience, reputation and level of rapport. Our industry is growing and many new people are entering this profession. I'm encouraged that it is becoming a more professional industry and you will have many recruiters to choose from. Sometimes having so many recruiters calling you is difficult and certainly you want the best you can get for the money. So here is some criteria that perhaps will be useful to you in making a selection.

A. Experience certainly is a testimonial to the recruiter's seasoning and success. Determine the length of time they personally have been in the placement business, the salary levels they've handled and the geographic territory they cover. This business is a high turnover industry and the ratio is 1 out of 20 make it. So if you are going to invest your time and money working with a recruiter, certainly you want to put it in a recruiter with a proven track record.

B. Just as you belong to INTEREX and various RUG associations where you can learn to better your skills, we have those organizations as well. Ask if your recruiter belongs to NAPC (National Association of Personnel Consultants), or if they belong to their State or Local associations. If they are truly interested in their profession, they will be active in those organizations. In addition, they should be acquainted with DP user groups to further educate themselves technically.

C. Ask if the recruiter is a CPC. This stands for Certified Personnel Consultant. As such, the recruiter would meet the qualifications set out by the NAPC which established this program to increase the professional level of job placement firms. To qualify to be a CPC, you must be in the business at least 2 years and pass an exam on general agency operations, state and federal laws and ethical practices. In our industry, there is an ongoing push for members to sit for this exam.

D. We have seen a trend in the last few years towards specialization in our industry. Not just specializing in DP as opposed to sales or accounting areas, but specializing in specific hardware areas, like IBM, HP or DEC. The successful consultants are those who do not try to handle everything, but rather those who specialize in a given marketplace building knowledge in that area and thereby credibility with their client companies. I'd advise choosing a recruiter who works the HP market exclusively, because they will have a better understanding of the marketplace, a larger supply of candidates and certainly have a commitment to the HP market. I can tell you from my own experience, that when I started working the DP market I handled everything--IBM, Univac, Burroughs, DEC. I did well but never truly developed client relationships or could speak knowledgeably in a given marketplace. The old adage--Jack of all Trades, Master of None was certainly in effect. The difficulty with trying to handle everything is that there is never time to completely establish a solid working relationship with a client and develop that level of rapport that the survey indicated was so important to hiring managers. An HP specialist should be your link to the marketplace, advising you on current hiring trends, salary surveys, and a general indication of what's going on in the world. Just for your information, I polled some of my own statistics last year, which I will share with you.

Figure 4

The survey criteria is based on both the HP1000 and HP3000 market, although 90% of it is for the HP3000 market. The placements column is for 1985/86 and the job order and candidates column is for 1986 only. The programmer analyst category covers all levels from entry programmers to senior programmer analysts as does the systems analyst category. The technical support category includes areas like DBA and data communications. I had to lump several categories together in the interest of simplicity and readibility. You'll notice that 62% of all my job orders and 61% of all

placements are in the programmer analyst category. This should not surprise any of you, as this is where the market is. It follows the breakdown of a typical DP shop where there are 6-8 programmer analysts, 2-3 systems analysts, perhaps a DBA, a technical support person and then a DP Manager. That ratio bears out in these statistics for the number of openings. You'll notice, however, that the number of candidates are more equal between categories. It appears that people are looking for jobs at all levels. What do you think are the most common reasons employees leave their jobs? Money is not the #1 reason. Most data processing professionals like to be continually challenged and when they feel they are starting to stagnate in a shop, it's time for them to move on. Also, a feeling of not being appreciated is a very common reason. So, managers, let your people know you appreciate them and try to keep them challenged and there's a good chance you can develop long term employees.

Figure 5

A. OK, assuming that you have selected your reputable recruiter to work with, the first thing you want to do is to give us a complete search assignment. The most important thing to begin with is to be descriptive. Know exactly what you're looking for. If you don't know, how can we find it? Asking for a P/A to do "normal P/A duties" means different things to different companies. Specs that are too broad open up the world and then you'll get flooded with unqualified candidates. Take some time to really discuss what you think you're looking for with us and perhaps we can even help you clarify your needs. My most successful placements have been where I had a client spend a good half hour upfront with me detailing the company, what he was looking for, the functions of the job, and why it was a good company to work for. Using that information, I was able to attract a top candidate quickly.

Figure 6

A helpful tip is to divide your requirements into a must and want column. Tell us what you absolutely must have in a candidate and then what you want if you can get it. Here is an example of how one company described their needs to me. Believe me, it was extremely helpful in determining what they valued the most. Feel free to state what you consider an Ideal candidate, but be realistic enough to know that you probably won't get it all. There is no such thing as a PERFECT candidate.

Of the musts, think of one or two major KEYs that are particularly important to you. You may settle for some qualifications as long as these most important areas are covered.

Be descriptive with the positions' job duties and functions. Our candidates want to know what they'll be doing and whether the position would even interest them or not. A programmer who really wants to get into more design will be enticed to a company that offers him that opportunity. But if we only have a vague idea of what they' ll actually be doing, there's no way we can recruit that candidate for you. Both you and I are in an information intensive business. Everything we do requires information. Therefore, the more information we have about our clients needs, the better equipped we will be to provide a worthwhile service to both the client and the candidate. Think of yourself for a minute as the user and I'm your systems analyst. Just as the analyst needs as much information as possible to solve your system problem, so do we recruiters need the information to solve your staffing problem.

B. Give us something to sell to the candidate about your company. What would make it attractive for them to come to work for you? I always ask this question of DP managers, and many times I'm shocked with the responses as if they've never even really thought about this. If you can't get excited about your company, how will we be able to attract the top talent you desire. I strongly recommend before starting any candidate search that you think about the positive things your company has to offer so that you can communicate them to us and ultimately the candidate. Some ideas may be to think about what you like best about your company, what future challenges you see, what the working environment is like--any flex time, family type atmosphere, an office window. Any PERKS that may be appealing to the candidate. Give us as much literature as possible about your company so that we can study and familiarize ourselves with your firm. Some good examples are an annual report, benefit summaries, relocation policy, employee newsletters, etc. Even information on the community where the job is located is helpful, so that we can inform our candidates about the area ahead of time. This will again avoid wasting your company's time and money if the location is totally unacceptable to the candidate or his family.

By the same token, you need to communicate to us any corporate skeletons in the closet to avoid any possible pitfall or embarrassment. Heavy turnover, possible corporate mergers must all be revealed. Don't try to hide any of this,

because it may come back to haunt us on the interview or offer stage, or worse yet after the candidate's been hired and then he leaves because of some surprise we didn't make him aware of. We as recruiters need to know both the negatives as well as the positives up front so that we can present a realistic picture to the candidate.

C. In filling your vacancy, timing is often the key. Tell us if you need someone immediately or if you have plenty of time. Let us know the urgency of your need. If it's urgent, let us know you're depending on us. Selecting 1-2 recruiters will give you the best results because that recruiter will work that much harder for you if he feels he has a reasonable chance to fill this position without competing against 20 other recruiters. I don't necessarily advocate giving the search exclusively to one recruiter, but if you do, let us know you're expecting results soon. The reason for this is what you don't see behind the scenes. We are busy with 20-30 searches at a given time, and there are only so many hours in a day. If I have a good client who has given me a search that is marketable and competitive, I know I have a reasonable chance of success,and then it makes good business sense for me to drop everything else and give that priority. But this goes two ways. If I have found a perfect candidate for you after dropping everything else, don't insist on waiting for paperwork. Resumes are fine for normal situations, but if it's urgent, this delay could cost you. Let me tell you enough about that candidate to determine if he's worth calling that night. The resume can follow. Certainly the burden is on us to determine if he is a good fit so as not to waste your valuable time, but if your need is truly urgent, then you must act with a sense of urgency. If I have recruited someone especially for your company, I've gotten him excited about the opportunity. He'll go cold on us if time is allowed to drag. This is where many companies lose good candidates. They drag their feet. And then we're all losers.

Figure 7

Once you've given a complete search assignment to us, let us know what you're expecting from the working relationship. A recruiter should be doing the ground work up front so that by the time you bring in the candidate for the interview, you know as much about him as possible technically and personally. Hopefully all that's left is for an eye to eye assessment to see if you like each other and if he fits into your group. This amount of up front work is what I feel you are ultimately paying us for. Not just a paperhouse to send you resumes that come floating across our desk. Our purpose

is to invest the time and effort in understanding your needs and the needs of the candidate to decide if there is a match for both concerned.

A. To accomplish this, I suggest that you require reference checking. I mean references in writing that you receive along with the resume. This will prove if the recruiter has really checked out the candidate or is just guessing. A reputable recruiter will give you the bad references along with the good and certainly you can double check behind us.You should also get as much technical screening as possible from us. I've devised a technical data sheet that I ask my candidates to fill out along with the application.

Figure 8

And I send that off to my clients so that they can see the depth of a candidate's experience in the various hardware/software/utilities areas. But if the recruiter does not have such a sheet, he should ask those technical questions of the candidate and relay them to you.

B. Along with the whole paperwork issue, I think it's reasonable for you to expect professional paperwork--not just someone's resume that was copied and sent on to you. You should be able to see from the paperwork that the candidate has been screened for you and the necessary questions asked. A cover sheet is helpful, with a summary determining the candidate's salary, job objective, present location, location desired, and whether he owns or rents. This will save you a great deal of time in handling that dreaded of all DP Manager's duties--screening resumes. A cover sheet will inform you up front that yes, he will come to your town, he is in the ballpark salary-wise, and he is looking for what you have to offer. As important as technical screening is, I believe that it is equally as important to screen a candidate for his personal and professional goals, and to make sure that those goals fit the goals of your company. If a recruiter hasn't done this, then you may pay for it by having your offer rejected and your time wasted.

C. Inform us up front of any particular pet peeves you may have in dealing with recruiters. This allows us to improve our service to you and again you're paying the bill. Be as candid and honest with us as possible. If you didn't like a candidate that we sent, let us know why so we will avoid making the same mistake again. I would much prefer a criticism than to be left in the dark. I polled my clients in preparation for this presentation and one of their most

common pet peeves was improper matching of the candidate to the position by the recruiter. Communication here again is the key! It's your responsibility to be clear and descriptive in what you're looking for, but also as Cliff Lazar of Systems Express said in his Chronicle article "Give the recruiter only one chance to send an unqualified applicant and then stick with it". Now, don't misunderstand me here. As I said before, we need to know if we're on track and occasionally we will make an honest mistake. But what I feel is unforgiveable is an outright total mismatch by a recruiter who sends an IBM Fortran programmer for an HP Cobol spot for example.

Another pet peeve is dealing with the "pushy" Headhunter. I personally don't feel that you need to be pushy to be a good recruiter, but rather be a good listener and ask the right questions. A recruiter should be in the game for the long haul, developing client relationships and not push for a fit that isn't really right. If you are dealing with a recruiter who is pushy, tell them how you feel and threaten not to do business with them. That should help to have them back off. Remember, you have a choice.

A flood of headhunting calls after you've put an ad in the paper is another common complaint. Unfortunately, your ad will generate numerous calls and even unsolicited resumes. What some companies do is return all unsolicited resumes and not take unsolicited calls, thereby putting the control back into their hands. Some of you try to run an ad along with working together with a recruiter, and that's fine, but this opens you up to that barrage from other recruiters that you have not chosen. Many companies don't use ads at all for exactly this reason. They like to choose which recruiters they'll work with and then be in control. And if they've chosen their recruiter properly, they'll be no need to advertise.

D. In choosing a recruiter, trust must be a guaranteed and assured part of your relationship. I'd like to dispell a myth some of you may have about recruiters. We DON'T recruit from clients. Your status with a recruiter will either be as a source company or a client, but in the development of long lasting client relationships, a good recruiter will take a hands off policy with their clients. Any relationship with a client that endures the test of time must be based on trust. Webster defines trust as "assured reliance on the character, ability, strength or truth of someone". I believe that in order to earn the trust of our clients, we must work very hard to provide a worthwhile service and absolutely must not cut corners.

Figure 9

A. OK, let's assume now that you have identified a suitable candidate that you would like to interview. Due to the expense of interviewing candidates particularly from a long distance, most companies like to conduct a phone interview prior to bringing the candidate to the company site. The phone interview if handled properly, can be an asset to the hiring manager. However many times this is where the interview process breaks down. Some DO's and DON'Ts here may help. Do call when you say you will. The fastest turn off for a candidate is sitting by the phone waiting for a call that never comes. Don't leave the time open (for your convenience), because you might catch the candidate at the worst possible time (in an argument with his spouse or in the midst of a party or with the kids screaming in the background). Certainly he will not be at his best for an interview. Instead, arrange with your recruiter, a convenient time that is scheduled with the candidate and follow through. Don't do all the talking, but draw the candidate out on his attitudes, his goals as well as his technical qualifications. Also this is the time to start selling your company to the candidate. Bottom line, if you're going to invest in the effort of the phone interview, make it worthwhile and valuable to your search process. Many of my clients have asked me to put together questions that they can use for this type of interview because sometimes they don't know which direction to take on the phone. I've done this and will leave a copy with you.

Figure 10

You need to begin your phone interview with a brief description of your company to break the ice and also to be informative. Then begin your questions from a general standpoint to get a good overall viewpoint of this candidate. Be descriptive about the position that you're interviewing him for, what he'll be doing and what challenges you have to offer. Then zero in on how he fits this position with particular technical questions relating to the position.

B. If after you've conducted the phone interview you decide you want to bring the candidate in for an on-site interview, tell your recruiter what the interview procedure is like. What can the candidate expect? Is there an itinerary with the potential interviewers listed by name and title? Let us educate the candidate and not send him to you blind. The more the candidate knows about the company ahead of time, the more he can relax and give you the valuable information

that you seek about him and his background. Use us as facilitators to work out the details and help you give a smooth running interview. Webster defines facilitate as to "make easy, less difficult, lessen the labor of". Let us earn our fee. If there is any testing to be done, let us know up front so that the candidate is not put off. I have several clients who give full 8 hour interviews complete with program aptitude testing, psychological exams, sometimes even a polygraph, and these days even drug testing. When the candidate is prepared for this, he generally can handle it all quite well. But the times when a client failed to inform me of this kind of testing, invariably the candidate was put off and looked unfavorably at the company as a result. No one likes surprises.

When the interview occurs, it is important for you as the interviewer to do your homework. Be prepared for the interview and don't take it lightly. Review ahead of time what's on the candidate's resume, what questions you intend to ask, and make sure everyone else who will speak to the candidate is also prepared. Interviewing should not be a game where you see how easily you can trip up a candidate. I strongly believe that it should be an information giving and information gathering session where two people meet to decide if they're right for each other. There is no sense in putting the candidate on edge. Keep in mind that the candidate is interviewing you as well. Make sure the impression you leave with the candidate is what you intend.

C. After the candidate has interviewed, give us some sort of feedback immediately or at least within 24 hours. The reasons are simple. If you like the candidate and are seriously considering him, you want to keep his interest and let him know you're interested. Even if you aren't ready to make a decision, communicate with us as to whether he's seriously being considered or not. That way we as facilitators can keep him on the burner for you or let him explore other opportunities. The best interview is to either rule him in or rule him out, so that timely decisions can be made by both parties. We need to work together at this stage and work as a team. If we know what's going on, we can help you get what you want. And believe me, we have more influence with the candidate than you may think. They are relying on us for our opinion and expertise in this situation.

The feedback stage is also an area where you managers need to educate personnel. They can blow it for you. If we are forced to deal through personnel, make sure you get your information back to them and to us as quickly as possible.

Chances are you may be seriously interested in the candidate, but if personnel hasn't communicated that to us, it's possible that you may lose the candidate. I have a horror story to relate here. My associate was working a hard to fill and narrow search assignment, and she found an ideal candidate which the company agreed they wanted to interview. Timing was extremely tight as the candidate was expecting another job offer soon. Long story short, we experienced a succession of delays, a final written offer forgetting to include the relocation package, and statements from personnel like, "I don't have time to discuss with her any questions she may have; I don't work weekends; if she doesn't want the job, just tell her to turn it down". This kind of no-care attitude really reflected poorly on the company, and the candidate ultimately did turn the job down, much to the despair of the department manager who had to start the search process all over again. They found a candidate that they wanted and then let her slip through their fingers.

D. By the same token, you should require immediate feedback from us. We should have spoken to the candidate shortly after the interview, and we can save you valuable time if you're considering someone who is not interested. Or more importantly, if the candidate has any questions or concerns about the interview, this is the time he will reveal them. You will want to know how the candidate feels about you and the position.

E. When it comes down to the offer, this is definitely the time to work closely with us. If there are any concerns that the candidate has at this point, they need to be addressed and worked out. It's important for you to get involved to help us as facilitators to work out any concern the candidate may have. Perhaps the candidate is worried about housing. Any information you can provide may be the critical key to his acceptance. Perhaps he is concerned about potential advancement with the company. Time that you can take to either speak directly with the candidate at this critical time or through us needs to be taken. You've gone this far with the candidate and invested a lot of time and effort, so a little more to make sure he's comfortable with his decision is not too much to ask. Our role as facilitators is to unmask any concerns the candidate may have, so if answered may free him up to accept your offer.

The offer stage is also where timeliness is essential. If you know that the decision will take awhile, inform us of this up front. If the candidate is prepared once again, he generally will be patient. There have been many instances

when I've had to use all of my imagination to cover for a company's indecision, but if they kept me informed of exactly what was going on, I could generally be successful for them. Remember, we're working in your best interests here. The worst scenario is when the company says "I'll make a decision next week", the decision keeps being put off, and we're all kept in the dark as to why. This only serves to reflect poorly on your company and often times is a reason that an offer is turned down. Delay creates doubt, and the longer you drag your feet in making a decision, the longer the candidate may begin to wonder whether he truly wants to go to work for you. I have a client company whose personnel manager relayed a horror story to me where they interviewed 6 viable candidates, narrowed it to 3 (any of the 3 would have been fine) and then dragged their feet for the decision. When they finally decided, all 3 of the candidates had found other jobs and this company was back to square one. Believe me, the company learned its lesson and this personnel manager makes sure her department managers don't repeat the mistake. I can attest to this, because I placed someone there where my candidate was selected from 2 good possibilities and offered the job 2 days after the interview. They knew he was right and there was no need to continue searching just to see more to compare. Remember the motto: "Rule them in or rule them out"!

The offer stage is the time for direct and honest communication between us and you, working together as a team. We should be an arm of your staff, an extension of your company working with you to get you what you want.

And hopefully, what the result of this effort will be is a satisfied client with a satisfied new employee.

Figure 11

They have found each other, they are right for each other, and hopefully it is a marriage made in heaven.

To review the points I've made in just a few words,my suggestion if you choose to use a recruiter, is to first of all spend a small amount of time selecting a reputable recruiter. Use the old adage "Buyer Beware". Educate yourself, ask sufficient questions which will help to develop a strong pattern of trust and confidence with your favorite recruiter. Look for experience, activity in personnel consulting associations, a CPC designation and a commitment to specialization.

Then take the responsibility to give us a complete search assignment with descriptive requirements and descriptive job functions. Give us as much information as possible on your company which will help us to sell your company to our candidates. Inform us of the urgency of your need so we can move quickly if necessary.

Be clear in what you require of us. Let us know you want reference checking, technical screening of some kind, and expect professional paperwork.

When the interview rolls around, work closely with us in conducting a useful phone interview, in giving us details of what to expect on the site interview, give and expect timely feedback and then be willing to work out any areas of concern that may hinder the offer being accepted. The closer you are willing to work with us, the better chance you'll have of getting what you want ultimately--and that is a successful hire.

What I've really said for the last hour is in order to get your money's worth out of a recruiter, communicate with us. That's the bottom line--communication.

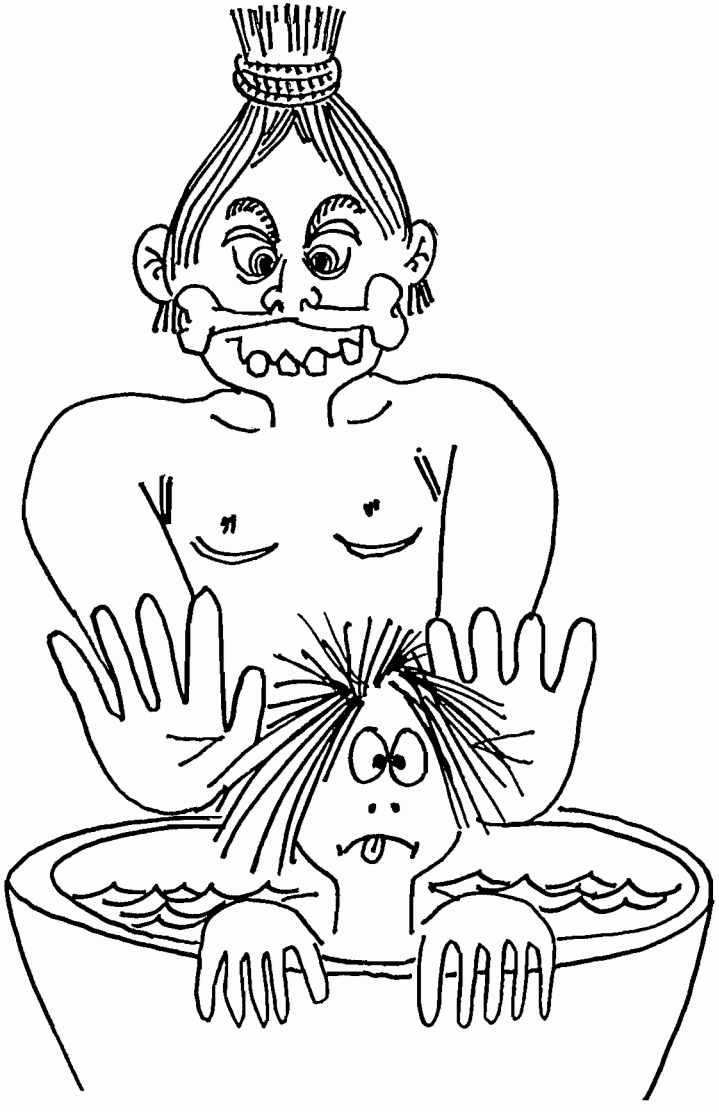


Figure 1

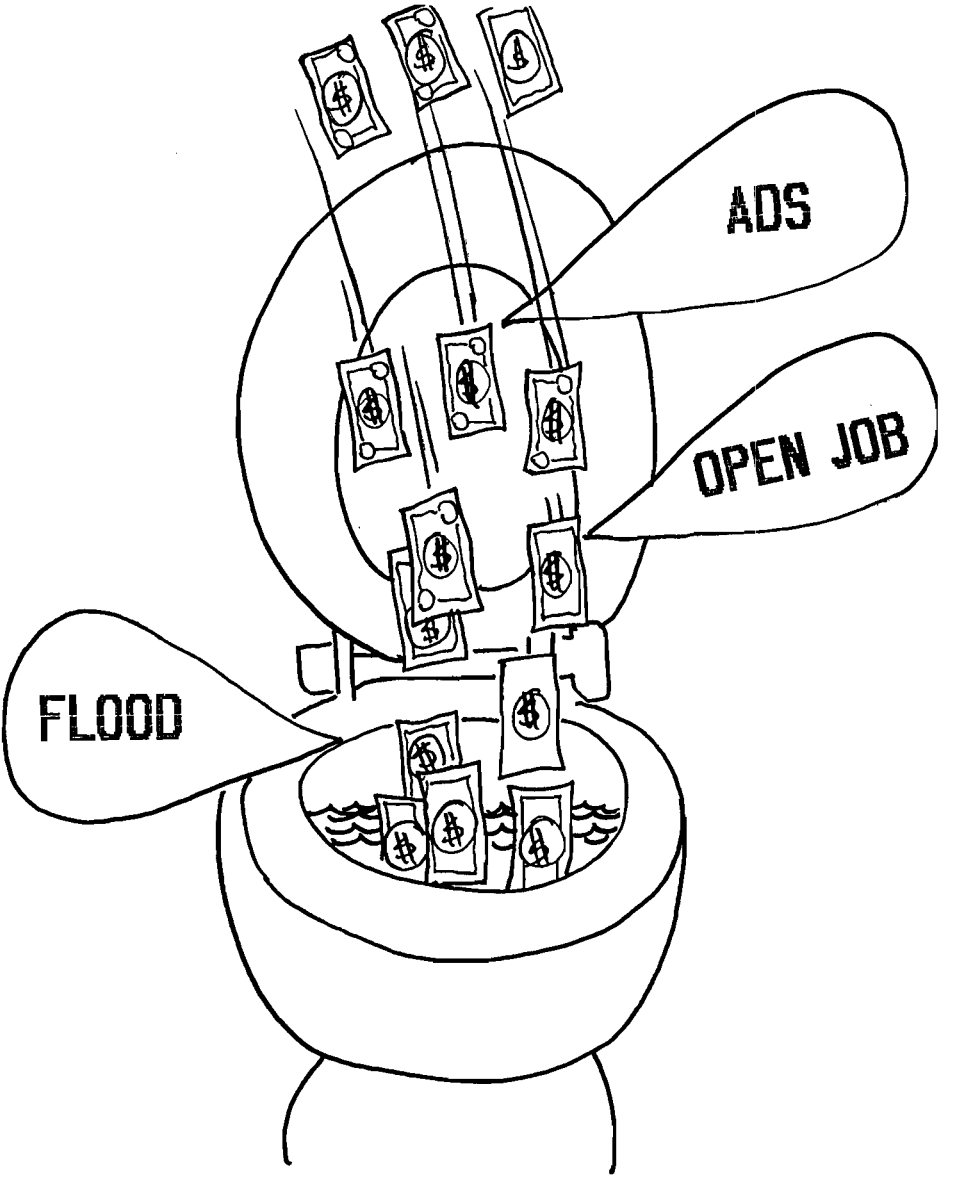


Figure 2

SELECT
A REPUTABLE RECRUITER

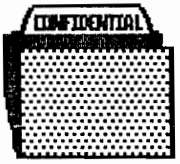
A. EXPERIENCE?
B. ASSOCIATION MEMBER?
C. CPC?
D. SPECIALIST?

Figure 3

AMOS & ASSOCIATES - SALARY SURVEY

TITLE	AVERAGE $	JOB ORDERS	CANDI-DATES	PLACE-MENTS
PROGRAMMER ANALYST	27,175	62%	38%	61%
SYSTEMS ANALYST	32,294	17%	13%	11%
SYSTEMS MANAGER	35,667	1%	10%	2%
TECH SUPPORT	33,325	3%	3%	7%
SE	29,398	2%	8%	7%
PROJECT MGR/LEADER	40,730	2%	5%	1%
DP MANAGER	37,058	4%	10%	7%
MIS MANAGER	44,899	3%	5%	5%
HARDWARE	31,543	3%	2%	-
OPERATIONS MGR	31,875	2%	1%	-
COMPUTER OPERATOR	16,460	-	2%	-

Figure 4



GIVE A COMPLETE SEARCH ASSIGNMENT

A. BE DESCRIPTIVE
B. SELL YOUR COMPANY
C. TIMING-URGENCY?

Figure 5

POSITION REQUIREMENTS - SYSTEMS MANAGER

MUST/WANTS	
W	Programming Background
M	Keep Computers Running
M	System Performance/Optimization
M	System Upgrades
M	Trouble Shooting (Hardware/Software)
M	Configuration of the System
W	Knowledgeable of System Utilities
M	Account Structure Knowledge
W	Telecommunications (DS/RJE)
W	Goods Interpersonal Skills
W	Ability to Develop Peers/Subordinates
W	Able to Write Procedures

Figure 6

STATE YOUR EXPECTATIONS

A. REQUIRE REFERENCES/ TECHNICAL SCREENING

B. EXPECT PROFESSIONAL PAPERWORK

C. EXPRESS PET PEEVES

D. TRUST

Figure 7

AMOS & ASSOCIATES
633B Chapel Hill Road
Burlington, NC 27215

SKILLS CHECKLIST

Name _____ Date _____

Please rate your skills on a 1 to 10 basis (10 high) or use "S" (school experience only) for each skill. Use back of form for comments and any pertinent information not on Checklist.

HARDWARE	SKILL RATING	NO. YRS. USED	LAST YR. USED
HP1000 SERIES			
HP3000 SERIES			
HP9000 SERIES			
SPECTRUM SERIES			
HP150/VECTRA			
IBM-PC			
MULTI-SYSTEM ENVIRON.			

OPERATING SYSTEM			
RTE			
MPE			
HP-UX			
MPE-XL			
MPE INTERNALS			
RTE INTERNALS			

3RD PARTY PACKAGES			
ASK MANMAN			
ASK OMAR			
MASH/E			
QMS			
AMAPS/3000			
BOSS/3000			
MM 3000			
PM 3000			
QED/3000			
MCBA			
GTS			
HP F/A			
SPEEDLEDGER			
MULTIVIEW			
PROFILES/3000			
FMS-80			
COLLIER JACKSON			
SFD			
TYMLABS			

APPLICATIONS			
ACCOUNTING			
MANUFACTURING			
ENGINEERING			
SCIENTIFIC			
PROCESS CONTROL			
INSURANCE			
GOVERNMENT			
EDUCATON			

OTHER			
STRUCTURED PROGRAMMING			
STRUCTURED DESIGN			
SYSTEMS DESIGN			
DATA COMMUNICATIONS			
SYSTEMS MANAGEMENT			
DATA BASE ADMINISTRATION			
PROJECT MANAGEMENT			
MIS MANAGEMENT			

LANGUAGES/UTILITIES	SKILL RATING	NO. YRS. USED	LAST YR. USED
COBOL			
FORTRAN			
BASIC			
BUS BASIC			
RPG II/III			
SPL			
ASSEMBLER			
PASCAL			
C			
PROTOS			
TRANSACT			
REPORT 3000			
INFORM 3000			
DICTIONARY 3000			
FASTRAN			
VPLUS/3000			
QUICK			
QUIZ			
QTP			
QDD			
DICTIONARY PLUS			
EXPERT			
SPEEDWARE			
MICRO SPEEDWARE			
TOOLSET			
HP MENU			
DSG/3000			
GRAPHICS/1000			
QDM/1000			
FORMS/1000			
PMC/1000			
PCIF			
CAL			

DATABASE MANAGEMENT			
IMAGE 1000			
IMAGE 3000			
TURBO IMAGE			
ADAGER			
DB GENERAL			
KSAM 3000			
QUERY 3000			
IMAGE INTERNALS			
HPSQL			
HPIMAGE			
RELATE 3000			

DATA COMMUNICATIONS			
RJE			
MRJE			
DS 1000/1000			
DS 3000/3000			
DS 3000/1000			
X.25			
LAN			
MODEMS/TELECO			
SNA NRJE/IMF			
FDL			
NS/1000			

Figure 8

THE PHONE INTERVIEW

I. DESCRIBE YOUR COMPANY BRIEFLY

- A. PRODUCT OR SERVICE
- B. HARDWARE/SOFTWARE IN SHOP
- C. # OF EMPLOYEES/# IN DP
- D. WHAT IS ATTRACTIVE ABOUT THE COMPANY - PERKS

II. GENERAL INTERVIEW QUESTIONS

- A. WHY ARE YOU CHANGING JOBS?
- B. WHAT ARE YOUR LONG/SHORT TERM GOALS?
- C. TELL ME WHAT YOU'RE DOING PRESENTLY
  1. What do you like best about your present job?
  2. What do you like least about your present job?
- D. WHAT DID YOU DO IN PREVIOUS JOBS?
- E. WHAT WERE YOUR MAJOR ACCOMPLISHMENTS IN YOUR CAREER?
- F. HOW DO YOU FEEL ABOUT YOUR PROGRESS TO DATE?

III. DESCRIBE POSITION INTERVIEWING FOR

- A. MAJOR JOB FUNCTIONS
- B. CHALLENGES HE'LL HAVE
- C. WHY IT'S AN ATTRACTIVE POSITION
- D. FUTURE CAREER PATH

IV. QUESTIONS RELATING TO THE POSITION

- A. WHAT KIND OF EXPERIENCE DO YOU HAVE FOR THIS POSITION?
- B. AS I DESCRIBED THE POSITION, WHAT INTERESTS YOU MOST ABOUT THIS POSITION?
- C. DO YOU HAVE ANY QUESTIONS?

Figure 9

THE INTERVIEW/HIRE

A. PHONE INTERVIEW
B. DESCRIBE SITE INTERVIEW
C. GIVE TIMELY FEEDBACK
D. REQUIRE FEEDBACK
E. SOLVE AREAS OF CONCERN

Figure 10

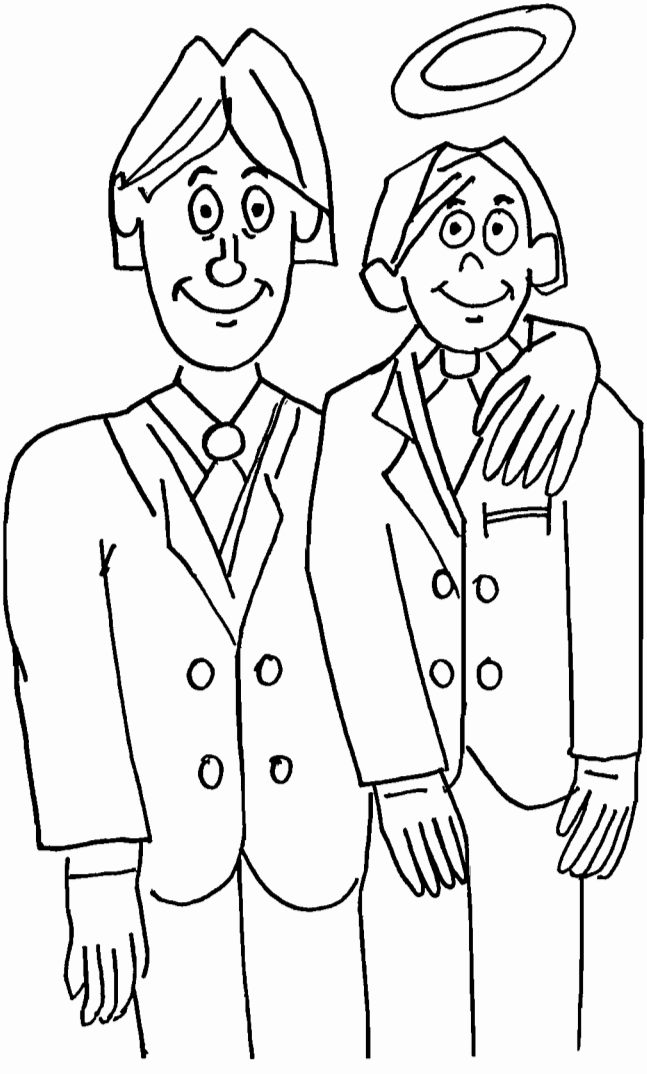


Figure 11

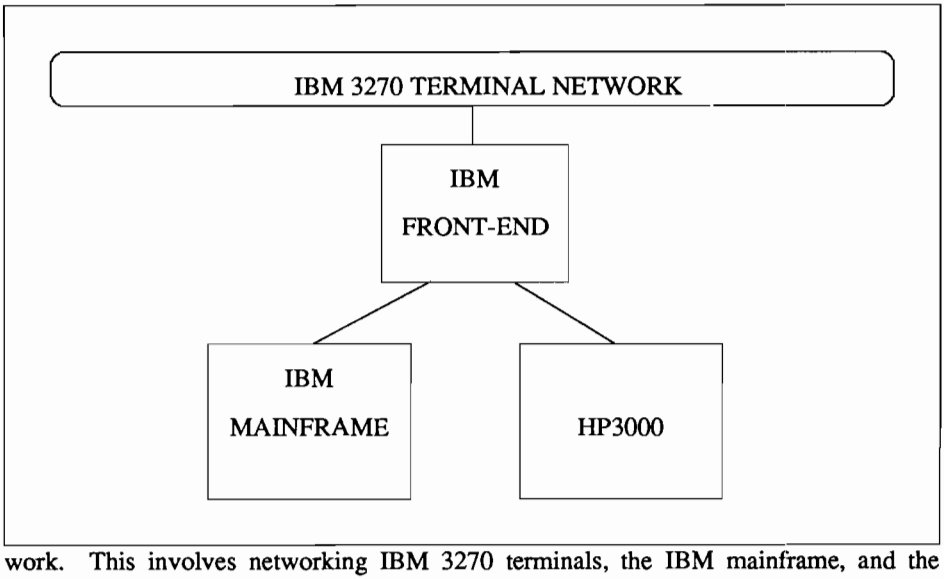
ACCESSING HP3000s FROM IBM TERMINAL NETWORKS

Jim Antonucci
Dale Nickels
Forest Computer Inc.
1749 Hamilton Road
Okemos, MI 48864

1.0 HP3000 RPT Defined.

IBM 3270 and compatible terminals are a type of terminal used for interactive communications with IBM mainframes. It is not unusual for large corporations to have a network of thousands of IBM 3270 terminals connected to one or more IBM mainframes. Without reverse pass thru, these terminals can only be used to access an IBM mainframe. Reverse pass thru is the name of the technique of providing these IBM terminals with access to non-IBM computers.

HP3000 reverse pass thru is defined as the ability to access HP3000 interactive applications from an IBM 3270 terminal connected to an IBM mainframe in the same net-



work. This involves networking IBM 3270 terminals, the IBM mainframe, and the HP3000 in such a way that the IBM 3270 terminals have access to interactive applications on both the IBM mainframe and the HP3000.

2.0 Uses of HP3000 RPT.

HP3000 reverse pass thru is of interest to companies which have already installed both HP3000 computers and IBM mainframes and companies which have only installed

IBM mainframes but are interested in installing an HP3000. Both types of companies can benefit from the integration of computer resources and cost savings that can be provided by HP3000 reverse pass thru.

2.1 Integration of Current Computer Resources.

To a company which has already installed both HP3000 computers and IBM mainframes, HP3000 reverse pass thru provides a solution for integrating their IBM and HP3000 computer resources. HP3000 reverse pass thru would provide the IBM users access to HP3000 resources, and IBM 3270 emulation (a topic not discussed in this paper) would provide HP3000 users with access to the IBM resources. This would provide both the IBM and HP3000 users access to both the IBM and HP3000 resources, allowing the data and application software on each computer to be shared.

2.2 Installation of New Computer Resources.

A company which has installed only IBM mainframes and has a large IBM 3270 terminal network may have several reasons for being interested in installing an HP3000 computer. The HP3000 computer line offers a full range of processor sizes, each competitively priced and compatible with one another. An entry level HP3000 computer offers the opportunity to install a departmental system or a system which will be dedicated to a new application for under $50,000.

An HP3000 computer can also cost much less to operate than an IBM mainframe. It typically requires less staff for programming and operating and less stringent environmental considerations. Quality software in the areas of office automation, manufacturing, data base management, application program development, and other horizontal and vertical software available from HP and third parties can also make the HP3000 attractive.

Without HP3000 reverse pass thru, a company with an IBM mainframe which is interested in the benefits an HP3000 computer has to offer would be faced with installing a second terminal network and additional terminals. This may prohibit a company from taking advantage of the benefits of an HP3000. HP3000 reverse pass thru offers a solution for installing an HP3000 computer and making it accessible from an existing IBM 3270 terminal network.

2.3 Cost Control.

HP3000 reverse pass thru also provides a solution for controlling costs. A company with both IBM and HP3000 computers can standardize on one terminal network. A single IBM 3270 terminal network can be used to access both IBM and HP3000 computer resources. A company would not have to install and maintain two different terminal networks, and the dual role of each IBM 3270 terminal could reduce the number of terminals required.

HP3000 reverse pass thru can also help extend the life of investments in HP3000 hardware and software. A company interested in making HP3000 based applications accessible to its IBM mainframe users does not have to undertake a costly conversion

effort to convert the applications to run on the IBM mainframe. HP3000 reverse pass thru is a solution for providing the IBM mainframe users access to these applications without converting them.

3.0 Incompatibility Issues.

An HP3000 reverse pass thru solution must deal with the differences between an IBM 3270 terminal and an HP terminal. These include differences in their keyboards, terminal operation, and cabling.

3.1 IBM vs. HP keyboards.

Most characters map easily between IBM 3270 and HP keyboards. For example, there is an "A" character on each keyboard. There are some characters on an HP keyboard which do not exist on an IBM 3270 keyboard, most notably the square brackets ("[" and "]"), control, and break key. Some HP characters have similar but not exact IBM equivalents. For example, the caret ("^") is often mapped to the logical not ("¬"). The IBM 3270 keyboard also has keys which do not have equivalents on an HP keyboard such as the cent sign ("¢"), solid vertical bar ("|"), and the dup and fieldmark keys. Both terminals have similar local editing keys although the IBM 3270 terminal lacks insert and delete line keys.

Another difference between the two terminals are the special function keys. The HP keyboard has 8 general purpose function keys. They can be programmed to perform local functions on the terminal or to transmit to the host. The IBM function keys are PF1 thru PF24, PA1 thru PA3, and CLEAR. Some IBM 3270 terminals do not have all 24 PF and 3 PA keys. All of these keys transmit to the host. The PF keys transmit data from the screen along with the key code. The PA and CLEAR keys just transmit the key code.

3.2 IBM vs. HP Terminal Operation.

HP and IBM 3270 terminals operate differently in the way data is presented to the user.

The most common IBM 3270 terminals have 24 lines of 80 characters. HP terminals also have 24 by 80 screens, but the screen displays part of a much larger terminal memory. Most HP terminals have at least 48 lines of memory. Some have much more. As data is sent to the terminal, lines may scroll off the screen, but remain in terminal memory. The IBM 3270 terminal has no equivalent to scrolling. The only way to move data on the IBM screen is to send a new screen of data to the terminal.

The IBM 3270 terminal always operates in block mode. That means that any keystrokes are processed locally. Data is sent to the IBM mainframe only when the enter or PF key is pressed. The HP terminal has many operating modes. The operating mode is controlled by a number of flags called "straps" which can be controlled by the terminal operator or by the HP3000 application program. These straps are block/character mode, line/page mode, forms/nonforms mode, and two handshake straps. This makes 32 modes some of which overlap. Block-page-forms mode is most similar to the IBM 3270 block mode and is also the mode used by VPLUS/3000 ap-

plications. Character-line-nonforms mode is the HP terminal mode used by most HP character mode applications. There is no equivalent to this mode on an IBM 3270 terminal.

The HP terminal has 5 ways to enhance the data displayed. These are blinking, inverse video, underlined, half-bright, and secure (i.e. hidden). These can be combined in any way for a total of 32 combinations. The basic IBM 3270 terminal supports only 3 display enhancements, normal, bright, and secure. Some IBM 3270 terminals also support extended highlighting which adds blinking, inverse video, and underlined. These enhancements can be combined with the basic enhancements but not with each other. This gives a total of 12 enhancements.

IBM 3270 and HP terminals differ in the way they present data to the user. On an HP terminal, no space on the screen is taken up by transitions between protected to unprotected data or by changes in enhancements. On the IBM 3270 terminal, any change takes a position on the screen. This position is occupied by the attribute byte which is displayed as a normal video blank. In order to display a form from an HP application on an IBM 3270 terminal, compromises must be made. Either fields must be moved so that they appear in different locations on the IBM 3270 screen or attribute bytes have to be planted over screen data.

3.3 IBM vs. HP Terminal Cabling.

IBM 3270 and HP terminals differ in the way they are physically connected to their respective computers. The type of cable, protocol, and network topology used by each type of terminal is different.

HP terminals typically use point-to-point connections to the HP3000. Even though there may be statistical multiplexers and/or modems, the link is ultimately an end-to-end connection from the terminal to a single port connection on the HP3000.

In contrast, IBM 3270 terminals are connected by a coax cable to a cluster controller. Several terminals may be connected to each cluster controller. The cluster controller may be connected by a high speed channel directly to the IBM host or via a remote datacomm line to a communications front end which itself is connected either to the IBM host or another front end. Complex IBM networks of many hosts, front ends, and controllers can be designed.

The HP terminal to HP3000 connection uses a very simple asynchronous ASCII communication protocol. When you hit a key, a character is sent to the HP3000. When the host sends characters, they show up on the terminal. Only a slightly more involved handshake is used to support block mode transfers. There is no error checking or sequence numbering of data messages.

The IBM cluster controller communicates with the front end via SNA SDLC or BISYNC, each a multi-level protocol. Data for all the terminals connected to the cluster controller is sent between the cluster controller and the front end over a single

link. Messages are broken into segments and assigned sequence numbers. Error checking and recovery are done to make sure the data arrives correctly at its destination.

4.0 Approaches to Providing HP3000 RPT.

All products which provide an HP3000 reverse pass thru solution must resolve the same incompatibility issues although they differ in their approaches to resolving them. This is due to the choices that are made when the solution is designed. The important design choices are: (1) where will the CPU resources come from to provide the solution; (2) what changes to the network will be required to install the solution; and (3) what level of compatibility with HP3000 applications will be provided by the solution.

4.1 CPU Resources.

HP3000 reverse pass thru, like any other application, consumes CPU resources. Unlike most other HP3000 applications, the CPU resources it consumes is not necessarily those of the HP3000. In fact most reverse pass thru solutions consume CPU resources other than those of the HP3000.

The CPU resources that a solution can consume are those of: (1) an HP3000; (2) an IBM mainframe; (3) a gateway computer; and (4) a black box processor. The HP3000 CPU resources are those of the same HP3000 which IBM users are accessing. The IBM mainframe CPU resources are those of the IBM mainframe to which the IBM 3270 terminals are connected. A gateway computer is a computer other than the IBM mainframe, to which the IBM 3270 terminals are connected, or the HP3000 which the IBM users are accessing. Ideally the gateway computer will have a better architecture than that of the HP3000 or IBM mainframe for handling the real time requirements of data communications and the intensive character by character processing required by a reverse pass thru solution. A black box processor is similar to the gateway computer in that it is a CPU resource other than the HP3000 or the IBM mainframe. A black box processor is usually designed to perform a specific task. Black box processors used in reverse pass thru solutions are typically a type of protocol converter.

The task of performing HP3000 reverse pass thru can be spread across one or more of these four CPU resources. The cost of the CPU cycles necessary to perform reverse pass thru depends on which CPU resources are used and which reverse pass thru related tasks are performed on each CPU resource. Typically the IBM mainframe CPU cycles are most expensive, the HP3000 CPU cycles are the second most expensive, and the gateway computer and black box processors are third and fourth respectively.

The tasks of performing HP3000 reverse pass thru can be broken down to terminal emulation, protocol conversion, and message switching. Terminal emulation being the most CPU intensive task, protocol conversion being the second most CPU intensive task, and message switching being the least CPU intensive task. The choice of which CPU resource will be used for each task will affect the CPU cycle cost of the HP3000 reverse pass thru solution's design.

4.2 Network Changes.

Allowing a network of IBM 3270 terminals to access an HP3000 will require changes to the network. The changes that are required depend upon the choices that are made when designing the HP3000 reverse pass thru solution concerning which protocols will be used and how the IBM 3270 terminals will be physically linked to the HP3000.

An IBM mainframe supports SNA SDLC and BISYNC protocols. An HP3000 will support HP asynchronous point-to-point, typically used for connecting terminals to the HP3000, SNA SDLC, BISYNC, and other protocols. Some HP3000 reverse pass thru solutions connect an SNA SDLC or BISYNC line on the IBM with the same type of line on the HP3000. Other solutions take an SNA SDLC or BISYNC line on the IBM and connect it to HP asynchronous point-to-point lines on the HP3000.

In addition to the protocol choices, physical links must be installed between the IBM and HP equipment. The HP3000 can be physically connected to the IBM mainframe, typically through the IBM front-end processor, or it can be physically connected directly to the IBM 3270 terminal network. If connected to the IBM mainframe, a message switching application will be required on the IBM mainframe to pass the data streams between the HP3000 and the IBM 3270 terminals. If connected directly to the IBM 3270 terminals, a hardware or software switch will be required so that the IBM 3270 terminal will be able to be switched between the HP3000 and the IBM mainframe.

Depending upon the networking choices that are made when designing an HP3000 reverse pass thru solution, the solution may or may not be shared among all the IBM 3270 terminals in the network. For example, a black box processor installed between an IBM 3270 terminal and the HP3000 may not be shared by other IBM 3270 terminals. Each IBM 3270 terminal which must access the HP3000, regardless of how frequently it must access the HP3000, will require its own black box processor and data communication line into the HP3000. The impact of this requirement depends on how many IBM 3270 terminals may need to access the HP3000. On the other hand, a solution which resides on the IBM mainframe may be shared by any IBM 3270 terminal which is connected to the mainframe but, depending on its design, may require more costly CPU cycles or provide less compatibility with HP3000 applications.

4.2 Compatibility With HP3000 Applications.

Another important design consideration is the level of compatibility with HP3000 applications which the HP3000 reverse pass thru solution will provide. Making an IBM 3270 terminal emulate all the functions of an HP terminal is a difficult task. The extent to which a solution can emulate all of these functions will dictate the number of HP3000 applications for which the solution is compatible.

Most HP3000 applications can be categorized into character mode or block mode (VPLUS/3000) applications. The tasks of emulating character mode and block mode applications are quite different. Emulating both modes requires much more work than

emulating just one. When designing a solution, a choice will need to be made as to whether character mode, block mode, or both will be required.

Supporting both character mode and block mode applications does not necessarily guarantee that all HP3000 applications will be supported. Some applications make use of seldom used character or block mode features which may not be supported by the solution. Some of these applications include HPDeskManager, MM/3000, and Powerhouse. When in doubt, an application should be tested with a solution to verify that the application can be supported.

5.0 A Sampling of HP3000 Reverse Pass Thru Solutions.

There are several products or combinations of products which provide an HP3000 reverse pass thru solution. Listed below are a few of these solutions listed in tabular form. Indicated in these tables are some of the design choices, described above, which were made by the product's designers.

5.1 Several HP3000 Reverse Pass Thru Solutions.

Company	Solution	Description
Carolian	ITLINK	ITLINK is software which runs on the IBM mainframe and the HP3000 and communicates using HP's IMF/3000 software and link interface card.
Forest Computer	GATEWAY/1000	GATEWAY/1000 is a gateway computer which can be connected to one or more HP3000s and IBM mainframes. Terminals in the IBM 3270 network access GATEWAY/1000 using the IBM Host Command Facility.
Gandalf Data	COAX 3270	COAX 3270 is a protocol converter which is connected to an IBM terminal's coax cable and provides the terminal with VT100 terminal emulation. With PREVIEW from Tymlabs, this solution can also access VPLUS applications.
Simware	SIM/PASSTHRU	SIM/PASSTHRU runs on the IBM mainframe and provides IBM 3270 terminals connected to it with virtual terminal capabilities to a remote IBM mainframe via the IBM bisync protocol. HP's MTS/3000 software and link interface card will allow the SIM/PASSTHRU bisync line to be connected to an HP3000.
(Any personal computer vendor)	(Any personal computer)	A personal computer with both IBM 3270 and HP terminal emulation software and hardware.

5.2 CPU Resources Used By Solutions.

	IBM Mainframe	HP3000	GATEWAY COMPUTER	BLACK BOX
Terminal Emulation		ITLINK SIM/PASSTHRU[1] COAX 3270 [2]	GATEWAY/1000	COAX 3270 PC
Protocol Conversion		ITLINK [3] SIM/PASSTHRU[4]	GATEWAY/1000	COAX 3270 PC
Message Switching	ITLINK GATEWAY/1000[5] SIM/PASSTHRU		GATEWAY/1000	COAX 3270 PC

[1] The terminal emulation tasks are performed by MTS/3000.
[2] The terminal emulation tasks for VPLUS applications are performed by PREVIEW.
[3] The protocol conversion tasks are performed by IMF/3000.
[4] The protocol conversion tasks are performed by MTS/3000.
[5] The message switching tasks are performed by the IBM Host Command Facility.

5.3 Protocols and Physical Links Used by Solutions.

	ITLINK	GATEWAY/1000	COAX 3270	SIM/PASSTHRU	Personal Computer
SNA SDLC	X	X	X		X
IBM BISYNC	X		X	X	X
ASYNC Pnt-to-Pnt		X	X		X
Host-to-Host link	X	X		X	
IBM 3270-to-HP3000 link		X	X		X
Shared by all terminals	X	X		X	

5.4 HP3000 Application Programs Supported by Solutions.

	ITLINK	GATEWAY/1000	COAX 3270	SIM/PASSTHRU	Personal Computer
Character mode applications		X	X	X	X
VPLUS applications	X	X	X [1]		X
HPDeskManager [2]		X			X
MM3000 PM3000		X			X
Powerhouse [3]		X	X		X

[1] When used with PREVIEW on the HP3000.
[2] Complete HPDeskManager support including HPDESK's block mode editor.
[3] Complete Powerhouse support including QUICK.

DISC CONTROLLER CACHE

What is it and How Does it Improve Performance?

MELODY ARMSTRONG
HEWLETT-PACKARD COMPANY
P.O.BOX 39
BOISE, IDAHO 83707

Cache, a commonly used term in today's computer industry, can be described as a place dedicated to data storage that improves the performance of data movement. Cache has become a cost-effective solution for optimizing disc performance. This optimization is a result of the relationship between the cache and the access patterns of the host.

Traditionally, disc drive designers have designed and tested disc products as if all sectors of the disc had an equal probability of being accessed by the host, but this assumption of random data access is not always accurate. In fact, the access patterns of HP 3000s are usually not random.

Certain areas of the disc are accessed over and over (system directories, for example) while other areas may not be accessed for long periods of time. This means that many requests are satisfied by disc accesses within the same proximity or locality. It is for this reason that a "cache" can provide a significant performance benefit. Since cache contains frequently accessed data, the more localized or sequential the data requests are the higher the probability requested data are contained in cache. The more requests that can be serviced through cache, or the higher the number of hits to the cache, the greater the performance gain. With this in mind, Hewlett-Packard has developed two types of disc caching solutions for use with the HP 3000 systems.

In-memory cache, referred to as MPE disc cache, is a portion of the CPU's main memory allocated to hold frequently accessed data from disc. This enables future accesses to this data to be serviced from "cache" or memory as opposed to a physical access of the disc. For example, when a disc read is performed, the information is recorded in the cache. If the same information is requested again, the data block is moved from the cache rather than incurring the penalty of a complete disc read cycle. Since memory is considerably faster than a physical disc access, performance is improved. At the same time, the financial investment is minimized since only a portion of main memory is dedicated to cache.

With disc controller cache, the function of in-memory cache is moved out of the CPU and into the disc controller. Dynamic random access memory (DRAM) is added to the disc controller to store information and respond to data requests without incurring the overhead of mechanical positioning. The disc's controller executes the software to manage the cache, freeing up valuable CPU cycles and main memory for other activities. This makes disc controller cache an attractive caching solution, especially for systems that are CPU- or memory-bound. As a result, Hewlett-Packard has developed controller cache for its high performance, high capacity disc drives, the HP 7933H, 7935H, 7936H and 7937H.

The HP 7933 and 7935 disc controller cache products (HP 7933XP and 7935XP) contain one megabyte of dynamic RAM for read cache. This cache stores the last 255 4096 byte

active areas of the disc so future accesses of these same areas can be provided to the host without physical reads actually occurring. This provides a higher level of disc performance over the standard HP 7933H and 7935H drives and can contribute to greater overall system efficiency in the right application.

The HP 7936 and 7937 disc controller cache products (HP 7936XP and 7937XP) have two megabytes of dynamic RAM for read cache that stores the last 440 4096 byte active areas of the disc. In addition, a 4096 byte, single transaction, non-volatile write cache is provided that allows immediate response to the host for write requests of 4096 bytes or less. These caches, coupled with the high speed disc controller, provide greater overall system efficiency to a much broader range of environments.

As part of the development of disc controller cache, extensive testing was performed to determine the optimal memory size for read cache operating on HP 3000 systems. Using a variety of real customer environments, this modeling has shown that the biggest performance gain is seen in the first two megabytes of cache with a cache domain size of 4096 bytes. (A size of 4096 provides the greatest efficiency based on HP 3000 I/Os.) Typical read hit ratios for a two megabyte cache are around 70 to 75%. (The percentage of requests that can be satisfied from cache is commonly referred to as the hit ratio.) As the cache is increased beyond two megabytes, the incremental gain becomes less. In fact, when cache is increased to four megabytes, less than 6% additional hit rate to cache is realized. This is because the most frequently accessed data is likely to already be in cache. Increasing the cache size beyond this point serves only to hold infrequently used data. Thus, there is little incremental benefit to increasing the cache size beyond two megabytes.

Read Cache

The implementation of read cache is based on two basic principles: 1) Store requested data with the assumption it may be requested again. 2) Store more data than requested in anticipation of future accesses being "near" a previous access (this is the case for a sequential read). These fundamental methods have been incorporated into the design of all Hewlett-Packard disc controller cache products, the HP 7933XP, 7935XP, 7936XP, and 7937XP, so they all handle read transactions virtually identically

When the host requests a read that is 4096 bytes or less, a search of read cache takes place to determine if the entire request can be satisfied from one cache domain. If the request can be satisfied from one cache domain, it is considered a read hit and a physical access of the disc is not necessary. If more than one domain contains the requested data, the first domain found is used. If the request cannot be satisfied from one cache domain, it is considered a read miss and a physical read takes place beginning at the start address requested by the host and encompassing a full 4096 byte block. This block is then stored in cache. Any domains that contain overlapping data are not flushed from cache until a write of that data is requested.

When a read request is greater than 4096 bytes, a cache search does not take place. The request is satisfied by a physical access of the disc. Since the average transfer size on typical HP 3000 systems is 1024 bytes, the requests that go uncached because of this size limitation are very few. (Customer data indicate that the average size of disc reads and writes on the HP 3000 is around 1 kbyte with very few in excess of 4 kbytes.) In fact, this

prevents the disc drive from trying to cache data which is used only once such as system backups (STORE); caching in this environment can actually degrade performance.

If a new cache domain is needed to satisfy a read request, the least recently used domain (LRU) is made available for the transfer. In this scheme, when a page is referenced for the first time, it is placed on the top of a push down stack. If the stack is full, the page at the bottom is removed. Each time a page in the stack is referenced, it is put on the top of the stack and all pages in between the top and the referenced page's former position are pushed down. Although this strategy is more complex to implement, it is significantly more efficient than other cache replacement algorithms, such as first-in-first-out (FIFO), where the first page brought in becomes the overlay candidate when a new page is needed.

Write Cache

Unlike reads, the manner in which writes are handled differs significantly between the HP 7933XP and 7935XP, and the HP 7936XP and 7937XP. The HP 7933XP and 7935XP do not provide immediate response for writes. They execute a "write through" cache, whereas the HP 7936XP and 7937XP provide write cache capabilities that allow immediate response for writes of 4096 bytes or less. This capability complements the existing read cache and together these caches provide a noticeable performance benefit.

HP 7933XP and HP 7935XP

When a write of 4096 bytes or less is issued to an HP 7933XP or HP 7935XP, read cache is searched to identify all domains that overlap the area to be written. If the write request is contained within one read cache domain, it is considered a write hit. The data block is physically written to disc while the target read cache domain is updated and retained. If any read cache domains overlap the area to be written, they are flushed at this time. As soon as the physical write is complete, the drive informs the host the write has been performed.

If the requested write does not reside completely within one read cache domain, it is considered a write miss. As soon as the cache search is complete, a physical write is initiated and any overlapping domains are flushed. At completion of the physical write, the drive notifies the host the transaction is finished.

The HP 7933XP and 7935XP do not cache writes greater than 4096 bytes. A search of read cache takes place and overlapping domains are flushed to ensure old data blocks are not kept. The requested write is then physically posted to disc. The host is informed the transaction is complete as soon as the physical write is finished.

HP 7936XP and HP 7937XP

When the host requests a write of 4096 bytes or less to an HP 7936XP or HP 7937XP, a search of read cache takes place to determine if the data block to be written is contained within one read cache domain (overlapping domains are not identified at this time). If this condition is met, the request is considered a write hit and the data block is posted to the 4096 byte write cache while the target read cache domain is updated. As soon as the read cache domain is updated, the drive sends a status report to the host signaling the completion of the write. The data block in read cache is then posted to the disc as a

background write. During this time, a search of read cache takes place to identify and flush any domains that overlap the area to be written.

If the requested write is not contained within one read cache domain, it is considered a write miss. The data block is transferred to the 4096 byte write cache. At the same time, the data block is posted to the drive's 32 kbyte non-cache, circular buffer. As soon as the circular buffer is updated, the drive issues the completion status. The data block is then posted to the disc surface from the circular buffer as a background write. Concurrently, overlapping domains within read cache are identified and flushed.

The HP 7936XP and 7937XP do not cache writes greater than 4096 bytes. Requested writes of this size are physically posted to disc. The drive sends a completion status to the host when the physical write is finished. As soon as the completion status is sent to the host, a search of read cache takes place and overlapping domains are flushed to ensure old data blocks are not kept.

To ensure data integrity with the use of write cache, a battery backup with a life span of approximately ten years is provided. In the event of a power interruption, this backup ensures data remain intact. If a power failure occurs before a write is cached, the host is responsible for the transaction; whereas the drive is responsible if power is interrupted after a write is cached.

In addition, advanced error detection capabilities exist within write as well as read cache, to ensure data remain intact and to warn the drive if problems arise. For example, if a single bit error is detected in cache memory, it is corrected. If a double bit error is detected, it is logged and the drive automatically disables the cache function.

Controller Cache and Physical Disc Performance

The primary effect of disc controller cache is to reduce disc access time. Disc access time is defined as the amount of time required for a disc drive to complete a transaction or I/O. The disc access time of a drive with controller cache is the same as a non-cached drive in that both have controller overhead, seek time, latency time, and data transfer time. (Seek and latency are terms associated with the mechanical positioning of the head to the target sector.) The difference is that with a cached drive the seek and latency, as well as the time for the controller to verify that the head is on track, are often eliminated from the transaction (that is, read hits). Since seek and latency comprise the largest portion of a disc transaction, this serves to significantly reduce the amount of time spent by the drive processing a transaction.

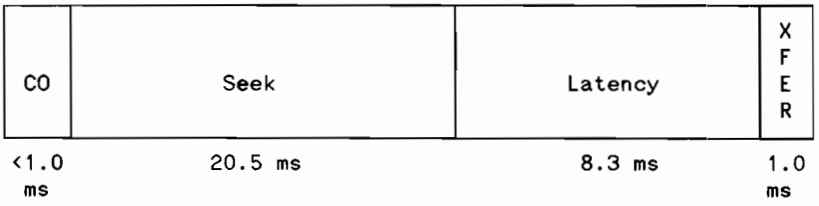
It is important to keep in mind, however, that any cache system imposes some overhead. This overhead is incurred when cache is searched and the requested data is not found and the data has to be read from disc anyway. Also, reading more data than the user requests adds additional overhead if these data are never accessed. The cache overhead of the HP 7936XP and 7937XP is significantly lower than that of the HP 7933XP and 7935XP, while both are only a minute portion of the entire transaction. In most cases, the elimination of the physical seek and latency from many transactions more than offsets any additional overhead incurred.

On average, 1 ms is added to search cache on the HP 7933XP and 7935XP, whereas about .3 ms is added on the HP 7936XP and 7937XP. The HP 7933XP and 7935XP incur this overhead for all reads less than or equal to 4096 bytes and all writes regardless of size. The HP 7936XP and 7937XP incur this overhead for reads less than or equal to 4096 bytes. The only search time incurred for writes on the HP 7936XP and 7937XP is for determining whether the request is a write hit.

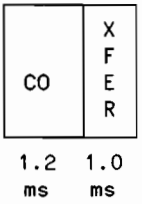
Any read requests that are 4096 bytes or less that cannot be satisfied from cache incur an additional 0 to 4 ms on the HP 7933XP and 7935XP and 0 to 2 ms on the HP 7936XP and 7937XP. This is necessary to read the extra data from disc to fill a 4096 byte cache domain (approximately 1 ms per extra 1024 bytes for the HP 7933XP and 7935XP and .5 ms per extra 1024 bytes for the HP 7936XP and 7937XP). Since the average read request is approximately 1024 bytes, this overhead usually averages about 3 ms for the HP 7933XP and 7935XP and 1.5 ms for the HP 7936XP and 7937XP. In a typical HP 3000 environment, this overhead is only realized for an average of 20 to 30% of the requests. The higher the read hit percentage in cache, the less overhead incurred in this scenario.

The following drawing provides a transaction time component breakdown of a non-cached disc transaction and a controller cached disc transaction. This example clearly shows the positive effect controller cache has on disc transaction time. A request that would normally take approximately 30.8 ms to process only takes 2.2 ms to process. (The numbers listed below represent average times in milliseconds for the HP 7936H and 7937H and the HP 7936XP and 7937XP drives respectively. Exact times vary from transaction to transaction.)

HP 7936/37H Non-Cached Disc Transaction Time Component Breakdown



HP 7936/37XP Controller Cached Disc Transaction Time Component Breakdown



CO: Controller Overhead XFER: Transfer of Data

NOTE: Drawing is not to scale.

Additionally, the implementation of write cache on the HP 7936XP and 7937XP provides the capability of reducing disc access times, as perceived by the host, for writes of 4096 bytes or less. As far as the host is concerned, a disc transaction is complete as soon as the completion status is received from the disc. Since the HP 7936XP and 7937XP send the completion status for a cached write as soon as the data block is buffered, the seek, latency and write times are effectively eliminated from the total transaction time. Although the physical write must eventually be performed, it can now be processed in the background.

The real advantage of caching writes is that other activities can be performed in parallel with the background writes. For example, any read requests that can be satisfied from read cache are processed while a background write is being performed. This overlapping of read hits and writes provides a significant performance improvement, especially in comparison to a drive that does not utilize write cache. In fact, a drive with write cache is capable of processing one write as well as many read hits in the same amount of time an uncached drive can process one write. This provides a considerable time savings.

This example is certainly a best case scenario. However, write cache always provides a benefit as long as the write requests do not exceed 4096 bytes. (Customer data show that the majority of writes are less than 4096 bytes.) The exact benefit is determined by the sequence and timing of reads and writes generated by the host and whether or not those transactions can be serviced through cache. For example, a cached write can be immediately followed by another write that qualifies to be processed through write cache. In this particular situation, the advantage of write cache is not as noticeable. This is due to the blocking condition during the processing of the second write.

As soon as the first write is cached, the host can generate the second write command. However, before the second write can be cached, the first write must be physically posted to disc. This is necessary to ensure data integrity. As a result, the second write incurs some wait time. Even with this additional wait period, the amount of time required to process both writes is less than it would be on a drive without write cache. This is due to the time savings realized while the host generates the second write. The generation of the second write command, as well as much of the controller's overhead in receiving and decoding the command, actually takes place in parallel with the first background write. As soon as the first write is finished, the second write is cached and immediate response given.

The exact amount of host time saved varies with the system and the application. The longer the host generation time, the closer the disc drive is in completing the first write. This implies that slower host computers receive more of a benefit in this situation than faster hosts. At a minimum, write cache provides a reduction in write access time between 4% (Series 70) and 20% (Series 37), depending upon processor speed.

A similar situation occurs whenever a cached write is immediately followed by any request that requires a physical access of the drive (i.e., a read miss). Since the actuator or head positioner is needed to post the first background write to disc, the second request is forced to wait for this process to complete. As soon as this is completed, the second transaction is performed. The extra wait period, combined with the normal processing time, increases the total transaction time for the second request. Although there is an increase in access time for the second transaction, there is still an overall time savings as a result of the first

write being cached. The amount of time required to process both transactions is always less than it would be for a drive that does not employ write cache.

The reduction in access time provided by disc controller cache serves to increase disc throughput. Disc throughput is defined as the number of transactions or I/Os processed per second. The exact disc throughput gain achieved varies with the application. Typically, there are some requests that cannot be serviced through cache.

The actual performance of a drive with only read cache is dependent upon the read/ write ratio as well as the percentage of hits to read cache. If the host does not have a favorable read percentage (at least 50 to 60%), a drive with only read cache has a hard time providing performance benefits. This is because the disc drive can only help reads and slightly degrades writes. Thus, the higher the percentage of hits to read cache, the greater the achieved I/O rate. In a typical HP 3000 environment, a minimum read hit percentage of 17.5% must be realized on the HP 7933XP and 7935XP to equal the average access time of the HP 7933H and 7935H, assuming 75% reads. In reality, there are very few HP 3000 environments where this measure cannot be easily exceeded.

In comparison, a drive that is capable of caching both reads and writes provides a performance benefit to each. This means actual performance is not dependent upon the read/write ratio. Instead, the timing between I/Os is more important. Read hit percentages, however, are still important since they are an accurate measure of how effective the read cache is. Because of improved controller efficiency, a minimum read hit percentage of less than 10% is needed to reach the break-even point on the HP 7936XP and 7937XP relative to the HP 7936H and 7937H, regardless of the read percent.

The following table provides disc access times and disc throughput figures for cached and non-cached drives. All numbers are averages based on a 1 kbyte transfer size. These figures represent fundamental disc performance without taking into consideration specific system attributes. The impact of controller cache is based on a typical HP 3000 environment with a read hit rate of approximately 70% and a read percentage of approximately 70 to 75%. It is important to keep in mind that the access time of the HP 7936XP and 7937XP varies depending upon the timing of I/Os. These figures represent access times based on maximum I/O rates.

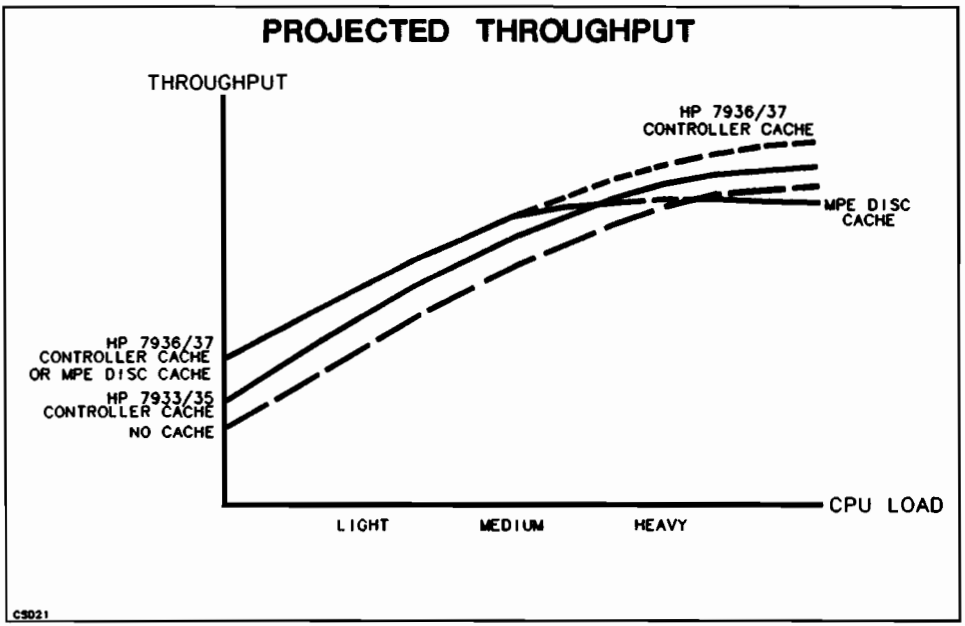
	HP 7933H HP 7935H	HP 7933XP HP 7935XP	HP 7936H HP 7937H	HP 7936XP HP 7937XP
Controller Overhead	3.5	4.5	<1.0	1.2
Seek Time	24.0	24.0	20.5	20.5
Rotational Latency Time	11.1	11.1	8.3	8.3
Transfer Time	1.0	1.0	1.0	1.0
Controller Cache Impact		-16.6		-15.9
Total Transaction Time	39.6	24.0	30.8	15.1
I/Os Per Second	25.3	41.7	32.5	66.2

Controller Cache and System Performance

Controller cache provides more than just higher levels of disc performance over the standard drives. By moving the caching function out of the CPU and into the disc, valuable CPU cycles and memory can be freed. These resources can then be devoted to other activities. This, combined with faster disc transactions, provides greater overall system efficiency in the form of system throughput and response time in the appropriate application. (System throughput being defined as more transactions completed in a given time period.) Since MPE disc cache can provide similar benefits, it is important to examine each caching scheme to determine which provides the greatest benefit.

A system's level of CPU utilization has a large impact on the potential performance gains either caching system provides. The following graphs show the effect of MPE disc cache and controller cache on system throughput and response time, based on CPU load. The throughput and response time of a non-cached system is also included for comparison (These graphs are based on data gathered from benchmarks and alpha and beta testing.)

System Throughput



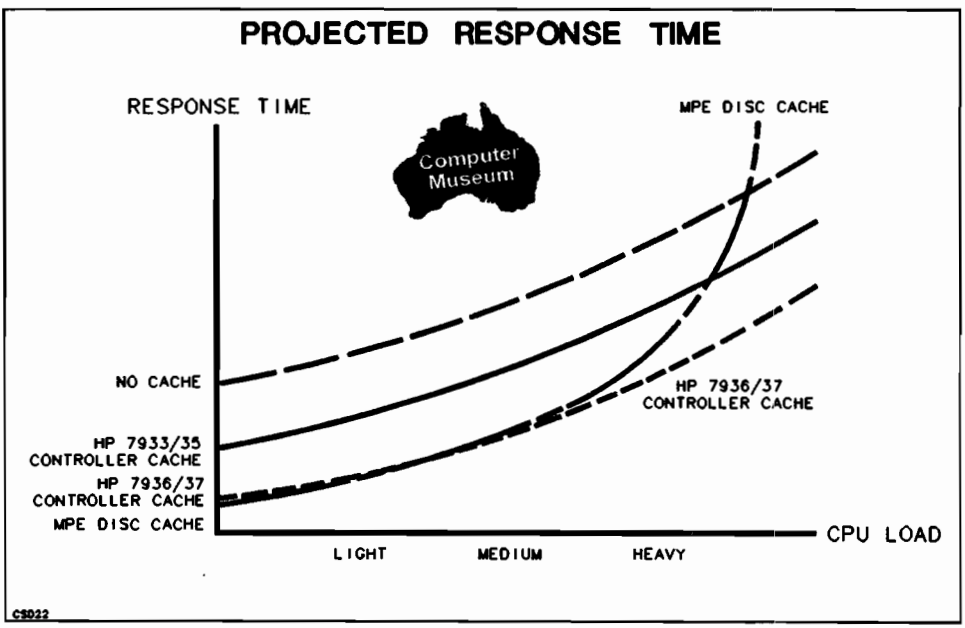
A lightly loaded, non-cached system (less than 75% CPU utilization) benefits from either MPE disc cache or disc controller cache. This type of environment is likely to be more I/O-bound than CPU-bound. All caching schemes greatly reduce the I/O bottleneck. MPE disc cache has an advantage over HP 7933 and HP 7935 controller cache because there is no controller overhead involved when reading directly from memory. In comparison, the

reduction in overhead and ability to cache writes, allows the HP 7936XP and 7937XP to perform as well as MPE disc cache in this case.

As the CPU load increases to a moderate level (75 to 90% CPU utilization), the throughput of a system with MPE disc cache is affected. The management of MPE disc cache must now compete for fewer available CPU cycles. MPE disc cache continues to be faster than a non-cached system at this stage and more than likely faster than the HP 7933 and 7935 controller cache. There is however, greater potential for the HP 7933 and 7935 controller cache to become more effective. In fact, at this point it begins to provide the capability of leveling out CPU peaks and valleys. On the other hand, the HP 7936 and 7937 controller cache continues to perform as well as MPE disc cache until the CPU load reaches approximately 80%. At this point, the HP 7936 and 7937 controller cache begins to show an improvement above that of MPE disc cache.

When a system reaches the stage where CPU load is heavy (90% and above), the impact of MPE disc cache on system throughput can become negative. In extreme situations, the system can actually perform more efficiently with MPE disc cache turned off. In this environment, the HP 7933 and 7935 controller cache provides a noticeable benefit, especially when MPE disc cache logical read/write ratios and read hit percentages are high. This indicates that a good deal of I/O is being eliminated and that CPU cycles and memory used for managing cache can be freed for other activities. The HP 7936 and 7937 controller cache provides the greatest benefit in this environment without relying on specific read/write ratios.

Response Time



All caching schemes decrease response times for a lightly loaded, non-cached system. The HP 7933 and 7935 controller cache is at a slight disadvantage because it cannot respond to I/Os as quickly as MPE disc cache. On the other hand, the HP 7936 and 7937 controller cache offers response times nearly equivalent to that of MPE disc cache. MPE disc cache has a slight advantage because cache search times are not as great when CPU loads are light. This advantage, however, is so slight that it may be difficult to perceive from an end user perspective.

The response times of the HP 7933 and 7935 controller cache continue to be at a disadvantage at the moderate CPU utilization level. Although throughput benefits may be realized relative to MPE disc cache, response times are often sacrificed. In comparison, the HP 7936 and 7937 controller cache can offer system throughput gains relative to MPE disc cache and maintain equivalent response times.

At the heavy CPU utilization level, response times can actually be negatively affected by MPE disc cache. Both controller caches provide improved response times in comparison to MPE disc cache, while maintaining higher system throughput levels.

Controller cache continues to significantly outperform a non-cached system, from both a system throughput and response time perspective, until the CPU load is increased to the point where the system is so CPU-bound that I/O is no longer a factor. At this point, a CPU upgrade may be the best solution. It is important to note, however, that the significant improvements seen from the HP 7936 and 7937 controller cache can extend the life of a system beyond the point where it would normally become CPU bound.

The potential growth of the system should also be a key consideration. Although the current load of a system may not be ideal for controller cache, it is important to consider the potential growth of the system. Initially controller cache may not provide noticeable benefits, i.e. faster processing times. It can, however, be a valuable investment for the future by allowing more processes to be performed in the same amount of time. This means additional users can be added to the system without degrading response times. This is especially true with the HP 7936 and 7937 controller cache. The ability to perform as well as or better than MPE disc cache in most environments, allows this product to provide both system throughput and response time improvements regardless of CPU utilization levels.

Monitoring Controller Cache

Controller cache performance statistics can be reviewed through the use of the CS80UTIL program (A.03.03 provides full controller cache support). Through CS80UTIL, a general idea of how well controller cache is functioning on each disc can be obtained. By issuing the CACHE STATS command from within CS80UTIL, the following information is listed for a specified disc:

Cache Status - tells whether controller cache, read and write if applicable, is enabled.

Cache Page Size - displays the size in bytes of one cache page or domain. The value is always set to 4096 bytes for the HP 7933XP, 7935XP, 7936XP and 7937XP.

Number of Pages - displays the number of 4096 byte pages or domains that constitute read cache. This value is always 255 for the HP 7933XP and 7935XP and 440 for the HP 7936XP and 7937XP.

Number of Reads - indicates the total number of reads that have been requested.

Number of Read Hits - the total number of requested reads that have been satisfied from read cache. This value divided by the number of reads results in the read hit percentage.

Number of Writes - total number of writes that have been requested.

Number of Write Hits - total number of requested writes where data in read cache was updated. This number divided by the total number of writes results in the write hit percentage.

Number of Write Cache Hits - total number of writes that were processed through write cache (total number of writes that were 4096 bytes or less). This number divided by the total number of writes results in the write cache hit percentage.

Read Hit % - percentage of total reads that could be satisfied from read cache.

Write Hit % - percentage of total writes where data was saved in read cache.

Write Cache Hit % - percentage of all writes that were processed through write cache (percentage of writes that were 4096 bytes or less).

Read % - Percent of all requests that were reads.

These controller cache statistics are cumulative and can be reset to zero by issuing the RESET STATS command, by enabling cache after it has been previously disabled, or by performing a system warmstart. The drive resets them to zero automatically during a power-on, and if there is an overflow of any of the 4 byte integers assigned for each statistic count.

Controller Cache Recommendations

Both controller cache products offered by Hewlett-Packard can provide a performance boost to many HP 3000 systems. The actual performance gain is very system dependent. In general, any system that does not utilize MPE disc cache sees an immediate benefit from the HP 7933 and 7935 or the HP 7936 and 7937 controller cache as long as most data transfers are 4096 bytes or less and the CPU is not so CPU-bound that I/O is not a major factor. If a system without MPE disc cache is experiencing very high CPU utilization levels, a CPU upgrade may be more appropriate than a controller cache upgrade.

If a system already uses MPE disc cache, the benefit from controller cache is more dependent upon the specific environment. Because of fundamental design differences between the HP 7933XP and 7935XP and the HP 7936XP and 7937XP, the specific conditions necessary to realize a performance benefit vary depending upon which controller cache is used.

HP 7933 and 7935 Controller Cache

In order for the HP 7933 and 7935 controller cache to provide a performance benefit beyond that offered by MPE disc cache, the following qualifications should be met.

* CPU utilization > 90% with MPE disc cache on

  The HP 7933 and 7935 controller cache provides the most benefit in those environments where CPU utilization is heavy and adequate resources are not present to utilize MPE disc cache effectively. By disabling MPE disc cache and implementing the HP 7933 and 7935 controller cache, valuable CPU cycles and memory can be freed for other activities. The actual performance gain from this depends upon the extent to which MPE disc cache is prohibiting other activities from being performed.

* 3:1 read/write ratio for MPE disc cache

  A high read to write ratio is necessary in order to realize maximum benefits from the HP 7933 and 7935 controller cache, because only the access time of reads is reduced. The access time of writes is actually slightly degraded.

* 70% or greater read hit percentage for MPE disc cache

  The more localized data requests are, the greater the performance benefit from any cache system. MPE disc cache read hit percentages indirectly reflect the level of data locality a system is experiencing. Thus, the higher the MPE disc cache read hit percentage, the greater the probability the HP 7933 and 7935 controller cache can provide a benefit.

* Balance of files over the discs

  A good I/O balance across discs is necessary in order for the HP 7933 and 7935 controller cache to perform well. If there is a great deal of I/O activity on one disc and very little on another, the benefit is not the same for each of these drives.

* Data transfers of 4096 bytes or less

  The HP 7933 and 7935 controller cache domain size is 4096 bytes. Any reads greater than this size are not serviced from cache. (Customer data show the majority of reads are less than 4096 bytes on the HP 3000.) Any applications that utilize transfer sizes greater than 4096 bytes do not see a benefit from controller cache.

* Number of drives >= megabytes of memory currently used for MPE disc cache

  In order to keep read hits at the same level as that of MPE disc cache, it is important to ensure that the HP 7933XP and 7935XP discs have adequate memory. In general, the number of controller cached drives should be equal to or greater than the current amount of main memory devoted to MPE disc cache. For example, if a system currently allocates 5 megabytes of memory to MPE disc cache, at least 5 HP 7933XP or 7935XP drives should be placed on the system (each HP 7933XP and 7935XP drive has 1 megabyte of memory).

HP 7936 and 7937 Controller Cache

The ability of HP 7936 and 7937 controller cache to provide immediate response to writes of 4096 bytes or less, in conjunction with the reduction in controller overhead and the larger memory size, enables this cache to fit a much broader range of environments without relying on specific positioning requirements. In certain environments, more benefit is realized than in others. The following information outlines several important factors to be considered in determining whether the HP 7936 and 7937 controller cache can provide a performance benefit for a particular system.

* Single-Threaded Batch Environment

  In predominately single-threaded batch environments (only one batch job executing at a time), MPE disc cache may provide better performance than the HP 7936 and 7937 controller cache. In these situations, the MPE disc cache search time is probably less than it would be for a multi-user or multi-batch environment. In comparison, the time to search the HP 7936 and 7937 controller cache remains constant regardless of the level of system activity. This can give MPE disc cache a slight advantage in this scenario. This advantage, however, is not guaranteed. Depending upon the speed of the processor, there may be single-threaded batch environments that see improvement from the HP 7936 and 7937 controller cache.

* CPU Utilization

  The HP 7936 and 7937 controller cache is not as sensitive to CPU utilization levels as the HP 7933 and 7935 controller cache. In fact, the HP 7936XP and 7937XP perform as well as MPE disc cache in most multi-user, multi-batch environments regardless of CPU levels. This means the HP 7936XP and 7937XP have the capability of returning CPU cycles and memory for other functions while at the same time maintaining a performance level at least equivalent to that of MPE disc cache for most HP 3000 environments. In addition, the HP 7936 and 7937 controller cache provides significant performance improvements that exceed those offered by MPE disc cache when CPU utilization levels reach the moderate to heavy range.

* Read/write ratio

  Since the HP 7936XP and 7937XP can provide immediate response to writes of 4096 bytes or less, the read to write ratio is not as critical a factor in determining whether or not a benefit is seen from these products. However, in heavy write situations, MPE disc cache provides a slight advantage. There are two main reasons for this advantage. First, MPE disc cache is capable of caching multiple writes while the HP 7936XP and 7937XP can only cache a single write transaction at a time. Second, MPE disc cache has the ability to respond to I/Os in front of the I/O system whereas controller cached writes have to pass through the I/O system first.

* Read Hit Percentages

  Read hit percentages are important to HP 7936 and 7937 controller cache, because they are a measure of how effective read cache is. Typically, if read percentages are high with MPE disc cache, they are similar with the HP 7936 and 7937 controller

cache. However, the HP 7936XP and 7937XP do not need to achieve as high a read hit percentage as MPE disc cache in order to provide equivalent or superior performance.

* Balance of files over the discs

  Balancing I/Os across discs is also important to achieve maximum performance levels with the HP 7936XP and 7937XP.

* Number of drives >= megabytes of memory currently used for MPE disc cache

  With the HP 7936XP and 7937XP a correlation still exists between the amount of controller cache memory and the amount of main memory allocated to MPE disc cache when it is in use. Even though the HP 7936XP and 7937XP offer two megabytes of memory, the greatest return is seen from the first megabyte. Therefore, in order to achieve maximum performance benefits, the number of HP 7936XP or 7937XP drives placed on the system should be equal to or greater than the amount of main memory devoted to MPE disc cache when it is in use.

* System Backups

  MPE disc cache has an advantage over HP 7936 and 7937 controller cache when MPE STORE or SYSDUMP is utilized for system backups. Although the data portion of the backup is not cached in either caching scheme, the reads and writes of file labels are. Since MPE disc cache can cache many more file labels than the HP 7936 and 7937 controller cache, the hit to the cache is much greater, giving it a performance advantage.

Summary

Cache has become an effective means of improving disc performance. All Hewlett-Packard disc caching schemes, MPE disc cache, HP 7933 and 7935 controller cache and HP 7936 and 7937 controller cache, optimize disc access times. This in turn, contributes to greater overall system efficiency. The exact performance benefit realized from any disc caching scheme is dependent upon the specific environment. Traditionally, MPE disc cache has provided performance benefits to the broadest range of environments. With the introduction of the HP 7933 and 7935 controller cache, those environments at a disadvantage with MPE disc cache (that is, heavy CPU loads), were able to receive the benefit of a cache while offloading CPU resources. Now, with the enhancements that have been incorporated into the HP 7936 and 7937 controller cache, the benefits of MPE disc cache can be matched and even exceeded in most environments.

Dictionary/3000
Home Grown Utilities

by

Matt Avery
Bancroft, Avery & McAlister
601 Montgomery Street, Suite 900
San Francisco, California 94111

Dictionary/3000 is a product that claims to be "a central repository for information about an organization and its data processing environment". This statement is just general enough to be accurate, but the utilities provided by Hewlett Packard do not allow users to access and manipulate all of the information that could be contained in their data dictionaries. In some cases these utilities are so tedious and frustrating to use, that they simply produce inaction. Our solution to this problem was to develop some additional utilities which allow us to update and maintain our Dictionary/3000 information with a minimum of effort, and a maximum of productivity.

This is NOT an attempt to sell some product to HP 3000 users, it is simply an attempt to provide other frustrated Dictionary users with hope and insight into how they can more effectively use their existing Dictionary/3000 systems. It is certain that there are other users who have experienced the same frustrations we have, and would probably benefit equally from our techniques. These utilities and their associated procedures have allowed us to increase the usefulness of our Dictionary/3000 functions tenfold, and to begin to use the Dictionary for things that previously would have been extremely difficult, if not impossible, to do.

Our primary goal in developing these Utilities was that they had to maintain the internal integrity of the Dictionary so that the functionality of the HP Utilities would not be affected. Many of the features of the Dictionary need not be duplicated by a user, but there should be no reason why we should be frustrated by an incomplete product when we have the ability to provide the necessary extensions.

The Dictionary is a complex beast and deserves a large measure of respect. Its structure can allow some impressive improvements in overall productivity within an organization. The problem with the product as a whole is that the

utilities provided to users lack the depth and scope needed to access the complex variety of information stored in the Dictionary. As the introduction of System Dictionary looms on the horizon with its greater scope, power and flexibility, what new challenges will current Dictionary/3000 users be faced with ? Will there be new utilities that resolve the old problems, or simply new problems to face ? An understanding of the problem is what will be attempted in this paper, and some attempt made to demonstrate the types of solutions that are available.

Background

Our company is in the business of providing legal services, and our computer system is primarily used to handle our Time and Billing, and Financial Accounting functions. Our Time and Billing software was written by our own staff, and all other applications were purchased from third parties (including HP). We have a staff of one Manager, one Programmer/Analyst, and three Data Entry Operators. This staff supports approximately 100 users in a primarily PIPO (Paper In, Paper Out) batch environment.

Our previous Time and Billing system was written in Fortran and Assembler and was running on a Sperry Univac V77 minicomputer. Maintenance of this system was a constant nightmare, since keeping track of versions of source code, job streams, object module libraries, etc. was a full time job. We had developed a documentation system, which over time consisted of reams of printed and hand written documents. This documentation was maintained by several persons, each with their own differing standards for organization.

When we first began the process of converting our Time and Billing Software to Transact, we were excited by the prospect of having the structure and documentation of our new system contained within this Data Dictionary where it would be easily maintainable and reportable. Well, the euphoria didn't last very long before we began experiencing some of the awkward maintenance problems associated with the Dictionary.

The primary problem we encountered was difficulty of maintaining the description within the Dictionary. The line by line method of maintenance was both tedious and frustrating, and led to the eventual neglect of the function completely. Another problem, again with the description, was that the Dictionary reports that we found most useful

required that the description be linked to the association between a file and its elements. When an element appeared in multiple files, the description had to be duplicated, and most of the time inconsistently. When you wanted to edit some description, you had to proceed on a potentially long and confusing quest to find the specific entity, association, or relationship to which the description was linked, and then if you had the conscience to do so, find all other references to the element and make the same changes.

We had one designer and three programmers, each of whom had their own information to contribute to the documentation for the system. The situation got to the point where each person kept a copy of the Dictionary "Show File" report covered with their own handwritten annotations. Reports were printed only when the databases were substantially changed structurally, but the documentation for the dictionary existed only on paper. There was very little sharing of critical information which caused tremendous delays and problems in the development cycle. We were committed to using Dictionary/3000 for the maintenance of our data structures since we could find no other product that would provide us with the features we desired and the degree of integration with the Rapid products. We had no desire to hire an additional person whose specific purpose was to manage the Dictionary.

When our staff size reduced to its current state, the need for accurate and dependable documentation grew to the point where it became the critical issue. The prospect of developing new systems held frightening consequences since we couldn't efficiently manage the information related to these projects with the tools at our disposal. Even the maintenance of programs became a tremendous chore that generated tons of paper notes waiting to be added to the description in the Dictionary.

At one point we had hired a contract Transact programmer to aid in finishing our conversion more rapidly. The programmer was given written programming specifications and a listing of the Dictionary. The two documents complemented each other, but since the Dictionary description was incomplete, inaccurate and inconsistent the contractors work product was very poor. No fault on the programmer, the programs were a clear demonstration of just how bad the chaos within the Dictionary was.

We came to the conclusion that our work load was only going to increase and become increasingly complex as more applications and user services were added to the system

requiring more and more control and management of both human and machine resources. We decided that something had to be done to ease our problems, and make each individual more productive when using the Dictionary. The solution we decided upon was to write our own Dictionary/3000 Utilities.

Solution

Our first task was to overcome the natural fear of treading where Non-HP personnel have never gone before. This apprehension was overcome by simply creating a test copy of our Dictionary and experimenting with our speculations about it. As a result, we quickly lost our fear and began to develop a genuine desire to expand the use of this tool. We located several informative articles in HP User publications which provided a great deal of insight into the internal structure and functioning of the Dictionary. Once we felt that we had the proper understanding of the task being undertaken (and a reliable backup), the process began.

We first created a wish list of what functions we felt would make the Dictionary more useful and functional. We determined that the primary area of interest was the maintenance of description within the Dictionary. We initially developed two groups of programs to provide us with the functionality we desired. The first group consists of utilities which allow us to remove existing description from the Dictionary, format it for transfer to a Micro based Word Processor where it is edited, and then transferred back into the Dictionary. The other group of utilities consists of new reports which present the information in the Dictionary in a form that is more useful for our purposes. Several of the procedures actually update the Dictionary, but most simply report about entities, relationships, and associations.

Since our primary source of frustration with the Dictionary was limited functionality of the tools provided by HP to maintain the description, our first utilities were procedures and programs which allow us to maintain the description outside of the Dictionary itself. We use a word processor on HP-150's as our primary documentation tool, so we decided to develop a method where we could use the word processor to maintain the description we wanted and then load that description back into the Dictionary in bulk. This function was the most complex to implement because of the way that Dictionary/3000 internally maintains

description, which can be quite complicated and if corrupted could cause considerable grief.

We determined that the all of our existing description was specific to Entities; FILEs, ELEMENTs and PROCEDUREs. However, the descriptions we most frequently relied on were linked to the FILE-ELEMENT relationship since the HP report that we found most useful provided that description only. Our first task was to re-organize our Dictionary description so that it was all associated with Entities, and not relationships or associations. To do this, a program was written to unload the existing description from the Dictionary into a file which could be used by our Micro Word Processor. Once this was done, we cleared all description from the Dictionary, reset all internal pointers to reflect the non-existence of description, and then prepared to reload this voluminous mass of data. Once the description had been re-arranged, unloaded, and edited it was reloaded into the Dictionary by our newly created upload utility.

Our next task was to write the new reports to provide us with the information that we wanted to see in the form that we wanted to see it in. We developed reports which keyed on each of the Entities of the Dictionary, FILEs, ELEMENTs and PROCEDUREs (the others we do not use but could be fit into this scheme also). Each of these reports lists the elements associated with the primary entity showing attributes, description, etc. For each element, we also show relationships with other elements, and associations with other files and procedures. We added an Image dataset to our production database which contains optional values for any element in the Dictionary. Our reports also provide a list of these options whenever the key element is included on a report. These reports provide us with the ability to cross reference information about an entity and its relationships regardless of which report we use.

Once these procedures were operational, they became an integral part of any new development that we attempted, as well as maintenance of existing systems. We had to develop procedures to ensure that all new information created was automatically incorporated into the new scheme. With the new ease of maintenance and reporting, the information was always accurate, and easy to locate.

Once the utilities to maintain the FILE and ELEMENT portions of the Dictionary were operational, the "What If's" and "Could We's" began to arise. The primary interest now was to somehow automatically update the Dictionary when a procedure was created or modified, and maintain the relationships between this procedure and elements defined in

the Dictionary. This was an interesting problem, since there was no existing physical link between compilers and the Dictionary except for data element definitions, One Way out from the Dictionary. There was one place where the information existed which could be used to update the Dictionary, the compiler listed the elements used in the program. This list could be read to determine which elements existed in the Dictionary and update the relationship between the compiled procedure and those elements. We use Transact and Fastrans as our primary programming languages, so we developed a program which determines which compiler is being used and decodes the compiler listing accordingly. This program reads a temporary file generated by the compiler containing the necessary item definitions. We have incorporated this program into job streams which are used to compile all programs and therefore automatically updates information in the Dictionary anytime a particular program is compiled.

BRW is an HP product which has had a large impact on how we use the Dictionary. Consequently, we have developed a utility to aid in the documentation and control of BRW reports. The Dictionary is required for use of BRW, and therefore demands increased management because of BRW's interaction with it. We created a version of the Procedure-Element Relationship utility that was specifically designed to document the links between BRW and elements defined in the Dictionary. This was quite interesting to do since there is no compiler listing produced by BRW. The only source of the "item" information that we needed was found in the Source Tables of the Report Specification Listing. A program was created which read this listing and updates the links between a specific BRW report and the elements that it accesses.

Once these procedures had been incorporated into our operations, the potential uses for the Dictionary's ability to track relationships became more clear and usable. We currently use the Dictionary as a true system wide repository of knowledge. There is no limit to the flexibility of uses for Dictionary/3000 once you develop a method which allows you to control it instead of it controlling you.

Description Unload Program:

This procedure was designed originally to provide an easy way to move all of our existing description from the Dictionary into a file which could be edited using a Micro Word Processor. We designed it to have a life beyond the initial conversion of the description since it could at any time generate a file containing the current description from the Dictionary. This program unloads the description on a file by file basis depending on the desires of the operator. Once it has been given a file name, it unloads the description for the file first, then proceeds to process the FILE-ELEMENT chain for that file unloading each element and its description.

The resulting file is equivalent in format and structure to the input file described in the Description Upload Program. The most complicated aspect of this program was determining the simplest file format to use in order to allow easy transfer to the Micro Word Processor.

Description Upload Program:

This utility reads an MPE file which has been created with the following specific format:

```
//ENTITY-TYPE//     ENTITY-NAME
ENTITY LONG NAME

From this point until the next //ENTITY-TYPE// or //END// is
encountered, the program will assume that it is dealing with
description for the Entity noted  above.  The description
can be in any form desired to make it more readable.
//END//
```

The input file can be as long as desired, and may contain any combination of entities. The "//END//" is a special entity which indicates the end of the input file. The only other special entity is "--COMMENT--", where the description following it will be ignored until the next entity is encountered. This is used to document the input file where desired.

The ENTITY-TYPE can theoretically be any entity, relationship, or association that is defined in the Dictionary. The only actual limitation is that the ENTITY-

TYPE must relate to some file that contains a DESCRIPTION-KEY field, and a Long Name.

This DESCRIPTION-KEY is used by the Dictionary to allocate the next key for the LINK-DESCRIPTION and DESCRIPTION-TEXT datasets. LINK-DESCRIPTION is an automatic master which simply links multiple lines of description contained in the DESCRIPTION-TEXT dataset. The DESCRIPTION-KEY for this chain is then also stored in whatever dataset it is logically linked to. This allows the description to relate to any entity, relationship or association whose record contains the DESCRIPTION-KEY.

The DESCRIPTION-KEY stored in DIC-CONTROL is used to determine what the last value assigned was. One capability that this utility provides, which could be considered an improvement, is that DESCRIPTION-KEY is an ever increasing number which represents the last DESCRIPTION-KEY assigned for any entity, association, or relationship in the entire Dictionary. When description is deleted using DICTDBM, this number remains unchanged, so a void is created where an particular DESCRIPTION-KEY value could refer to no description at all. Our Description Upload Utility uses the DESCRIPTION-KEY of DIC-CONTROL when new description is being added, and if the description being loaded is replacing existing description, then it reuses the existing DESCRIPTION-KEY. This works consistently because the only value really affected by the difference in the description is the POSITION value within the DESCRIPTION-TEXT chain of the DESCRIPTION-KEY that is being replaced. When our utility needs to add new description, it follows the same rules as DICTDBM by reading DIC-CONTROL, incrementing DESCRIPTION-KEY, Updating DIC-CONTROL and then writing the new description. When existing description is replaced, it saves the existing DESCRIPTION-KEY, and simply replaces the DESCRIPTION-TEXT chain for the key with whatever new text is added. There is no change made to the entity related to this description. This feature allows for a minimum of damage potential and also reduces the frequency of Dictionary rebuilds to remove the junk numbers from within the description scheme.

The POSITION value in the DESCRIPTION-TEXT dataset serves as a sort field to maintain the order of the lines of description that are added. This item is defined in IMAGE as a K-Type SPL logical value, which means that the effective limit for a single chain of description is probably larger than the entire capacity of the DESCRIPTION-TEXT file. This ability allows for a very large volume of description to be associated with an entity, relationship or association.

Using the Micro Word Processor and this bulk Upload Utility the maintenance of description stored in the Dictionary can be quite easy. There is no specific limit to how much or how little can be contained in a single upload file. This capability allows for easy and rapid modifications to a single dataset, a single entity, a small set of procedures, etc. When programs are changed, the programmer simply submits a file containing the new description, and the Dictionary Administrator uploads the new description into the proper location. Having the creator of the description be the individual who is most familiar with the entity being documented, produces a high level of accuracy. It also adds some extra control by defining responsibilities for documentation more clearly.

Procedure / Element Relationship Program:

This utility updates the Dictionary when procedures are created or modified. It functions in such a way that it is highly improbable that it would corrupt the internal integrity of the Dictionary.

It functions like a human operator manually entering the relationships between a procedure and elements that already exist. Once the procedure has determined which compiler it is dealing with, it parses the compiler name out of the page heading and reads the DATA-PROCEDURE file to determine if the procedure being processed already exists. If it exists, then the DATE-CHANGED is updated, PROCEDURE-LANGUAGE is set to the name of the compiler, IDENTITY-CHANGED is set to the User's name and the record is replaced. If the procedure is new, then a PROCEDURE-NAME is constructed from the name of the procedure and a phrase indicating that it is new. The DATE-CREATED is set, PROCEDURE-LANGUAGE is set to the compiler name, IDENTITY-CREATE is set to the User's name and a new record is written.

Once the procedure is updated, the program deletes the PROCEDURE-ELEMEN chain for this procedure and rebuilds the relationships from the item list provided by the compiler. It fills the DATE-CREATE and IDENTITY-CREATE fields with the compile date and User's name. This list of elements only contains elements that are already defined in the dictionary, so that it doesn't add unnecessary overhead to the Dictionary. One consideration here is that we do not maintain description associated with this relationship, but it is feasible to modify this program to retain any existing description when replacing the chain of elements.

Job stream standards have been developed which require that any program compiled will use this program to update the Dictionary. Any programmer creating a new procedure need only create a Job Stream from a template file provided, and stream the job to compile the program and update the Dictionary.

BRW Report / Element Relationship Program:

Since the introduction of BRW, the need for centralized control of the Dictionary has grown. BRW uses the Dictionary to create its RDIC file which contains its own special data dictionary. This physical segregation makes it particularly important that changes made to Dictionary/3000 be transmitted to the RDIC file.

BRW can be quite tricky when the Dictionary contains child elements, since it ignores the parent elements, and only allows access to the children. This can cause substantial problems when the children relate to an Image Search Item. We created a separate Dictionary specifically devoted to BRW reports. It contains only the minimum information that BRW requires, but no description. This structure added even more complexity to the documentation requirements, so that it became imperative to automatically update the relationships.

One "feature" of BRW is that is automatically re-compiles ALL reports whenever the RDIC file is modified. This "feature" has caused us certain problems because we had no advanced warning of what reports might be affected by the changes to the Dictionary and RDIC. What we needed was a cross reference report showing what BRW report used what elements.

We created a modified version of our Procedure / Element Relationship program which uses the BRW Report Specification Listing as input and parses out the PROCEDURE, PROCEDURE-NAME, and all elements contained in the Source Tables. Any item that exists in the Dictionary and is found in the Source Tables is linked to the report and marked with DATE-CREATE and IDENTITY-CREATE.

Job streams were designed which compile, and list the report and update the Dictionary at the same time. This utility requires a higher degree of control over report designers, because it adds an extra step that must be performed outside of BRW. But given the increase in control and awareness, we found it a small price to pay.

This program began to pay for itself immediately since it allowed us to document BRW reports provided by HP with our Financial Applications System. Since HPFA is something of a mystery in its own right, having the ability to document some of the workings of HP software was a major payoff.

Job Stream Usage:

Another utility that we have added recently is an automatic update of when job streams are used and by whom. This may not seem very useful on the surface, but when job streams pile up as fast and prolifically as they do here, the ability to quickly and easily determine whether or not one should be deleted is quite useful.

This utility has been inserted into the beginning of all job streams and immediately updates the Dictionary for date and identity. This program uses the JOBINFO intrinsic to determine the Job Name, and then locates that procedure in DATA-PROCEDURE. It then either adds or modifies the information for this PROCEDURE, and proceeds with the remainder of the job.

We have found that this utility has increased the consistency and maintainability of our jobstreams substantially.

Self-Documenting Dictionary:

One by-product of our utilities created for BRW, was the discovery that we could make the Dictionary self-documenting. We use DICTDBD to load the definitions of the Dictionary into itself and thereby allow BRW reports to be created to produce Dictionary Reports. This has allowed us to incorporate the Dictionary itself into our documentation system.

Dictionary Reports

These reports have been created to enhance the usability of the Dictionary and satisfy the specific needs of our applications, but the designs should be of use to anyone who uses Dictionary/3000. All of these reports use the same basic format and logic so that there is no confusion generated by the "Which report showed X ?"

situation. All reports are Read Only and therefore can cause no damage to the Dictionary.

File Report:

This report produces a report for any file which exists in the DATA-FILE data set. The information presented in this report is the set of Elements associated with the requested file. The File entity can be a parent, a child, a grandparent, or whatever. The filename that the user supplies is simply the starting place in the genealogy. For instance if the name given is that of a base with related files, then the BASE file information will be reported and then each child file will be reported. If any individual child file is requested, then that is the only file that is reported. If All files are requested, then each file is reported serially.

Within the report the user has the ability to limit the amount of information shown. The report always shows any and all elements associated with the file, but selectively shows the description of those elements, what other files each element is associated with, and what procedures each element is associated with. We have added a feature of our own which lists optional values for elements which can have multiple values. The values are contained in separate data sets which contain a chain of optional values and their meanings for each element. This feature is necessary for operation of other applications within our system, but could be added to supplement the Dictionary as a generic function.

Procedure Report:

This report shows the same information as the File Entity report except that DATA-PROCEDURE is the key link. Relationships between procedures are supported in this report in the same way they are between Files. The report shows all elements associated with a procedure, files associated with each element, and optional values for each element.

Element Report:

This report shows all elements in the Dictionary, associated files, procedures, and optional values.

Summary

The main intent of this presentation is our desire to share our experiences with other Dictionary users. The frustration that we experienced because of our sheepish obedience of HP's "Hands Off" attitudes was severe. Once we determined that we could safely and easily interface with the Dictionary in ways that we could control, the improvement in productivity and comfort was profound.

A secondary desire is to perhaps convey to HP that there is real potential in some of its products, and if they were properly aware of their user's needs and desires, that they could produce some really impressive stuff. None of the utilities that we have produced are particularly complex or difficult to implement. The needs of the users are changing to require each individual to be able to handle and digest more and more information all the time. This requirement can only be served by products like Dictionary/3000 if they themselves are adaptable or designed to be so.

The future is something that we should all be considering with the introduction of System Dictionary just around the corner. There are some substantial increases in the ability of users to design their own structures to help manage their organizations. Programmatic access will allow users to officially write utilities to enhance their use of the functions provided by the Dictionary. Extendible structure will allow site specific relationships to be designed into the Dictionary. The ability to have multiple domains will help to organize and manage the variety that exists within most systems.

System Dictionary will offer more flexibility and extensibility, but what will accompany these "improvements" will be an even greater need for documentation, control and management. Without the proper tools, these needs can cripple the efficiency and productivity of any organization.

What guarantee do we have that the same frustrations that we have experienced with Dictionary/3000 will be resolved by System Dictionary. Will it be able to automatically update the dictionary when a program is compiled ?; Will it provide an easier to use description editing interface ?; Will it provide the ability to use an external editor or word processor for maintenance of description ? I would be more than happy to use and praise an HP product that provided me with the functions that make me more productive.

Bibliography

Butler, Stephen M. Dictionary/3000 Walkthrough. SUPERGROUP, May/June 1984, page 18ff

Butler, Stephen M. DICTIONARY: Hub of Systems Development and Documentation. Proceedings of the HPIUG Anaheim Conference, 1984

Butler, Stephen M. Dictionary/3000 Extended Tour. Proceedings of the HPIUG Washington Conference, 1985

Colwell, Nancy. Dictionary/3000 - An Inside View. Proceedings of the HPIUG Montreal Conference, 1983

Harnar, Ron. Linking to HP System Dictionary. Proceedings of the HPIUG Detroit Conference, 1986

Hewlett Packard, Dictionary/3000 Reference Manual.

Hewlett Packard, System Dictionary Reference Manual.

Are you a 'User Loser' ?

Flo Barley
Pekin Memorial Hospital
Court and Fourteenth Streets
Pekin, Illinois 61554-5098

Introduction:

You can have software that will do just about anything and everything, but the best program in the world is useless without good documentation. Sure, you can use the applications by answering the prompts and filling in the blanks, but is it really doing what it should? Are you having data integrity problems because what you thought you entered was correct? Is it, in fact, actually doing something unknown to you?

The word 'documentation' can cover a vast area in software language. There's documentation for programmers - so one programmer can understand what another programmer has done in writing the source. This helps in following the course of events that the programming logic is to take. But this is invisible to the users who don't see those comments because they don't have the source or possibly the level of knowledge needed to understand what a programmer is trying to do.

I believe that user documentation is the most important part of the programming process (besides a working program, of course!) and this seems to be the area where many software packages fall short and end up collecting dust on the shelf or simply get tossed aside and never used again.

My goal with this paper is to help those frustrated users work their way through a maze that a brilliant programmer created and to show them what they can do to try to prevent this from happening in the first place. I also hope to touch the hearts and minds of some of those 'brilliant' programmers and encourage them to document thoroughly just what they expect the user to do.

I will point out certain areas where users can help themselves and other users, investigate several documentation packages on the market. The main point of help can be useful procedures written by the users themselves. I will outline how this can be done effectively.

Definitions:

Being a 'User Loser' can span both sides of the fence. You can be a user who is lost because of lack of good documentation or you can be e software or hardware vendor who loses users because of poor or non-existent documentation.

Have you ever played e game and found yourself in the middle of a situation that is not described in the rules? What do you do then? Make up your own rules, of course, and continue on. Not so easy with software! Can you imagine the havoc that would create?

Documentation is one area that seems to be the most dreaded process to programming. It is not an easy task to write, in simple terms, instructions for something that took hours, days, weeks or even months to create. You want to be careful that you don't insult the end user by being too simple, but yet write it plainly enough for even the most inexperienced user to comprehend. There are some users out there that are easily offended when trying to install their new software and have to page through it to get to the 'meat' of the manual. Yet, I feel that they need to be written in this manner, keeping them generic, with the idea in mind that your users are all beginners and have never even seen a computer before. This will cover all areas of users and will put your software in the best possible place - useable and functioning.

In defining the term 'end user', you should consider the individuals who would be the general users of your software. For instance, if it is an accounting package, you would think of CPA's or general accountants - but there are also your entry level employees who work in accounting that could also possibly be using this software and would need a little more instruction than the average accountant. If you keep these people in mind in your documentation writing, you will cover a vast majority of your audience and you will create a manual that will be useful when it's needed.

Now we are ready to define the types of documentation. Basically, there are four types: installation manuals, user manuals, operations manuals, and support manuals. Each of these contain information specific to the end user and the types of jobs they will be performing. Keeping them in a generic format will help in three ways: (1) you can easily provide quality control from your end and create consistency in your writing to make your job easier; (2) your users can migrate easily from one manual to the next with a consistent format that's easy to read; (3) your support and updates will be much easier to deal with.

The Beginning:

Your MIS Department is only as good as your hardware and software will allow and your software is only as good as your documentation. Without well written documentation for reference, your users will never be able to function to their full capacity. The better they understand the software and what they are doing, the better they will be able to do their jobs. This is the makings for a well-run business. I truly believe that users need to know why they are doing what they are doing and we should leave the 'robotics' for the gymnasium. They have a lot to learn from your documentation and it can be the most useful tool they can have.

For the creation of useful documentation manuals, first determine what types you will need for your package. One simple checklist will help you do this.

CHECKLIST FOR DOCUMENT CREATION

Will the system user have to . . .

- install the system without assistance?

  ( ) No
  ( ) Yes ==========> Installation Manual

- perform activities beyond those expected of most users?

  ( ) No
  ( ) Yes ==========> Operations Manual

- enter data to be processed?

  ( ) No
  ( ) Yes ==========> Procedure Manual

- modify the system in any way?

  ( ) No
  ( ) Yes ==========> Support Manual

- memorize many commands or file names?

  ( ) No
  ( ) Yes ==========> Quick Reference Guide

The Contents:

Installation Manual:

This should be written for users who install the software themselves without vendor on-site assistance. It should contain configuration requirements, the amount of memory needed to install the software and what additional memory may be needed to run the programs after the installation. You should include required or suggested file sizes, the file names needed for normal operation, groups and accounts needed and whether these need to be created in advance of installation or whether there is a job to stream to create them. If no job stream is included you could possibly give an example of one that could be written to do this. Include capabilities needed also, for the new user, groups, and accounts that were created. Write instructions for use on the current operating system and most importantly, an approximate time schedule to give the user some idea of how long it will take to complete the install.

Operations Manual:

This manual should contain information and instructions telling when programs should be run and how to run them, flowcharts, system commands and a list of programs and in what sequence they run. Data flowcharts should also be included, different than those for programs. They are simpler and show only input files, program processing, sorting and output files. Show processing points that could help in case of problems when a restart is necessary. List out all reports and perhaps a description of their output and any specifics needed for printing.

Support Manual:

This should include a complete system overview with system and program flowcharts, a complete listing of all programs in the system with a brief explanation of each program as to what process it will run or what report it will create and a description of any algorithms used in the system. Also include instructions for creating system interfaces and a comprehensive cross-reference for program use.

Procedure Manual:

This is keyed specifically to end users. It should explain what type of information the user must provide to the system, identify source documents and the method of transferring this information to the system. A good example of this would be screens used for input. Describe the system capabilities such as add, change, delete, inquiry, function keys, etc. and the results to be expected. Simple basics with how-to instructions such as what is needed to begin using the system - ie. setting up code tables, parameters and master files in a step by step fashion - is most important.

Additional Helps:

Quick Reference Guides:

These guides are a very useful addition to each manual. They contain frequently needed commands or procedures. It's simply a courtesy document that helps users after they have learned the system. It's not to be used as a training tool. Remember to keep it brief and concise and label all the information correctly. These guides will be designed depending on the system and it's use.

Tutorials or Teaching Modules:

These are independent teaching documents that would include illustrations and samples of what the user will see on the monitor. Walk through sessions are good training tools that takes the user from the beginning to the end of the process. Question and answers at the end of each session are also good. This can give the user and/or the trainer an idea where additional help and explanation is needed.

User Written Procedures:

Having users write their own procedures once they have mastered the process can be the most helpful document of all. They get a sense of accomplishment when it's done and also have a very good training tool for new users. These can be written in many different formats, depending on the application and your organization. Set up content standards specifying volume, level of detail, illustrations, examples and explanations or instructions. You may also want to set up a standard format for all users to follow so they are consistent with each application. User written procedures should be written on each application and assigned numbers. These can be indexed and cross-referenced alphabetically by procedure name and numerically by procedure number.

Combining Manuals:

Some manuals can be combined depending on their intended use. For example, the installation manual could be combined with the operations and support manual. Depending on the system it's written for, for instance on micro's, all could be combined into one document with sections for each manual type. Keep the user in mind when combining these and use the checklist to determine which types are needed and how they could possibly be put together.

Why Procedures?:

Our procedures are written with the beginning operator in mind so they can be used as training tools. After a general walkthrough on the system, most of our operators can train themselves in many areas. Until they become familiar and comfortable with the MPE help system, the procedures provide that little extra they may need. These could also be kept on-line, but if disk space is of a concern, a hardcopy is sufficient.

We also maintain a current report distribution listing in this manual since we print anywhere from 300 to 400 different reports. (Computers aren't completely paperless, now are they?). This manual was written on a personal computer until they became rather lengthy and hard to keep up with because of so many changes. We now have them on-line and enter them through a QUICK screen. They can be referenced on-line or modified and reprinted for the manual at anytime.

We have procedures written for our third shift operator that walks them completely through their shift from beginning to end. In an unsupervised, batch oriented environment like ours, this can be a lifesaver - if not save the DP manager some sleepless nights, too!

It's not an easy task to undertake, but getting the users involved in helping to write these procedures can give them some incentive to learn more about the system and what they are doing. It may be easier for you to write the procedures and have the users follow them, but you can turn them into robots this way. If the procedures are well-written, anyone can sit down and use them, which I feel is a good way to do it, but you can bypass the whole idea of having them understand just why they are doing what they are doing. Some explanation is needed in training or you can expand your procedures to include the explanation of what they are doing. Either is acceptable.

A quick reference guide can be used for certain commands to look up parameters or the order in which the parameters are to be used. As an initial training tool once the manual was read, written procedures seemed to work best for us. They can also serve as a quick reference tool if a little more explanation is needed in some areas.

Working in Data Processing, users will invariably call us when something goes wrong - whether it be hardware or software. Not coming from a true MIS environment, this sometimes is difficult. You can know and understand all the hardware, but knowing the software for different applications is difficult when you don't work with it day in and day out. Being somewhat familiar with the logic involved and the application itself because of handling the install, I can use the procedures for that particular application to walk through the problem. This is part of the idea for keeping them consistent as to content.

Formats:

The following is an example of a format we use in our operations procedures. It may seem rather simplistic but, rather than having to look up the most used commands in the manuals each time until you become familiar with them, we have certain procedures in an open binder by the console. These are numbered and then cross referenced in the index. They have been an invaluable reference tool for training new operators.

I have chosen our procedure written for Disc Drive Condenses since most of you may be familiar with the procedure. Most of our operations procedures were written shortly after we installed the system. It was an effective training tool for me because if forced me to use all the commands in order to write explicit procedures for other operators. They in turn, have used this for reference in training.

I include any UDC's that were set up so as to incorporate these with the command and further their knowledge of how else it can be done. They can learn and reference all the commands from the one procedure. These are maintained on a PC but could be kept on-line if you so desire. Once they are written, there is not a lot of maintenance unless you would change operating systems (which may only affect certain procedures) or add or change UDC's for a particular command.

Some procedures can become rather lengthy, for example, the command for SPOOK. I have incorporated within our procedure for SPOOK not only the UDC's, but a step- by -step how to in putting spoolfiles out to tape and then bringing them back again and in verifying that the tapes did indeed copy the spoolfiles.

You can choose just how detailed you want them to be and in what procedures you may feel further explanation is needed. They can be personalized or de-personalized. Many of our procedures are site-specific, but could be written in a more generic way that others could also benefit by them.

By having the users write their own procedures, they can incorporate source documents which can be a tremendous help both in training new users or in problem solving from a DP standpoint. If I should have to work out a problem for them, my knowledge of the logic flow along with their input procedures can greatly reduce the time and effort spent in solving it.

This basic format can be used for any procedure. An outline will give you a rough idea on a walk through and can touch important points as you go along. It can be modified in such a way to give more detail. A blurb at the beginning explaining what the application or particular function does will help; then follow with a step-by-step procedure.

Example Procedure Formats:

Example 1:

| DATA PROCESSING |

SUBJECT: PROCEDURE FOR CONDensing DISC DRIVES

DATE: 08/16/86 NUMBER: A335-03 PAGE: 1 of 2

WRITTEN BY: Flo Barley REVISED

APPROVED BY: _____

To utilize disc space and enhance response time, all disc drives will be CONDensed weekly with the exception of Drive 6 (private volume). Condenses are scheduled for Sunday, Monday and Tuesday nights. Check the Log to see which drive is due for condense.

I. Do a full backup per usual procedure (A335-11) being sure to check the listing prior to condense that all files were successfully backed up.

II. After backup, leave limits at 0,0.

III. Print a FREE5 listing.

A. On the console enter the file equation.

1. Type "FILE FREE5OUT;DEV=LP" ~ (~ = press RETURN)

B. Run the FREE5 program.

1. Type "FREE5" ~

NOTE: PRINTER WILL PRINT FREE5 LISTING WITHIN TWO MINUTES.

3. Label printout "BEFORE COND. DRIVE# nn".

(nn = logical device for drive you are condensing.)

Are you a 'User Loser' ? B

SUBJECT: PROCEDURE FOR CONDensing DISC DRIVES

IV. When the colon prompt returns:

A. Type "VINIT" ~

B. When the ">" prompt appears type "COND n".
(n = logical device number for drive being CONDensed)

V. Warning messages will display on the console.

A. At the beginning of each CONDense "HH:MM/$Snnn/nn/WARNING*System logging disabled while CONDensing (PUWARN 179)".

B. At the end of each CONDense "HH:MM/$Snnn/nn/WARNING*System logging enabled by CONDense (PUWARN 180)".

NOTE: IF ANY OTHER WARNINGS OR ERROR MESSAGES OCCUR WRITE THEM DOWN IN THEIR ENTIRETY. CALL THE D/P PERSON ON CALL - DO NOT PROCEED!

IV. After the appropriate drives have been CONDensed successfully: .

A. Exit the VINIT UTILITY.

1. Type "EXIT" ~

B. Record CONDense in the CONDENSE LOG.

C. Run a FREE5 listing.

1. On the console enter the file equation.

a. Type "FILE FREE5OUT;DEV=LP" ~

2. Run FREE5 program.

a. Type "FREE5" ~

3. Label printout "AFTER CONDENSE - Drive nn".

D. Send FREE5 listing to Data Processing Coordinator.

Example 2:

| BILLING |

SUBJECT: PROCEDURE FOR LATE CHARGES AND CREDITS

DATE: 01/30/86 NUMBER: A430-60 PAGE: 1 of 1

WRITTEN BY: Dee Driskell REVISED

APPROVED BY: _____

I. Late charges and credits are delivered to the billers desk daily. The following should be done at this time:

A. Do a REBILL.

1. From the Main Menu select THIRD PARTY BILLING
2. Then select REVENUE RECLASSIFICATION AND REBILL
3. Enter the account number
4. At MODE enter '2' (for REBILL)
5. Enter a 'Y'
6. At MODE enter 'UR' (for UPDATE RETURN)

II. This procedure will not only give you a new statement and a new UB82 but will also correct the log.

III. Exception: Medicare Inpatient Accounts

A. If the account has already been billed, do nothing. This leaves late charges in Account Detail Activity ART040. This also leaves the Medicare Log showing the late charge or credit.

B. If the account has not been billed, follow step I.

Third Party Documentation Packages:

There are several good documentation packages on the market today. It was difficult to evaluate all of them to get an even comparison and be equally fair to each vendor. I was restricted in some respects by the data dictionary that we use at our installation site and also by my time schedule. But an overall view of all of the packages that I did receive information on was very good. They even had good documentation on how to create documentation! Some of the key points of interest that you would look for in creating useful documentation with a third party documentation package are: 1). What levels of documentation does it provide? 2). What languages are supported? 3). What dictionary does it require for access? 4). Does it have cross-referencing capabilities? Other options to look for are on-line help, menu driven for ease of use, default prototyping techniques and ease of maintenance and updates.

The documentation packages that I had access to were Robot 3000 and Documentation/3000. I had reviewed a tutorial on Powerhouse Architect but was unable to actually run this because our installation uses QQQR and Architect uses the QQQ data dictionary. Two others I am also aware of but not fully informed of are Q-DOC and LL'Spirit. All of these packages will fully document your programs and create useful documentation from the programming level. This can be a big labor saving device from a programmer's viewpoint since their time is very valuable and good documentation can be very time consuming. Many of them have additional tools that can be very useful and would be well worth looking into if you are concerned with the documentation that you now have or if it's nonexistent. They can produce reports from various levels and give you good hardcopy documents for users.

I was very impressed with the degree of accuracy that these packages can produce. They are capable of providing valuable information with little or no manual input once the files are defined. They can also provide documentation before the program is even written. This can be a tremendous help in prototyping to user specifications and being able to demo it before completion. When the final project is done, you can have a fully tested version complete with documentation.

Having this type of documentation can put you on the right road to properly written manuals. Documenting the source code can help installers lead the way for users in writing how-tó procedures. Good documentation will keep the software on the market and useable. After all, would you attempt to build a house without any plans or blueprints? How many of you have sat under the tree on Christmas Eve struggling with a bike or toy that had to be assembled before the next morning? With the proper instructions, your job at hand can be so much easier and enjoyable. This is what we need to strive for in the world of computer technology.

Summary:

Documentation is a much needed tool in many ways. Programmers need to document their source for support purposes and for installation - whether it be for installers or users who self-install the product, and for end users who need to know what to do with it when they get it installed. Sometimes the documentation process can seem to take longer than writing the program, but it's needed equally as much.

When looking at software packages, don't just look at the product functionality - take a good look at the documentation. Do they meet your requirements? Will your users be able to follow and understand them? Are the necessary manuals available? All of these questions can be very instrumental to your installation.

Users can help themselves and other users by writing procedures internally whether it be site-specific on how the applications are used in their particular business or generic in fashion for multiple users or businesses.

Set up format guidelines for them to follow if you want a more generic format for your institution or allow them to create their own. Anything that's workable so they will get the use they deserve is the main goal.

Reference:

Perryman, Polly, "Standards and Documentation", Information Center, August, 1986, p.31.

THE CROSSROADS OF FOURTH AND FIFTH GENERATIONS

Presented by: Ken Begbie
of: BBJ Computers International Inc.
3707 Williams Rd.
San Jose, CA 95117

INTRODUCTION

The author has had extensive experience in the design and use of a fourth generation language (TODAY) which has been implemented for a variety of operating systems, (MPE, VMS, UNIX, XENIX, DOS). During the last three years, he has been involved in prototyping Expert Systems, including the MARLOWE Prolog based Crime Detective System, and a number of machine learning systems.

This paper discusses the integration of a specific machine learning process with the fourth generation language (TODAY-ES) and describes how such an integration brings the application of 5th generation techniques within the scope of a traditional development environment.

FIFTH GENERATION

The term "fifth generation" conjures up visions of human like robots, immense knowledge data bases and super intelligent computers which can converse in every known language. These are all goals of the research currently being carried out, but there are many, more menial, but more immediately achievable, goals which are now almost within the realm of commercial usefulness. We will look at one of these in detail, namely the "Machine Learning" process whereby a computer can be programmed to derive knowledge from records of historical events.

It is useful to consider the evolution of programming languages. The different generations are generally considered to be as follows:

1st generation	Machine Code
2nd generation	Assembly Languages
3rd generation	COBOL, FORTRAN, BASIC... etc...

4th generation	FOCUS, MANTIS, LINC, TODAY, POWERHOUSE, etc...
5th generation	LISP, PROLOG, ADA

It is important to understand that whereas the first 4 generations were evolutions, one from the other, in an attempt to improve programmer productivity, the 5th generation is a departure, and represents a starting all over of our approach to the use of computers whereby knowledge and reasoning ability becomes the dominant objective.

Research and development in 5th generation systems have been going on for over 30 years in many countries:

USA:	LISP programming language Extensive Support Hardware and Software Large "Rule Based" Systems (ie MYCIN, R1)
JAPAN:	MITI 10 year project New Hardware designs Robotics PROLOG programming language
EUROPE:	Logic Programming PROLOG programming language Machine Learning
AUSTRALIA:	Logic Programming PROLOG Programming Language Machine Learning (Active Software Research and Development Industry with substantial Government backing)

FOURTH GENERATION

Even though there are many (perhaps hundreds) of 4GL's on the market, it is still not clear just what the term "fourth generation" means. To confuse things further we talk of 3 1/2 GL's and 4 1/2 GL's! However, in general, 4GL's have tried to improve productivity by the use of "non procedural" techniques and an attempt to eliminate conventional third generation syntactical code.

We have come to expect certain key features in a 4GL:

1. Visual (WYSIWYG) Interfaces
2. Easy modification, customization, versioning

3. Automatic Documentation
4. Central - Active - Data Dictionary
5. Portability - Hardware, Operating Systems, Data Bases
6. Multi Lingual - the language and applications
7. Efficient Performance
8. Complete in itself - no need to use a 3GL

Typically we use most of these features in the 4GL's available on the market. It is interesting to note that whereas the user systems generated by a 4GL are often very similar in operation and appearance, the development languages are usually vastly different in approach and technique.

CROSSROADS

Where do the fourth and fifth generation languages come together? It turns out there is no immediately available or obvious application of one technique to the other. Hence, unless a deliberate effort is made to bring fourth and fifth generations together, they will tend to diverge along their own separate paths.

Some common objectives for the two generations might be:

1. Applying Expert Systems to the 4GL to aid in system development. An excellent and potentially achievable objective, but with the present state of research, we can't expect any "real" contributions in this area for 5 to 10 years.

2. Applying Expert Systems to the design of Systems for implementation in a 4GL, perhaps an Expert CASE System front ending a 4GL. This is conceivable in the near term, perhaps 2 to 5 years.

3. Use of Expert Systems is optimizing data base access and query performance within the generated code of a 4GL. We are still a long way from a useful implementation of this technique.

All the above are possibilities for the future, but are not immediately available to us.

An immediate benefit can be obtained however by simply "bringing together" the functions of the two generations, whereby the 4GL is used to support all the user interfaces and record management, and the 5GL is used to provide the derivation and processing of knowledge. The two generations overlap very little, but can be nicely integrated with a little careful planning.

MACHINE LEARNING INTERFACE

We will look at a specific integration of the two generations, namely the TODAY 4GL and the ID3 machine learning algorithm.

Machine learning techniques have been developed allowing computers to extract intelligence (ie knowledge) from historical data. Computers can be made to learn from:

1. Examples, whereby an "expert" will tell the computer of a series of examples from his own experience.
2. Historical Records, whereby the computer reads all or part of the known history and derives knowledge from it.

In the past, machine learning by example has been used successfully using the ID3 algorithm. The disadvantage of learning by example is that it requires an expert to prepare the examples, taking up his valuable time.

More recently, interest has grown in machine learning from naturally occurring historical data, as is a common by-product of many conventional computer systems (ie history of customer sales, stock movements, customer payments, etc). Typically this involves much higher volumes of data, and we must expect some errors and/or inconsistencies in the data. We should also expect some incomplete information, which may nevertheless still hold useful intelligence.

During the last year the ID3 algorithm has been redeveloped to handle naturally occurring history and as such has been renamed C4. This new algorithm also provides selective tree pruning based on a controllable incremental error rate predictor, resulting in a more condensed (and readable) form of the knowledge.

HOW DOES MACHINE LEARNING WORK!

We have a goal, to make a decision relating to our problem. We have established (or know intuatively) that certain attributes are relevant to solving our problem and making a decision.

Although a trivial example, say our goal was to decide what to wear today. We know that relevant attributes relate to the weather. So we make observations over a period of time as to what people wear on different days with different weather conditions. These observations contain the intelligence which is "mined" by the machine learning algorithm to help make future decisions.

The Algorithm builds a decision tree in which each of the relevant attributes are tested in turn following a specific branch to a final decision.

In general, the use of the decision tree in decision making is very fast, as only one branch need be followed, and the C4 algorithm eliminates any redundant attributes in each branch. However, the generation of decision trees is time consuming, as is their checking and validation.

APPLICATIONS OF MACHINE LEARNING

Machine learning techniques can only be applied where all the relevant attributes are known and can be appropriately quantified. Such applications are "closed" systems where external factors do not influence the decision.

Some suitable targets for Machine Learning Methods are:

- Breakdown Analysis
- Production Monitoring
- Maintenance Scheduling
- Insurance Applications
- Credit Card Applications
- Cash Flow Projections
- Stocking Strategies

5GL FUNCTIONS IN THE INTEGRATED LANGUAGE

The C4 machine learning algorithm is the core function and is implemented in the "C" language.

Language functions relating to the Decision Tree:

1. Generate the Tree
2. Display the Tree (visual format)
3. Test the Tree (measure expected performance)
4. Modify the Tree (change the tree structure)
5. Update the Tree (feed back errors and regenerate)

Problems which must be addressed:

1. If the historical data has errors
2. If the historical data is incomplete
3. If the historical data is insufficient to provide the necessary level of knowledge
4. If attributes have been omitted
5. If redundant attributes have been included

Application of the decision tree can be either:

1. Interactive consultation where the operator is asked for information on each of the relevant attributes and is provided with the derived decision. Only the relevant unknown attributes are called for.

   The operator can ask the computer:

   a) Why did you ask for information on that attribute?
      or
   b) How did you come to that conclusion?

   The consultation process must give sensible and meaningful answers.

2. Stream decision making where the attribute data is already on file and decisions can be made in a batch mode and applied automatically. There is no interactive user review of the derived decisions in this mode.

4GL/5GL INTEGRATION STRATEGY

The development features of the(TODAY) 4GL have been deployed to support the knowledge acquisition, analysis and decision making functions of the 5GL. The two come together at a single integrated development environment (TODAY-ES).

1. The Data Dictionary is extended to identify and categorize attributes for the decision tree.

2. The Data Base is used to hold the historical data used for analysis in the decision tree generation.

3. The File command system is used to retrieve the required attributes from the historical data.

4. The Decision Tree control parameters are input using a specially prepared developer screen.

5. The Decision Tree generation is a separate C program producing pseudo code much like a decision table. (The knowledge is embodied in the application pseudo code).

6. The Data Base is used to hold the generated "critical set".

7. An automatic consultation screen function is provided based on the standard automatic screen builder.

8. A "DECISION" command is added to the language command set.

A NEW ROLE FOR THE SYSTEM DEVELOPER

With the integrated 4GL/5GL product, the role of the system developer is extended to encompass "knowledge engineering".

A knowledge engineer (KE) has the task of ensuring that the knowledge in an Expert System is correctly set up and maintained, is sufficiently accurate and complete for the decision making process, and is correctly applied by the inference method supplied.

The TODAY-ES developer can write an application embodying both conventional and knowledge based processing as a single integrated system.

The Developer may:

1. Define a data dictionary, including decision goals and attributes.

2. Define records and files, including those for historical analysis and tree generation.

3. Define screens for collecting data, including history.

4. Define and generate decision trees from user history files or example sets.

5. View and validate the decision trees.

6. Define interactive consultation screens.

7. Use the inbuilt DECISION command in data edits or batch processes.

8. Define reports resulting from example set input, validation, decision tree generation and decision making processes.

The System User may:

1. Utilize the decision tree in consultations to obtain assistance in decision making.

2. Generate reports from decisions made.

3. Feed back any anomolies or incorrect decisions to the developer for tree modification or regeneration.

BENEFITS OF EFFECTIVE INTEGRATION

Brings 5GL techniques within the reach of the traditional application developer.

Provides visual (WYSIWIG) support for expert system development.

Provides a vehicle to develop the systems supporting both traditional and knowledge based computing.

FUTURE LANGUAGE STRUCTURES

We are likely to see more extensive integration of 5GL methodologies, in the form of "tool kits" within the development environment. The "tool kit" will offer many options ranging from Machine Learning to full blown Rule based systems and offer interface to 5GL languages (LISP, PROLOG, ADA, etc). These options will be offered alongside the 4GL development language capabilities together with the traditional interfaces to Spreadsheets, Word Processors and Graphics.

Integrating the user environment with HPDeskManager

Graham Bell
Hewlett Packard Company
3410 Central Expressway, CA 95051

INTRODUCTION

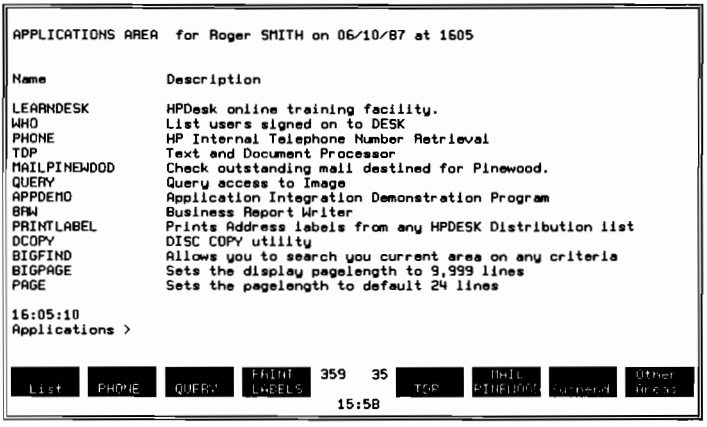
People who use computers in their everyday work often remain within a single primary application as they perform their tasks. To the user, this application provides the environment with which he or she is most comfortable. These "environments" differ from user to user just as tasks vary with each user's job responsibility. The secretarial or clerical worker will often remain in a word processing environment. The professional/manager might stay primarily in a spreadsheet application, and the accountant would probably want to stay within the accounting application. All of these environments provide different functionality and user interface.

But all of these user types have one fundamental need in common - the need to communicate and distribute the end results of their work. Once a document or report is generated, that information will invariably need to be distributed within and beyond the organization in order to realize its usefulness. Electronic mail provides the ideal vehicle for this, so it would make sense that the ideal user environment would be one that gives easy access to distribution from whatever application the user is in. But rather than introduce yet another user environment, the electronic communication application should have the ability to simplify user tasks, not add to them, thereby adding value to the applications of choice.

This paper describes how HPDeskManager can be used to integrate with existing applications in order to simplify use, enable the user to be more productive, as well as provide an interface to electronic distribution.

THE APPLICATION UMBRELLA

The most obvious way that HPDesk can be linked to other applications, is through its ability to run another application as a "son" process. Any application can be installed into the Application Area of HPDesk and then called by using its assigned "command" from any HPDesk prompt. The program can also be run from anywhere within HPDesk and at anytime (except when in block mode) by selecting SUSPEND followed by the command for the application. In either case HPDesk will suspend (in the MPE sense) and the application will take over. On exiting the application, the user will return to HPDesk.

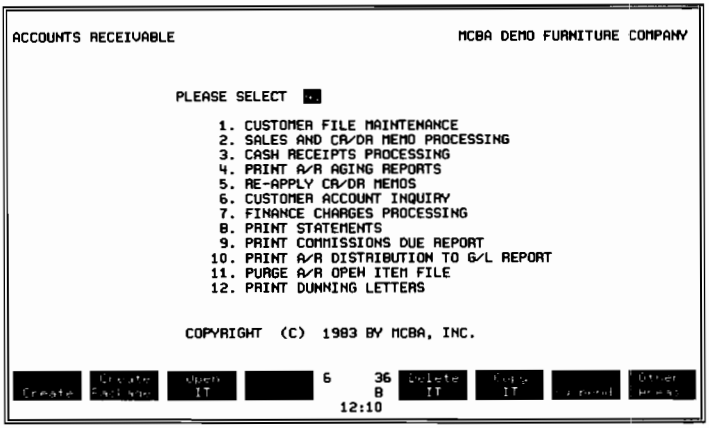


The Applications Area of HPDesk lists those programs that are available to the user. To help maintain system security, when items are added to the Applications Area, the administrator can choose which group of users will have access by the User Group security feature. Every user on HPDesk can be placed in one or more of 48 Security Groups which can then be used to control access to items such as Applications.

Applications can be accesssed either by their command Abbreviation name or a substring of it. To simplify selection even further, the function keys can be redefined as illustrated. (Seven of the eight HPDesk function keys can be redefined anywhere within HPDesk. Function key f7 is always used for Suspend/Resume and cannot be changed).

It may be more desirable to create separate menus which request more information than simply the name of the program. The example illustrated shows a method for calling an accounting application module. Here the user specifies which functions need to be performed rather than

just the name of the application to be run. Another menu then appears with a new set of options, so some of the decisions which the application would normally request, can be handled within HPDesk. This is very useful for simplifying access to applications which might otherwise be unfriendly.



This integration was achieved through the use of Script files and is outlined in the next section.

SCRIPT FILES

In its simplest form an HPDesk Script File is equivelant to an MPE UDC, chaining together a number of HPDesk and MPE commands into one. These commands can be used to automate a routine task. For example, a script file to automate the mailing of a monthly report would look like the following:

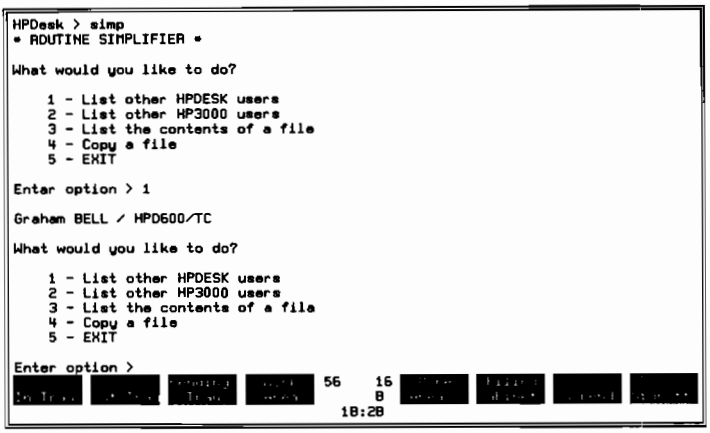
```
send "This month's status report" to "Exec"
ack 4
urgent
mail
&EXIT
```

But Script files also have their own special directives and functions and are able to pass parameters and manipulate strings as well as many other features. This gives Scripts full programmatic capabilities, so that new HPDesk commands can be created.

The following Script implements a menu to simplify the use of some MPE and HPDesk capabilities. For space utilization, comments are shown in italics on the right rather than as "&comment" lines within the script.

```
    &print * ROUTINE SIMPLIFIER *
    &print
$START                                                    label
    &print What would you like to do?                     begin the menu
    &print
    &print     1 - List other HPDesk users
    &print     2 - List other HP3000 users
    &print     3 - List the contents of a file
    &print     4 - Copy a file
    &print     5 - EXIT
    &print                                                prompt for choice
    &save OPT  <prompt "Enter option > " "5" 1>           and save in OPT
    &forward LABEL<VAR OPT> <NOT <GREATER <VAR OPT> 5>>check if in range
    &back START
$LABEL1                                                   label for item 1
    &print
    &who                                                  list HPDesk users
    &print
    &back START                                           go back to menu
$LABEL2                                                   label for item 2
    &print
    :showjob                                              mpe :showjob
    &print
    &back START                                           go back to menu
$LABEL3                                                   label for item 3
    &save FNAME  <prompt "Enter File Name > " "" 8>       get file name
    :file input=<var FNAME>                               set up file eq.
    :file output=$stdlist                                 for TYPE program
    &save LENGTH 72
    execute type.util.sys <var LENGTH>                    run TYPE program
    :reset input
    :reset output
    &back START                                           go back to menu
$LABEL4                                                   label for item 4
    :run copy.util.sys                                    run COPY program
    &back START                                           go back to menu
$LABEL5                                                   label for item 5
    &EXIT                                                 exit script
```

As with Applications, Scripts can be installed with restricted access, and executed by a single command (or Abbreviation which is the term used within HPDesk). The above Script - SIMP - would display the screen shown below when executed:



HELP!

Expanding on our accounting application as an example, the screen illustration shows the application itself being run under the HPDesk umbrella. This particular application does not have its own on-line HELP, however, by typing HELP at any prompt within the application, the user can switch to the appropriate HPDesk HELP screens, which provide help not on HPDesk, but on the specific segment of the application in which HELP was typed.

The ability to customize and integrate HPDesk's HELP facility is clearly a very useful integration feature and warrants further elaboration.

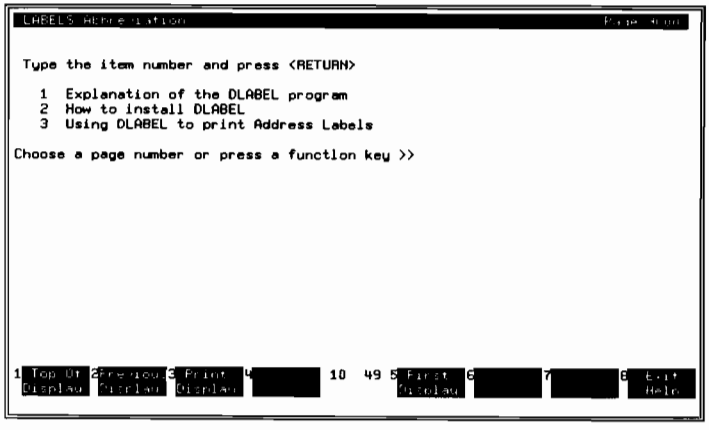
HELP screens are arranged in pages, each with its own unique page number. Pages can reference each other by the use of command headers, so that supplemental text pages can be added to existing ones, new menu pages can be added, and the new HELP pages can be referenced by existing ones. This is achieved by including HELP Directives with each page, and can best be shown with the following example.

This example shows HELP added for an application called DLABEL which enables the printing of full name and address labels using an HPDesk Distribution List. Customized HELP pages can be added at page numbers starting at 9000. The following example shows the format for the first few of those pages:

```
$IDNONHP 9000 "Printing Name and Address Labels"
$TITLE "LABELS Abbreviation"
$SONS 9001,9002,9003
Type the item number and press <RETURN>
```

This first page defines that there are three option pages for HELP on this topic. These options (taken from the quoted string in the first line of the son pages) will be displayed as a menu. The next two pages are the first two option pages (son pages) which also define the HELP text by the $TAG Directive. i.e. selection of "Explanation of DLABEL" will cause text page 9001,1 to be displayed.

```
$IDNONHP 9001 "Explanation of the DLABEL program"
$TITLE "Explanation of DLABEL"
$TAG1

$IDNONHP 9002 "How to install DLABEL"
$TITLE "DLABEL installation"
$TAG1

$IDNONHP 9001,1
DLABEL is a utility which runs on an HP3000 system with
HPDeskManager (36570A) installed, and allows the HPDesk user to
print a full 6 line name and address label ...
```



To complete the customized HELP, page 9000 can be added to the HELP main menu page as a "son", so that "Printing Name and Address Labels" is included in the menu:

```
$ID 0,"Help System main menu"
$TITLE "Help System Main Menu"
$SONS 600,1200,1600,2000,2400,2800,3200,3600,4000,4200,9000
```

Once the new pages have been installed, the HELP screens can be called from within HPDesk in the usual ways. In addition, HELP can also be called from the business application. This is achieved by using application code to pass data to and from HPDesk via an Extra Data Segment (XDS). This process is known as Application Data Passing and can be used for much more than just online HELP.

APPLICATION DATA PASSING

As discussed in the opening paragraph, an essential user requirement is the ability to communicate data. The easier and more transparent it is to transfer this data to electronic mail, the more value it is to the user. The application could simply create an MPE file for copying to HPDesk for subsequent mailing. Alternatively the files can be passed directly as HPDesk mail via the Foreign Service Connection. The method discussed below goes a step further by providing much greater flexibility for data transfer and distribution. As outlined above, one of its capabilities is the transfer of HELP information, but there are a number of others.

An application running under HPDesk can be suspended in different ways, known as Interrupt Modes. There are 13 different interrupt modes each with its own function. The mode which is selected is controlled by setting a JCW (Job Control Word). The 13 modes are:

Mode 1: General Interrupt 'Suspend Key' - The application suspends and returns to HPDesk in Suspend mode.

Mode 2: Return a file to the current HPDesk Area.

Mode 3: Copy a file from the current HPDesk Area.

Mode 4: Return a file to the Work Area.

Mode 5: Copy an item from the Work Area.

Mode 6: Replace an item in the Work Area.

Mode 7: Initialize Forms processing.

Mode 8: Read the next Forms batch file.

Mode 10: Convert file from one document format to another.

Mode 11: Access the HPDesk HELP subsystem.

Mode 12: Copy from the current Area using the Item Number.

Mode 13: Copy from the current Area using the Subject.

As can be seen, the hooks into HPDesk from the application are varied and numerous, and this opens the door to a number of different possibilities. Here are a few examples of what can be done besides accessing HELP:

* Perhaps the host application is the primary environment for the user, with only occasional HPDesk usage. The user can Suspend from the application into HPDesk, read mail, check the calendar etc., and Resume back to the same part of the application.

* A survey is sent out to HPDesk users, using a VPLUS form which must be completed and returned. An application can be written to collect and process the replied forms, automatically and transparently.

* A database text retrieval application can be integrated so that a user can retrieve information directly from the database, to their HPDesk WorkArea.

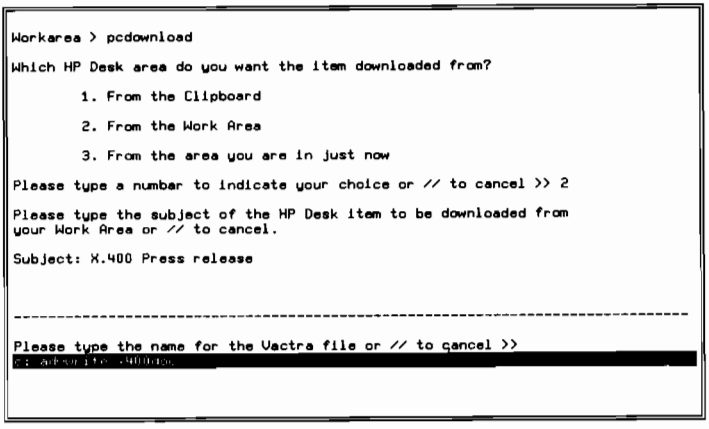
* An application can be written to transfer data in the form of MPE files directly into HPLibrary along with relevant search attributes.

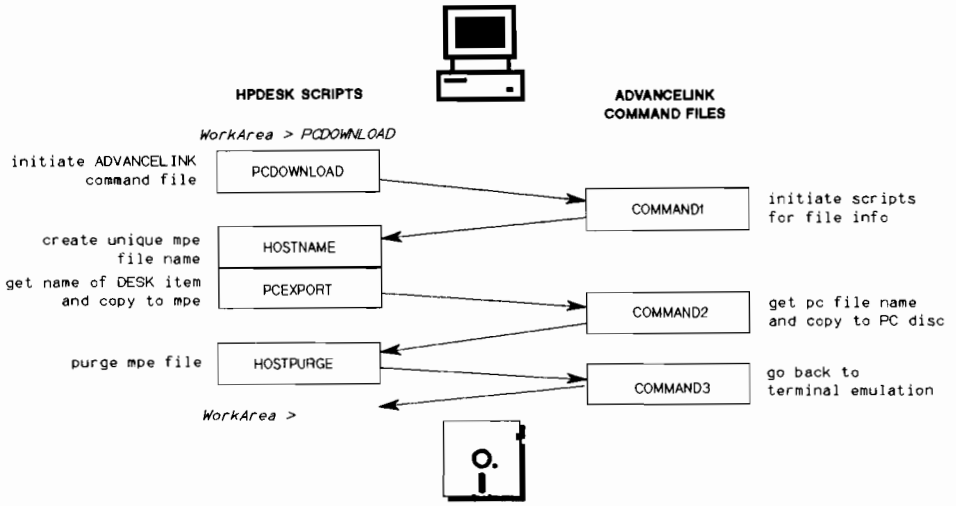
These facilities truly create an integrated business computing solution, one which allows practical business applications to be transparently integrated with distribution capabilities.

All of the above examples have been concerned with integrating host applications with HPDesk. But what about PC integration? The next section discusses how the PC user can enjoy the functionality of HPDesk without losing the benefits of the personal computer.

PC INTEGRATION

Most PC users will want to mail PC files as well as messages, to other PC users, and AdvanceMail may be the application of choice. But many users will need all of the functionality of HPDeskManager and HPLibrary etc., as well as other integrated host applications, and they will also have an equally important need for PC file transfer and mailing, without having to switch back to a PC application. This functionality is available and can be addressed in a number of different ways. The following example illustrates one of them.





The diagram above is a simplified representation of one method for PC file transfer while within HPDesk. It shows the download of an HPDesk item to the PC's disc drive, and its main operation is as follows:

* The PCDOWNLOAD command invokes the PCDOWNLOAD Script.

* The script uses an AdvanceLink HOSTCONTROL command sequence to CHAIN the command file COMMAND1.

* COMMAND1 invokes the HOSTNAME script to create a unique name for the MPE file to be copied to.

* COMMAND1 then invokes PCEXPORT to request the name of the HPDesk item and copy it to the unique MPE file.

* HOSTCONTROL is again used to CHAIN the command file COMMAND2, which asks for the PC file name and uses the AdvanceLink command &DSCOPY to copy it to the PC.

* COMMAND2 then invokes the HOSTPURGE Script to get rid of the MPE file.

* A third command file COMMAND3 is CHAINed, which resumes normal terminal emulation.

This routine forms the fundamental link between HPDesk and the PC without having to switch to a separate PC application. The transfer of PC files itself is no great accomplishment but this ability does open the door to achieving our original objective of providing the user all of the benefits of the HPDesk environment while maintaining easy access to the applications that they use most, whether host or PC.

CONCLUSION

The preceeding sections have all considered different aspects and components of a solution that can be provided with current products and technology. But what is the overall picture? What is the ultimate solution (if there is such a thing) for the computer user today? Perhaps the specification for the average user's ideal environment would look something like this:

* Ability to switch easily between both PC and host based applications.

* Customized screens, HELP, and automated single keystroke commands for ease of use.

* Ability to read, edit, and send any type of information, PC or MPE files, from within HPDesk.

* Accessibility to shared (LAN based) PC files and documents through a central directory.

HPDeskManager provides a solution which incorporates all of these capabilities via its open architecture foundation. Used in conjunction with products such as Resource Sharing and AdvanceLink plus job specific applications, office automation is truly evolving towards business automation.

The ABC's of Modems

Tom Benedict
Hewlett-Packard
Customer Escalation Center
2 Choke Cherry Road
Rockville, Maryland 20850

1. INTRODUCTION

It is difficult to transmit directly the digital signal from a DTE over an analog telephone line. A modem (modulator/demodulator) gets around this problem by modulating the digital signal on a carrier wave which will propagate well over an analog channel. A carrier wave is a continuous frequency capable of being modulated. Modulation is a technique used to encode the digital signals into a spectrum that is easily passed over analog telephone lines.

2. MODULATION TYPES

There are many different techniques used to modulate a digital signal on a carrier wave.

Amplitude Modulation (AM) encodes the digital signal by changing the amplitude of the carrier wave. AM is the technique that is used by radio telegraph and AM radio. AM encodes the digital signal by raising and lowering the amplitude of the carrier; the frequency stays constant. In the simplest form of AM the presence of carrier represents a one and the absence of carrier represents a zero. Amplitude modulation is not normally used for modems because of amplitude distortion that is normally present on a phone line.

Frequency Modulation (FM) encodes the data by using one frequency for mark ("1") and another frequency for space ("0"). The amplitude of the carrier remains constant. A mark frequency is sent when the line is idle. FM is used in low speed modems such as the Western Electric 103A and 212A. FM is also used in FM radio. FM offers good noise immunity over telephone lines.

Phase Modulation (PM) encodes digital signals as positive and negative phase shifts of the carrier. The amplitude and frequency of the carrier stay constant. The common method of demodulation in PM is called differential detection. The receiver samples the data and compares the current phase with the previous phase. In a two-phase modem a zero degree phase change could represent a zero, and a 180 degree phase change could represent a one. PM is used only in synchronous modems and offers good noise immunity.

Quadrature Amplitude Modulation (QAM) is a combination of phase and amplitude modulation in order to encode more information on the carrier wave. First generation high speed modems can run reliably up to 9600bps using QAM. A common implementation is to use eight phases 45 degrees apart, and two levels of amplitude, resulting in 16 signal points. In each signal element four bits can be encoded.

Second generation high speed modems use a variation of QAM called Trellis Coded Modulation (TCM). TCM adds coding information to the traditional QAM technique. This results in a dependency between successive signal points. TCM is continuously looking backwards and forwards, comparing already received data to make a final signal point identification. Second generation modems using eight-state, two-dimensional TCM have good performance at 16.8 kbps.

Third generation high speed modems can run reliably up to 19.2 kbps. Codex uses a 64-state eight-dimension TCM. This coding scheme concatenates 4 two-dimension signal constellations into one eight-dimension constellation. Each one of the four constellations uses 160 points. Modems running at 19.2kbps are state-of-the-art and run very near the theoretical limit of modem speed on a leased analog phone line. If a four-to-one compressor were installed together with a 19.2kbps modem, a theoretical rate of 76.8 kbps could be achieved. A compressor is a device that represents repeating characters, strings, trailers, and headers as shorter sub-codes.

3. MODEM TYPES

There are four basic types of modems:

- Asynchronous
- Synchronous
- Short-Haul
- Medium Distance

Asynchronous Modems

Asynchronous modems are mainly used for data communications from terminal to host. Most asynchronous modems can be strapped to run synchronously.

The price range for asynchronous modems is:

low speed	(110-1200bps)	$50 - 500
medium speed	(2400-4800bps)	$400 - 2000
high speed	(9.6-19.2kbps)	$999 - 4450

Asynchronous modems use no transmit or receive clocks but clock on data. This means that the modem transmits the logical condition of mark or space on the telephone as long as it is present on the DTE interface. When receiving a mark or space condition from the phone line, the modem transmits mark or space to the DTE. The modem has no knowledge of bit times or character definition. Asynchronous modems can operate at variable data rates because of this.

Telequality Associates tested four 1200bps asynchronous modems: Bell 212A, Hayes Smartmodem 1200, Novation Smartcat 212, and the Microcom SX/1200. The modems were tested on both worst case and best case telephone lines. The Bell 212A modem was rated best on a good quality phone line, winning three out of four tests. On a very noisy phone line the Bell 212A transmitter beat all the competition by a wide margin. Also tested were four 2400bps modems: Concord 224, ATT 224B, Micom M3024, and Hayes 2400. In these tests the Concord 224 was the best performer; receiver signal degradation on a poor quality line did not start until 22db.

The hot market in asynchronous modems is 9600bps and above. The fastest modems in this market are the Telebit Trailblazer and the Microcom AX/9624c. When using the 9624 from Microcom you can log onto your computer port at 19.2kbps. The modem compresses the data and sends it across the phone line at 9600bps with a fallback speed of 4800bps. Actual throughput as high as 17423bps can be achieved. This modem uses adaptive duplex. The modem does not transmit and receive at the same time but when used with a terminal logged on to the HP 3000 at a speed of 9600bps, line turnaround is not noticeable. If you logon at 19.2kbps you will notice line turnaround. This modem can also operate as a 212A, V.22 bis(2400bps), and V.22(1200bps). The cost is $1799/unit.

The Telebit Trailblazer uses a modulation technique that is similar to the one that Gandalf uses in their SM9600 superModem II. The Trailblazer modulates up to 512 different carriers on one dial-up phone line. The theory is that slow speed modems exhibit excellent noise immunity and that low bit rates do not require complex equalization methods. High data rates can therefore be achieved by operating a large number of low speed modems in parallel. The Trailblazer sends up to 100bps on each one of its carrier frequencies. When the link is established, the modems determine which carriers to use and which bit rate to use on each carrier. When the line degrades the modem will only need to reduce the bit rate in increments of 100bps. This modem has been tested using a 2564B (600 lpm) printer. Using a low quality voice grade phone line the Trailblazer was able to drive the printer at its maximum speed of 13670bps. The Trailblazer sells for $2395/unit.

The third technique that is used is called asymmetrical. One high speed channel is used at 9600bps and one low speed channel is run in the

reverse direction. The theory is that nobody can type faster than 15cps so why allocate more bandwidth than is necessary. The asymmetrical modem will turn the line around when the user starts sending large blocks of data from the terminal, (e.g. block mode or PC data transfer). Two companies make asymmetrical modems: The USRobotics HST 9600 and the Caterfone ADCoMM PLUS. The ADCoMM PLUS uses 150bps on its secondary channel and costs $2000/unit. The HST 9600 uses 300bps on its secondary channel and costs $1000/unit. The HST use TCM to modulate the carrier.

The fourth technique is to take a conventional V.22 bis 2400bps modem and use a four-to-one data compressor between the modem and the DTE device. TELCOR makes a modem called the ACCELERATOR that uses this technique. The modem works very well at 9600bps when hooked up to the HP 3000 in both character and block mode. The modem runs in true full duplex all the time. Actual throughput is 7000-8000bps. The cost/unit is $1000. The modem will also run as a 212A and as V.22 bis without compression. The modem will allow the terminal user to use type-ahead.

Synchronous Modems

Synchronous modems have separate transmit and receive circuits which perform modulation, demodulation, and clock for the DTE. The transmit circuits encode the data from DTE on the carrier and provide transmit clock for the DTE. The receive circuits decode the data from the carrier and recover receive clock from the sidebands of the carrier. Synchronous modems are normally used for CPU to CPU communications and CPU to network communications.

2400bps

A 2400bps modem is normally used to communicate from within the USA to overseas locations. The defacto standard is the Bell 201C. The 201C is a dial-up half-duplex modem. The 201C uses phase modulation. A carrier of 1800 Hz is phase shifted to one of four angles in response to two bits of data from the DTE. The 201C is a good modem to use when calling overseas because it works very well on noisy phone lines. If you are buying the modems on both ends, a full duplex modem such as the Concord 224 (V.22bis) will run much faster because of no line turnaround. If using satellites, stay away from half-duplex modems such as the 201C. The line turnaround delays will be unsatisfactory. Price range of the 201C is $700-$800.

4800bps

A 4800bps modem is normally used in the USA for dial-up access from CPU to CPU. The defacto standard is the Bell 208B. The 208B takes 3 bits from the DTE and encodes them on the carrier as one of 8 phase angles. The standard used in Europe is V.27. The 208B and V.27 are not compatible with each other. The two newest modems in this market are the Paradyne 208B and the Racal-Vadic 4850. The Paradyne 208B has a built-in 801 Auto Call Unit (ACU), an integral Bit Error Rate Tester, and an integral eye pattern generator. The Racal-Vadic 4850 has a built-in 801 ACU, and can respond to dialing commands issued in BSC, HDLC, or SDLC. The 4850 can run as either a V.27 or 208B modem. The 4850 can be upgraded to 9600bps, but you lose the 208B function. Cost of both modems is $1295/unit.

9600bps

A 9600bps modem is normally used for CPU-CPU and CPU-Network communications. The standard for 9600bps on a four-wire leased line is V.29. Most V.29 modems made today are compatible with each other and can be ordered with an optional built-in four channel mux. The V.29 modem uses a carrier frequency of 1700hz. The modem receives four bits from the DTE and encodes them on the carrier using eight phases and two amplitudes. The Paradyne V.29 modem is used by Telenet for most of the host connections to the Telenet PDN. The Anderson Jacobson 9601-MD has a dual dial-up back-up feature. If a leased line failure occurs, the AJ modem will automatically dial two user-defined phone numbers to establish a switched four-wire connection at 9600bps. When the leased line is back in operation, the modem will automatically go back to the leased line. Cost/unit is $2000; without the dial back-up the cost is $1300. This modem is used for Telenet host connections with dial-up back-up. The Universal Data Systems 9600/A/B is a V.29 modem on a leased line. The modem will also operate in half duplex on a dial-up line, but is only compatible with itself. Cost is $2000/unit. Paradyne makes the HDX 9600. This will operate at 9600bps half duplex over dial-up lines. Fallback speed is 4800bps as a 208B modem. Cost is $2000/unit. The Paradyne HDX 12000 has similar features to the HDX 9600 but the data rate is 12kbps with fallback to 9600bps. Cost/unit is $3600. This modem will beat 9600bps full duplex performance, when traffic in one direction is at least four times the volume of traffic in the reverse direction.

14.4kbps

A 14.4kbps modem is used for CPU-CPU communications. Universal Data Systems makes the UDS 14.4 that uses TCM. Fallback speeds are 12kbps and 9.6kbps. At 9600bps the modem is V.29 compatible. Cost is $4000/unit. AJ makes the 1411LS which has the same dual-dial back-up as the 9601-MD. The unit also has a built-in six-channel mux. Cost is $5000/unit. The Paradyne Challenger 14.4 is available for $3600/unit.

16kbps

A 16kbps modem is available from Micom. The M40168 uses TCM and requires a D1 conditioned leased line. A dual-dial back-up is optional that runs at 9600bps when the leased line goes down. Cost is $5500/unit. The Paradyne Challenger 16.0 uses TCM and requires a D1 conditioned leased line. Fallback speed is 14, 12, and 8kbps. Cost is $3750/unit.

19.2kbps

A 19.2kbps modem is the highest speed modem that is available for a leased analog phone line without using compression. Paradyne has three models available that operate at this speed. The price range is from $4500 to $10000/unit. The Codex 2680 uses 64-state, eight-dimension TCM that positions the data in a smaller less error prone constellation. The 2680 has a built-in two-channel mux with an option to go to four or six channels. Two-line dial-up back-up is an option that will automatically back-up the leased line when it fails, and disconnect the dial-up connection when the leased line is back up. Price is $12000/unit.

Medium Distance

GANDALF is the only company that I am aware of that makes medium distance modems. The LDM 409 runs synchronously at 9600bps over a four-wire leased line. Range is 200 miles and the cost/unit is $895. The LDM 419 will also run asynchronously and the cost is $1095/unit.

Short-Haul

Short-haul modems can be used to extend RS-232C cable lengths up to 12 miles at a speed of 9600bps. Lower bit rates can run longer distances. Price range is $85 to $450. The BLACK BOX SHM-NPR attaches directly to the back of your terminal's RS-232 interface and gets its power directly from the interface. Range is 3.2 miles at 9600bps. Cost is $85/unit and two modems are needed per line. The customer can use his own two twisted pair cable or use in-house cable from the phone company. Bell calls the phone line a "Private Line Metallic Circuit". These circuits are described in Bell publication 43401 (tariff # 3081). A 3081 circuit does not have any loading coils on the line because most short-haul modems will not work with loading coils. The GANDALF LDS 309A will run 19200bps synchronous or asynchronous. Using 22 gauge wire the maximum distance is six miles at 19200bps and 12 miles at 9600bps. Cost per unit is $450.

4. CONCLUSIONS

Today's modem market is moving very fast. A 1200bps terminal to CPU link can easily be changed to 9600bps at a cost of only $1000/modem. Your dial-up DS/3000 access from 3000 to 3000 can be changed easily to 12000bps at a cost of only $3600/modem. If you need X.25 dial-up access into your own private network at 9600bps, the cost is only $2900/modem. The modems are getting faster; the prices are going down. Now is the time to take advantage of new modem technology!

Note: This paper represents the opinions of the author and not of Hewlett-Packard. Prices in the article are subject to change.

ARTIFICIAL INTELLIGENCE: A NEW TOY FOR SOFTWARE ENGINEERS

Maira Benjamin
ASK Computer Systems Inc.
730 Distel Drive
Los Altos, CA 94546

A new dawn in software engineering, Artificial Intelligence, is here. What was just a twinkle in someone's eye, is now a reality. And, just what is Artificial Intelligence all about?

When AI is mentioned, people immediately think of the possibility of making computers think like human beings by making them smarter. Hence, intelligence that seemingly is viewed artificial. Rather, I like to think of AI as an attempt at representing knowledge in a programmatic fashion by the combination of data and procedures. Yes, it is a start at making computers smarter. Stripped of its essentials, AI still boils down to programming. The magic lies in its attempt in the simulation of the human thought process.

At the heart of AI is the concept of intelligence. This is the ability to think, reason, acquire, and apply knowledge. Therefore, applications in AI deal with concepts and not calculations. As can be expected, the world of AI is not as black and white as the world of traditional programming simply because intelligence has an uncanny ability to adapt to changes.

AI is many things to many people. Some think of robotics, others think of natural language systems. But, the application that will affect software engineers the most is the expert system. The expert system can be thought of as a set of programs that houses the knowledge of an expert. Theoretically, if an expert system is programmed correctly, then it will be able to mimic the answers given by an expert. But, before we discuss any more attributes of the expert system, a little background on each of the foremost AI languages would probably enlighten us.

There are many variations on the major AI languages out on the market place. I will, however, limit the discussion to the general characteristics of each of the principle AI languages. The languages that I will discuss are: LISP, Prolog, OPS5, and Smalltalk.

LISP is best thought of as a set of functional paradigms. LISP is at the top of the list because it is the most popular AI programming language in use in the United States. It was developed by John McCarthy of MIT in the late 1950's. Most interesting is the fact that it is the second oldest programming language still in use (only behind FORTRAN!). Thus, LISP has benefitted from many years of use. It also has quite a few "dialects" such as ZetaLisp, Fanz Lisp, InterLisp, T Lisp, MACLISP. There is currently a push for standardization in the form of Common LISP.

LISP's functionality includes: independency of functions, rule mapping, use of control structures such as IF and recursion, and list processing primitives. LISP provides functions that extends the list processing to concatenation, building, extending, and dissecting. One can write LISP functions to manipulate other LISP functions. When stability for the different versions of LISP settles in, one will be able to see portability. A popular feature of LISP is its ability to let the user program in symbolic logic. A disadvantage of LISP is that it is parenthesis-laden leading to incomprehensibility.

Prolog, second only to LISP in popularity in the United States, is the favorite language of the rest of the world. It can be thought of as logic programming. This is usually based on a fact which proves a theorem which will tell us if a given fact is true or not. Prolog probably had its origins in mathematics. Components of Prolog include: English-like syntax and structure, integrated database facility, pattern-matching, some procedural components similar to that of conventional languages, and a built-in interpreter. Versions of Prolog in the market include: MPROLOG, Quintus Prolog, Prolog, and ESI's Prolog I.

Prolog's biggest advantage is its apparent ease of use. Code written in Prolog is more comprehensible than almost any other language. Thus, novice users can achieve results fast. And, many Prolog adherents claim that the code is easily ported from machine to machine even though there is no common Prolog standard. Another advantage of Prolog is that it uses less system resources than it's predecessor, LISP. Interestingly enough though, LISP proponents claim that one of Prolog's disadvantages comes from Prolog's biggest advantage--ease of use. They argue that to write anything that is complicated forces that user to learn the innards of the language. Another disdavantage to Prolog is the inability to find engines here in the United States.

Probably not as well-known as LISP and Prolog, is OPS5.

OPS5 can be thought of as a "production system." It consists of a global memory, which can store data structures, and a set of rules. The set of rules has a left- and right-hand side. The left side contains a set of condition elements and the right side contains actions. A rule is applicable only if ALL the conditions of the rule match. A feature of OPS5 is the use of "conflict resolution strategy" which will chose one rule. This is a definite advantage of OPS5--it has the ability to make unplanned responses. OPS5 is basically a computer program consisting entirely of "IF-THEN" statements.

OPS5 was developed by Charles L. Forgy at Carnegie Mellon University. It was designed for applications in expert systems and cognitive psychology. It encodes knowledge extracted from human experts in the form of "production rules." Production rules are a set of "IF-THEN" statements. A version of OPS5 is OPS83. There are claims that OPS83 is four to five times faster than its predecessor. It is an algorithmic language related to PASCAL and C. Other claims are that it is more easily transportable. An advantage to OPS5 and OPS83 is the ability to handle tens of thousands of rules. This makes it a plausible choice for the development of expert systems. Disadvantages of OPS5 and related languages are: the difficulty of being able to perform recursion and iteration, and the independence of OPS5 rules making it difficult to set up modules with related rules.

And just what is Smalltalk? Smalltalk can be thought of as object-oriented programming. This means that the program consists of objects and messages. Smalltalk is best suited for applications where there is a clear-cut hierarchy of objects. Smalltalk is also known for its modularity. Other components of Smalltalk include: data abstraction, and inheritence. In data abstraction, data-structure definitions are encapsulated in the procedures. This means direct manipulation of the representation of the data. In inheritence, properties that represent knowledge about an object can be encapsulated to other objects in the hierarchy scheme. Smalltalk is relatively young to AI even though its origins date back to the 1960's.

From all these different AI languages which are interpretive or object-oriented, we can generalize some facts about the AI environment. We can ascertain that the AI environment is:

1. Flexible
2. Imprecise
3. Integrated
4. Prototypical
5. Symbolic processing-oriented

AI is flexible because we can improve and add on to it. We can use a fluid design to follow a line of reasoning. This flexibility plays an important role when dealing with expert systems due to the fact that knowledge is easily updated and added to.

AI techniques will allow us to develop conditions that can approximate the human decision-making process mainly because AI lets us handle variable, ill-defined, imprecise programming parameters. Imprecision is inherent in the thinking process of experts. What happens when there is disagreement? It is not possible to "average" knowledge! Input will need to be critiqued in order to achieve any results. An underlying assumption with imprecision is the fact that an AI application will be able to learn from past experience.

AI is highly integrated. Many AI tools are built in and many languages and applications have traceability. This is the ability to locate objects of current interest. Traceability will let you account for relationships and attributes. This is a very useful feature for the user that happens to change a particular environment.

Rapid prototyping is possible in the AI environments. In fact, it is a given that testing an AI system is done in this mode due to the imprecision of the input and output. There are so many rules that can be activated that the only way to test an AI system is to prototype it. Good prototyping tools are essential to an AI system because the application can change quickly.

AI is a total symbolic processing environment. AI involves the programming of symbolic logic as well as numbers. One must remember the most basic tenet of an AI application: manipulation of knowledge. The very nature of knowledge involves the representation of it symbolically.

Now that we have discussed the basic characteristics of AI, we can examine the characteristics of the conventional programming environment. These are:

1. Rigidity
2. Precision
3. Lack of well-integrated tools
4. Minimal prototyping
5. Process numbers in a "black box approach"

At first glance, it seems that the conventional programming environment is the exact opposite of the AI environment.

This is true! Because of the Host/timesharing approach that many of the conventional environments use, rigidity is inherent in many of the applications. Rigidity implies that a precise knowledge domain is present. This is apparent because programming is done on data, not on knowledge as in the AI environment. There is also a precise definition of the steps that lead to a solution. Traditional methodology is ill-suited for the uncertainty that is prevalent in AI systems. There is no such thing as exploratory programming in the traditional environment.

With so much rigidity and precision in the domain of traditional programming, one would think that there would be some well-integrated tools available. In actuality, the tools are not so. Even worse, the structured languages allow minimal prototyping. The programs become difficult to change and allow few modifications due to the fact that logic is programmed right into the module versus the separation of logic and data in the AI system.

Using compiled-only languages, one can only follow complete pre-defined step-by-step procedures and algorithms which only process numbers, not symbols. Thus, a line of reasoning cannot be explained easily or traced. The "black box" approach is commonly used in conventional programming.

Even given all the advantages of an AI system, one should only choose to use it over traditional programming when the domain knowledge is not firm, fixed, or formalized.

We have discussed the two programming environments, AI and traditional. Now, we should examine how software engineering differs in each domain.

The most pre-eminent application of AI lies in the expert system. An expert system can be thought of as a computer program that combines a knowledge base and inference procedures in order to solve problems that need human expertise. The expert system needs two elements in order for it to do this: an expert and a knowledge engineer. We shall focus on the knowledge engineer because this is the software engineering part of the solution.

In the AI environment, the knowledge engineer is expected to:

1. Process an expert's knowledge and experience into an AI system
2. Build a knowledge base using an inferential process
3. Use heuristic reasoning to obtain rules

Because the expert system must possess expertise on the subject matter at hand, the knowledge engineer must be able to extract this information. The art of extracting this knowledge has still not been defined precisely. This introduces a level of complexity for the knowledge engineer. In fact, there are new AI programs being introduced in the market-place known as "knowledge-engineering assistants". The object of these programs is to interview the experts in an attempt to "learn" from them. However, the programs have demonstrated some form of rigidity in the approach they interrogate. So, this still basically leaves the knowledge engineer with this very important task to perform.

Besides "interviewing" the expert, the knowledge engineer must create a knowledge base with the appropriate inferences so that the system will be able to mimic the expert's responses and/or thinking pattern. Building the inferential engine takes a thorough knowledge of the expert's reasoning skills. That is why the interrogation process is so important to the knowledge engineer. Furthermore, it can take several steps to do this, and it is akin to the design phase of the traditional programming cycle. Some of the steps that a knowledge engineer can take in the interviewing process are:

1. Define the principal problems to be attacked
2. Develop a set of strategies that the expert uses to solve the defined problems
3. Develop appropriate inferences and relationships according to the expert's solutions to the problems

Even worse, experts sometimes have difficulty expressing their decisions given the problems that were to be solved.

After the interviewing phase, the knowledge engineer has to sit down and develop the knowledge base based on the inferences that were gathered from the expert. This stage is akin to the "production" phase of traditional software engineering where actual design/coding is done. The development of the inferential engine is accomplished using heuristic techniques to obtain the rules for the knowledge base. I would venture to say that this phase does not take as long as the interviewing process which is actually a continual one.

As the knowledge engineer refines the rules, he/she usually presents this to the expert for further "fine tuning." Thus, the whole cycle is a circular one in nature. This is very easy to do because of the ease of prototyping in the AI environment. And, since, one is not programming the

logic into the program itself, the program is easily modifiable to access even more relationships and rules in the knowledge base.

Because knowledge is not static data, many knowledge engineers find that maintenance is an on-going process even after the product has been "released."

What are the steps that software engineers go through in the traditional programming environment? These steps are probably very familiar to most of you, but I will identify and discuss them nevertheless:

1. Identify the problem in order to specify the objectives.
2. Design the program to solve the problem
3. Code the program
4. Test the program in order to release the product

Just as the knowledge engineer had to "interview" his expert, the software engineer must gather data and knowledge about the problem at hand. This process will help him/her specify the objectives surrounding the problem. The big difference here is that the software engineer is working with data. Remember that the data the software engineer is working with is not like the knowledge that the knowledge engineer is working with. It is more static in nature and is not incorporated into the program unit itself.

In the designing phase, the software engineer begins actual design of the program and any of its interfaces to the outside world. This phase is part of the "production" environment in the traditional programming world. It is here where a lot of fine-tuning takes place and it is here where we get into the actual "nitty-gritty" of how the problem will be solved. The "interviewing" is just about done and over with by the time the software engineer finishes this phase.

I think that we are all familiar with what goes on in the coding phase. Simply put, the program is coded! The relationship between the logic of the program and the data has already been established and put rigidly into place. Ideally, if the design has been done up-front with enough research supporting it, there is little chance that any major parts of program logic will change.

Finally, the program is tested before it is released. Testing here just verifies that calculations are accurate,

representation of data is correct, and retrieval of information is done in an orderly and predictable fashion. Testing usually consists of boundary and path checking of the conditional phrases in a program unit. This is in sharp contrast to testing in the AI environment which is done explicitly by prototyping and verification.

Maintenance of traditional programs? Well, logic dictates that if the identification and design stage were done correctly, maintenance of the program is minimal. But, if there is a change, the software engineer usually has to change the problem-solving logic--sometimes necessitating a complete new start.

At first glance, there may not seem to be very many differences between knowledge engineering and software engineering. In fact, the process is almost the same. But, there is more emphasis on the design/testing stage in knowledge engineering. That is because verifying the validity of rules is not easily tested except through prototyping. The knowledge engineer slowly evolves his system; the software engineer produces the system.

The area of maintenance is also where one can find vast differences between the two styles of programming. AI demands much more in the way of maintenance because new rules and relationships continuously form new pathways in the knowledge base. Data is of a static nature in the traditional programming environment.

We have discussed and investigated the similarities and differences between knowledge engineering and software engineering. Although knowledge engineering is currently different from software engineering, all software engineers will evolve into knowledge engineers. This is because AI will slowly invade all domains of programming.

In fact, the invasion is taking place now!! One of the complaints that software engineers have voiced for many years was that the coding phase of programming was rather tedious. Tedium stand aside! There are code generators now available thanks to the science of AI. This is one of the newer applications that will definitely have an impact upon the software enigneering cycle. If code generators become readily available, this will give software engineers more time to spend on the design phase.

As mentioned before, not all problems are candidates for AI programming. Many AI languages are now attempting to integrate traditional programming languages like FORTRAN,

COBOL, C, and PASCAL. This opens even more doors for development of AI systems. It may even help software engineers cross over into knowledge engineering and the world of AI.

AI is currently very well suited for those problems that need human expertise where the domain is not exact. AI has found a niche in determining VLSI design. It is also being used to create natural language systems (similar to the "talking software" concept).

And, finally, we dare to ask the question, "Will machines ever think like man?" Although, the answer to that question is beyond that scope of this paper, it seems to lie underneath everyone's breath whenever AI is mentioned, doesn't it? The answer, I believe, lies in the response to following question, "Will man ever understand how his mind functions?"

AI is here and it will stay no longer just a twinkle in someone's eye.

REFERENCES:

Barr, Avron and Feigenbaum, Edward A., The Handbook of Artificial Intelligence, HeurisTech Press, Stanford, CA, vol. 1, 1981.

Bishop, Sharon, "A Siren's Song of the Future," INTERACT, June 1986.

Brower, Brock, "Machine What Think," California Magazine, July 1984.

Davis, Dwight B., "Artificial Intelligence Enters the Mainstream," High Technology, July 1986.

Epstein, Jonathan A., "Intelligently Speaking," Digital Review, May 1986.

Meredith, Dennis, "Why You're Smarter Than Any Computer," High Technology, July 1986.

Miller, Richard K., Artificial Intelligence: A New Tool of Industry and Business, SEAI Institute, Madison, GA, 1984.

Patton, Carole, "Knowledge Engineering: Tapping the Experts," Electronic Design, May 2, 1985.

Ramamoorthy, C.V., et al, "Software Development Support for AI programs," Computer, January 1986.

Waterman, Donald A., "How Do Expert Systems Differ From Conventional Programs?" Expert Systems, January 1986.

Developing Large Scale Applications Using A Fourth Generation Language

Kenneth Bekkering
Michigan Education Data Network Association
1350 Kendale Boulevard
East Lansing, MI 48826

Using techniques and technology currently available only through a fourth generation language (Synergist), applications can be written which are far superior to conventional online programs. This paper details the techniques and technology used in writing large complex applications to achieve a new level of online programming standards.

People have their own definitions of what they consider large applications. I am not going to say who is right or wrong, or where the borderline is that determines if you have a large application. However, I believe you will agree that our shop has a large application.

We have 120 work stations in three different buildings. They process 3000-4000 insurance claims (checks) per day. To support these users we have huge databases with some datasets that have two to three million records in them. This is an online system with most updates occurring immediately. From the work stations 50,000 online transmissions (pressing the enter key) per day are done. Some of the work stations are shared by two people, so that they can alternate between doing desk work and entering information onto the system. It is safe to say that the application is large enough and complex enough that it would not work using a typical third generation approach employing COBOL, V/3000 and 120 asynchronous terminals.

One of the major reasons that fourth generation languages are becoming more and more popular is that they allow the programmer to produce a finished product faster. The faster an application can be developed, the cheaper it becomes to produce the application. What we have to be careful about is to make sure that this faster development for a relatively short period of time is not offset by other factors during the life of the system, which, hopefully, will be many years.

One of the tradeoffs of fourth generation languages is that they typically use up greater amounts of computer hardware resources to produce the finished product than a third generation language would. This additional CPU requirement might make it necessary

to upgrade to a larger, more expensive system. Remember, this is not necessarily bad. If you can save $100,000 in programming costs and only have to pay $50,000 for additional hardware, you are still coming out ahead. However, the results become questionable if you save $100,000 on programming costs, but must spend $100,000 or more for additional hardware.

Through the past several years, we have seen the cost for personnel rising every year and the cost of hardware decreasing every year. Also, fourth generation languages have been maturing. Because of these factors, the point where it becomes economical to convert to a fourth generation language has been reached by more and more companies.

Several years ago companies went through a similar change as they moved from their second generation Assembly programs to third generation programs such as COBOL. In doing this, the programmer gave up some of the control over what he was doing and let the machine have more control. Because the compilers available for third generation languages did not always do a good job of optimizing the code, they did not create code that was as fast as that written by a good second generation programmer. Because of this, going from generation two to three also required additional CPU power. However, the savings in programming were great enough that only a few companies still use a second generation language as their primary language.

Another one of the typical problems is that for large applications, you do not want to outgrow the machine that you are running on. If you have an application that is running on an HP3000/70 and you do not have enough CPU available for everyone, you are in a tight spot. You cannot move up to a larger HP. This leaves you two options.

The first option is going to multiple systems. A distributed system is very complex to implement and manage. This extra headache is something that many people will go to great lengths to avoid.

The second option is to move to some other system besides an HP. This is an expensive, time consuming task, but I know of several shops that have taken this approach instead of option one.

However, we were able to implement a large application on a single HP using a fourth generation language. We were well aware that the HP3000/70 did not have enough horsepower to do this, so we looked for alternatives. We needed to keep our databases on one machine, so we could not buy an additional HP(s).

HARDWARE REQUIREMENTS ARE MORE COST EFFECTIVE

What we did is invest in Personal Computers. For every $1.00 spent on a PC you get considerably more for your money than if you had spent it on HP3000 hardware. For example, the cost per megabyte of main memory is cheaper on a PC. Also, the cost of disk storage on a PC is cheaper.

Now that we had these PC's, we wanted to take as much of our processing off the expensive HP and put it on the PCs. We needed a central database that could be shared by everyone, so we had to leave that on the HP. What we did move to the PCs included:

- All online application development.
- Application execution.

The applications execute on the PC and only talk to the HP when they need to retrieve some information or send back some updated data. This turns the HP into a big file server during the day. The following examples will show how this reduces the work load on the HP.

A) Change a customer record with a typical COBOL/VIEW application:

1) HP reads information from database.
2) HP formats information.
   - Changes codes to text.
   - Formats information for screen layout.
3) HP sends information to terminal.
4) User makes necessary changes - sends it back.
5) HP edits data to see if valid.
6) HP sends it back to user if incorrect.
7) HP reformats it into something usable.
   - Changes text back to codes
     (for example shipping method changed back from text to a two digit code).
8) HP updates database.

A total of 7 steps on the HP.

B) Changes using integrated HP/PC approach:

1) HP reads information from database.
2) HP sends information to PC.
3) PC formats it nicely putting it into screen layout.
4) User makes necessary changes.
5) PC edits data to see if valid (If invalid, user must correct. No bad data gets back to HP.)
6) PC reformats it into something usable
   - changes text back to codes.

7) PC sends data to HP.
8) HP updates database.

A total of 3 steps on the HP

While each step does not take the same amount of time, the total saving of time on the HP is significant.

This approach can be used with either third or fourth generation languages. It is very feasible to have a COBOL program running on a PC that is sending and receiving information from another program on the HP.

However, we decided to take the route of a fourth generation language. By going with a fourth generation language, we were able to generate systems faster than before, and these systems looked nicer to the end user.

Now we are using a fourth generation language. With it we are using additional CPU. However, the vast majority of the total CPU use is now occuring on the PCs. This means that we actually have a significant decrease in the amount of processing that is being done on the HP. This is very important because at this time, the HP is our critical resource that we do not want to outgrow. As we add additional users to the system, there is very little additional load put on the HP. In doing some performance analysis, it was determined that our average work station uses only two minutes of CPU time in a full work day. Because the HP is not bogged down, the users get excellent response time and are therefore more productive.

MORE EFFICIENT USE OF HP3000 AND PC

Many organizations are buying PCs and incorporating them into their organizations. The uses typically include terminal emulation and standalone applications such as word processing and spread sheets. While you are getting more functionality out of the PC than you would out of a terminal, much of the power of the PC is unused.

By going one step further with the Synergist, the PC is no longer working just like a terminal. You are using the additional power on your desktop that is normally sitting idle.

In our case, having 120 cables running over three buildings would be very difficult to install. We went with a new approach that the Synergist supports. We have several local area networks with 20 to 30 PCs attached to each. These machines on each network are connected together with coaxial cable. On each

network, there is one machine that acts as a gateway or bridge machine going into the HP. This allows us to have all the PC's in one work area connected together and only have one wire running from the work area to the HP. This results in a tremendous cost savings in buying cable, stringing the cable, and buying additional ports on the HP to plug the cables into. Because of the high speed of the network and data compression done by the Synergist the users are not aware that they are really sharing one line.

BETTER RESPONSE TIME

With Synergist, the processing is spread over many machines (one HP and many PCs). Because of this, each of the users get more total CPU and therefore better response time. In our testing, the response time is at least as good as would be possible with COBOL/V3000 and in most cases better.

Because all the logic is on the PC and the only processing on the HP is the retrieval and storage of data, there will be less information being passed back and forth. With a terminal running at 9600 baud, it takes about two seconds to pass the information for a full 24x80 screen from the HP to the terminal. Of this 1920 bytes of data being passed, 600 might be actual data, and the remaining 1320 might be the screen layout to make a nice presentation. The only thing that is passed with the Synergist is the actual data. The screen layouts are always on the PC. In addition to that, the Synergist does compression, so only about 400 bytes of information would be passed. This is about 20% the amount of information that would be passed with a terminal. This would decrease the actual transmission time by more than 1 1/2 seconds.

Sending information back to the computer is even faster. Synergist determines what information has changed and only sends that back. Therefore, if there are 20 fields on the screen and only 1 changed, only that one field would be sent back to the HP. With the way data is passed back and forth, Synergist works quite well on lowspeed dialup lines.

Because the application is running on the PC, the configuration of the PC also affects the response time. Configuration options that affect response time are the amount of memory, the cycle speed of the processor, and the speed of the hard disk. However, even if you go with a lower cost, lower performance PC, you will still come out ahead using the Synergist.

USER FRIENDLY

The final application makes it almost impossible for the user to make a mistake. Using earlier technology, the user would send in a screen of information, and all fields in error would be highlighted for the user to correct. With Synergist, errors are detected at the keystroke level. For example, if a field can only have a value of 1-4 in it, the system will not allow you to type in anything besides those 4 numbers.

Another feature that makes the final application user friendly is the conversion of codes. Internally in the dataset a code is stored. In the past, users would have to have a manual describing what the codes meant, so they would know what to type in. One example would be a employee position list, where you wanted to know what position a person held. Valid choices might be:

1) Director
2) Manager
3) Systems Analyst
4) Programmer
5) Computer Operator
6) Clerical Staff

Using the typical approach, the person would have to look in a manual to see what code went with what position. Of course, after working with the system for awhile, you would get to know the codes. However, this still does not make it very easy for the new user or the user that uses the system infrequently or on lengthly code lists.

With Synergist, the conversions are done automatically. If a person has a code of 4, it is automatically displayed on the screen as "Programmer". If a person wants to change it to "Systems Analyst", they only needs to types in an "S." Because only one word starts with an "S", the rest of the description is automatically filled in. If the position is changed to "Computer Operator," "CO" would have to be typed in. By typing in only "C," the system would know you wanted either "Computer Operator", or "Clerical," but not which one. If you ever want to see all the valid positions, they can be displayed in a window on the screen by pressing a function key. From this window, you can move the cursor to the desired selection and press enter.

An additional feature that makes the system user friendly is the ability to conditionally allow a person into a field on the screen, depending on some criteria. For example, in a membership system, you might have several fields related to a certain type of membership. If a person does not have this type of

membership, you would not want to tab through all of these fields. You can have them automatically protected, so that the cursor will jump right pass them. You could also go one step further and not even display those fields on the screen.

EFFICIENCY IN CREATING APPLICATIONS

One of the key concepts that makes it easier to create applications with a fourth generation language is the use of a centralized dictionary. This dictionary should have as much information in it as possible, so that it is not necessary to recode the same logic each time a certain piece of information is used. With the Synergist, the dictionary contains the standard things you would find in any language, plus some additional ones. Some of the dictionary items are:

A) Elements, fields, and files.
B) Security
C) Prompts. A one line prompt that appears each time you enter data into a field.
D) Extensive edit checks.
    1) Optional / Required.
    2) Numeric? Range if numeric. Display format.
    3) Alphanumeric? Which characters are valid?
    4) Select codes? Define the codes, such as "Male," "Female".
    5) Additional edit criteria that you can code.
E) Error messages in case of user errors.
F) Online help screen information.

By putting all this information in the dictionary, we do not have to continually recode it each time we use it, or make a change to it. For example, ACTIVE-DATE might show up on 10 different screens with the same edit check each time. When we put the field on a screen, all we have to do is enter its name. All the logic is picked up from the dictionary. Also, if it is necessary to change the logic on ACTIVE-DATE for some reason, the change only needs to be made in the dictionary. No longer do you have to do lots of searching all over to find which programs use the data item and need to be changed.

Another advantage is that many of the errors that are usually picked up during compiling are caught immediately as you are developing a screen. For example, the system will tell you if you enter a data name that does not exist. If this is a typing error, you can display all similar names and pick the actual one. This makes it so that you get a clean compile more often and speeds up the development by eliminating additional compiles.

MAINTENANCE IS MUCH EASIER

Maintenance is easier with a fourth generation language because there is less code to work with. In fact, there is very little actual coding with Synergist. Most of the work is painting the layout of the screen, and then going through a menu system to answer questions and prompts about the different attributes about each of the fields that are on the screen.

The design of Synergist forces the screens to be structured which leaves fewer decisions up to the programmer. Due to this structure it is much easier to make a change to an individual piece and not have to take as much time to familarize yourself with the things occurring elsewhere on the screen.

Our programmers agree that if they have to do maintenance to an unfamilar Synergist screen or unfamilar COBOL program, they will take the Synergist screen. They can get in and make the change faster, and it works correctly the first time more often.

IMPROVED INVOLVEMENT OF USERS IN DEVELOPMENT AND MAINTENANCE

Due to the ability to quickly change the layout or edit criteria of the screens, it is much easier to get user involvement and get feedback during the actual development cycle. This way the user can make suggestions for changes ranging from cosmetic changes to much more elaborate changes. With the flexibility of Synergist, changes can be made in minutes instead of hours.

By involving the user in this initial phase, there are several advantages. First, the user gets the feeling that this is his application. He helped develop it and will have a much more positive attitude when the final product is installed. He will not resent something new being forced on him.

Secondly, the application will match the user's needs better than if user input during development was not used. This means that there should be less maintenance to the system after it is installed, because it closely matches the user's needs. How often have we written a new application and then had to go back and rewrite it again to give the end users what they really needed? The key here is to do it right the first time, so you do not have the application haunting you forever. With fourth generation languages this is much easier than in the past.

CONCLUSION

Fourth generation languages have a great potential for decreasing the amount of work needed to bring up a new application.

Unfortunately, due to limitations in the fourth general language programming languages and their use of additional machine resources, they are not used for large scale applications very often. We feel that by using a fourth generation language, such as the Synergist, you can get around these limitations and bring up applications that are better and take less time to develop than using traditional third generation languages. We feel that at the present time, there are no other fourth generation languages that offer the features and functionality of the Synergist.

COMPUTER SECURITY: The Best Defense is a Good Offense

ISAAC BLAKE
Intel Corp
145 South 79th Street
Mail Stop SP1-78
Chandler, AZ 85224
(602) 961-5053

INTRODUCTION

Each day concern over computer security is increasing. As more and more businesses are becoming victims, they are finding that they must take steps to insure their system integrity. Unfortunately, most of these business only invest in securing their systems after they have been victimized, and not before. They are learning, like the rest of us, that the days of not securing your house, car, and business are gone. Because not taking steps in protecting these assets, will make them easier targets for the would be criminals. This paper will attempt to cover ways to secure your computer system, and reduce the chance of being a target.

WHY IS COMPUTER SECURITY IMPORTANT?

As fundamental as this may sound, often you will find people wondering why they have to spend the money, or put up with security precautions. This issue is an important one, for if you do not get the support of implementing security at all levels, it will be difficult to have a fully secure system. Take the time to explain to the people involved the reasons for the security restrictions. If you explain about the privacy act, copyrights, and critical business data, most people will understand. Also, make them aware that they are critical in the success of the the system. You should be aware that the human factor most times is the weakest part of a security system.

WHAT IS COMPUTER CRIME?

It is important first to realize that these are crimes, not a joke. For too long, we have chuckled over teenagers causing havoc by cracking into computer systems, or enjoyed watching the "fun" of overcoming computer systems as in the movie "War Games". Until recently, these were viewed as harmless pranks, or headaches. But businesses are losing millions of dollars each year to hackers stealing time from computers, pirates cracking access codes to use long distance lines free, criminals using unauthorized credit

card numbers to purchase items, and employees misusing their computer systems. The end result, is that us, the consumer, must pay for these acts.

Most states now have added sections to their criminal laws dealing specifically with computer related crimes. Many states have made certain sections felonies, and even minor areas a misdemeanor. In Arizona where I live, it has one of the most progressive and aggressive computer laws in the nation. Under ARS 13-2316 (Arizona Revised Statues) it makes accessing, altering, copying damaging, or destroying without authorization any computer, computer system, computer software, program, data, or network a class 6 felony. If such actions are done in a fraudulent manner it's a class 3 felony. A class 6 felony is punishable by up to 1.5 years in prison, 3 years probation, and up to a $50,000 fine. A class 3 felony is up to 5 years in prison, 7 years probation, and up to a $150,000 fine. In short, Arizona like other states have realized the problems with computer related crimes and are committed to stop this type of crime and to punish the violators.

You should find out what laws are established in your area. They are usually grouped into the following areas:

Trespassing

> This can be either physical or remote access to your system. If a person accidently accesses your system, and doesn't immediately log off, they are guilty of trespassing. If the person knowingly accesses your system without your permission, or continues to access your system, it is a crime.

Theft

> There are several areas of theft, there is the traditional taking of software and/or hardware, but also includes theft of information and theft of services. Service related thefts are items such as moonlighting using company equipment, using timesharing facilities that you are not entitled to, unauthorized access to data/tele communications. A very popular area is employees taking software from their company. Often it's utilities, not major software packages that are stolen. These utilities are often considered "trivial" in nature and value. This may include items like UDC files, home grown utilities, copying PC software, etc. Employees should understand that this is as much of a violation as stealing a payroll system.

Fraud and Embezzlement

These are the crimes that you usually see in the papers. About people who alter computer programs for their own benefit (divert funds, altering payroll, A/P, A/R, personnel, and credit history records).

Tampering and Sabotage

Any changing of software or data is usually a felony. This includes such actions as changing customer data, leaving obscene messages, or logic bombs. Logic bombs are instructions that are placed into a program so that if a certain criteria is not met, the program will blow up. Often these bombs are placed by people who are leaving the company or very dissatisfied. There has been several cases of programs "blowing up" after a certain individual has left a company.

Civil actions

Most actions in this area are the invasion of privacy, but can also be breeches of contract such as non-disclosure and non-competition agreements.

Most state, county, and local police departments have started their own bureaus that work on computer related crimes. You should contact your local law enforcement agencies to determine if they have such a bureau.

WHO DOES IT?

As with all crimes, the victims wonder why they have to protect their property, and why the police don't spend more time catching the criminals, rather than teaching people how not to be victims. The answer is simple, since criminals don't go around with "I'm a criminal" tatooed on their forehead it's hard to identify them, where it's easy to identify a potential victim.

The computer criminal can be anyone, a Chairman of the Board, data entry clerk, cleaning crew, ex-employee, teenager, a competing business, the list goes on and on. These people can cover a wide range, from a hard core criminal who's trying to steal millions, to those who don't realize what they are doing is wrong.

THE BEST DEFENSE IS A GOOD OFFENSE

As with any basic crime prevention, the idea is to make your site, like a bank, car, home, or business, so difficult to get into, that the criminal will bypass you for an easier target.

The foundation for this is a security plan that's integrated into your business and operations. There are many ways to develop such a plan, go through a EDP audit, hiring a computer security consultant, or doing it yourself. The important thing is to develop and have a plan, rather than being without one.

YOUR SECURITY PLAN

First you must identify what areas need to be secured. This can be types of data or programs, backup tapes, physical access to the computer system, etc. Once identified, then classify them into levels like public, company use only, restricted, and confidential.

Then determine who needs to have access to the area. Develop a threat potential and action plan for each person. For example, what needs to be done if a data entry clerk leaves verses a system manager. Often when a person leaves a company, they could still gain access to the system, even a year later, because their logon and passwords are not changed or removed.

Look at putting functions into seperate compartments, along with dividing responsibities and authority in different areas. I.E. operations, system management, program management, data management, and communications. Whenever possible make it so that it would take a conspiracy to violate security. For example, one person to be the operator, another to be the tape librarian.

Make sure to evaluate two major threat areas, internal verses external. The external threats are from persons outside your company, such as ex-employees, hackers, competing business, etc. However, the greatest threat will be from the internal user. This can be anyone who has legitimate access to your system, like the data processing staff, consultants, and other employees.

Your plan should also include non-computer related areas; personnel, policies & procedures, building security, employee and non-employee identification.

Develop hiring practices; do background checks, inspecting all references, and the signing of agreements such as a code of ethics, non-disclosure, and non-competition agreements.

Also look at termination proceedures. It might be a good idea as part of your exiting procedure to review security related issues. You would be surprised the effect of reminding the employee that if they remove any software or data without permission they will be prosecuted. Also it's

a good time for the employee to take, with permission, any software they might want. I found in my prior jobs that if I asked my manager for some software they were in all cases willing to give it to me. I also had the manager give me a signed letter stating it was ok for me to have the software.

COMMON WEAK POINTS IN A SYSTEMS SECURITY

MPE has very good security, if you take the time to set it up correctly. Most security breeches on the 3000 are due to human failure, rather than design failures. When I worked at HP's Response Center, I had the opportunity to dial into dozens of different systems each week. It never ceased to amaze me the number of systems that have no security, or just have the default passwords on their systems, such as the default passwords for the TELESUP account.

This leads me into the first weak point of security, being predictable. Make sure that you change and/or add passwords to common accounts like TELESUP, PACHnnnn, SUPPORT, SYS, HPPLnn, HPOFFICE, ITF3000, VESOFT, REGO, etc.

Next, identify users and accounts that have SM, PM, or OP capabilities. If a user has ANY of these capabilities they can totally compromise your system in a short period of time. Within this, identify any programs that require PM capabiltity to run and make sure that they are not released and have execute access only.

Look very carefully at ANY released files on your system, especially if they are program files with PM capabilities. These files, even with lockwords on them are very vulnerable. In my opinion, a released PM program file is suicide to your computer systems security!

Setup a program that is run as part of your system logon UDC. This program can enforce ANY security specifications that you setup. Such program should notify you if there is a security breech and keep a log.

Speaking about logging, turn on all of the system logging. MPE system logging provides alot of valuable data about who did what, when. You can then store the logfiles to tape using either :STORE or LOGSNAP from the Telesup tape.

As far as creating users and accounts, make sure if at all possible, to give every user a unique logon and password. Trying to track down who did something when 20 people could log on as USER.APPL is difficult. If you have to use a common logon (I.E. MANAGER.SYS, OPERATOR.SYS, etc), make sure that the people use a different job or session name.

Guard your dial up lines. If your shop is not a 24 hour, 7 day a week shop, a person could dial into your system and it would be a couple of days before you found out what happened. If you don't need the dialup line, unplug it or down the device(s) until you do need it.

Protect your tapes. If a person has access to your backup tapes they can get any information about your system. They can use FCOPY to get your password, or take the tapes offsite and :RESTORE your files. Make sure to enact a policy on possession of tapes, and removing them from your site. Be sure to erase the tape when you are done with them, especially if they are going off site. I've always found it interesting to get a scratch tape from a person then running VALIDATE or FCOPY against the tape, to see what software or data the person has given me. I'm sure you can understand the implications.

Another area to consider is data encryption. There are several ways to encrypt data, programs, and communications to ensure privacy. Surprisingly, there has been several cases reported of persons hooking into telephone lines to get information. If you encrypt your sensitive data, this would reduce the chance of a person using something like QUERY or DBDRIVER to look at your data.

Lack of maintenance, alarm bells, etc is another weak point in security. If you have a violation, make sure someone knows about it immediately. Adopt the "land mine" philosophy of security reporting, so that if someone trips or violates security, it's well known.

Eariler I covered that for Trespassing, a person has to know that they are not suppose to be on the system. An easy way to handle this is with either the system message catalog or a :WELCOME message. The follwing message is an example of such a message:

```
**************************************************************
*                    Welcome to MY SYSTEM                    *
**************************************************************
* This is a private system operated for XYZ Company          *
* Business ONLY! Authorization from XYZ management is        *
* required to use this system. Use by unauthorized persons   *
* is prohibited and may result in prosecution.               *
**************************************************************
```

WHEN YOUR PLAN IS FINISHED

It is important to identify and layout your security system first. Once done, you can then go out and look at the numerous security systems available. Publications like

Interact, The Chronicle, SuperGroup, etc, have page after page of advertisements of vendors dealing with security. There are several companies that specialize in security. Also, HP's new system security package, which is available with V-MIT, is worth looking at. Most of all, make sure the security system meets your needs, rather than you meeting the security system needs.

Once you start pulling the pieces together to form your security system, have yourself and a few others attempt to crack the system. This sounds like encouraging people to do what you don't want them to do, however if the system is sound, it will stand it. Remember, the criminal who's has his sights on you will try anything. An example is a very good security system I was evaluating once, it did a great job of enforcing the security I desired, but I was able to breech security by logging on as a job from my terminal (instead of :HELLO USER.ACCT, I did a :JOB USER.ACCT).

Don't think that once your security is in place, that is all you have to do. Security, like system management, is an ongoing process. You will have the constant maintenance of userid's and passwords. Most of all is the inspection of the logs and the security systems performance. Look at the military, police, and private security layouts, they all include human monitoring no matter how sophisticated the security design.

As much as they are a pain, go thru yearly EDP audits. It helps to have a outside view of your system. Plus it's a good reason to make sure that everything is in place. Of course you should be keeping on top of things on an ongoing basis, shouldn't you!

WHAT TO DO IF SECURITY IS BREECHED

You must accept that no matter how secure you make the system, at some point it will be violated. The reason to install security is to reduce the chance and minimize the loss of a breech. Hopefully your system was designed with alot of traps, mines, and logging to help you detect problems. Once you have detected a problem, try to identify the basics, like who did what when and where. Your plan should of included an action plan covering most possible occurances.

The basic duty is to collect evidence. This will be the listing of your logs (both system and security) and any additional supporting documentation. If the violation involved the changing of data, make copies of the altered and normal data. In a case of a theft, identify the items, information, or services stolen and their appropriate value.

A very important point in the collection and preservation of evidence, is to secure the evidence in a safe place and maintain a chain of evidence.

If you do not specifically know who did the violation, then keep the knowledge of the violation to a "need to know" basis. This is to allow you time to gather more information and watch for clues. However, if you know, or at least feel you know, who did the violation, question them. You unlike the Police, can question them without having to give the person their rights, and then you can testify in court. A police officer must give the person their Miranda rights (right to remain silent, attorney, etc...) before any questioning.

Bottom line is to PROSECUTE, whether criminally and/or civilally. Only prosecution will deter these activities. Further by gaining convictions, other businesses will be able to identify these criminals in their background checks. This will keep the problem from passing from one business to another.

CONCLUSIONS

Although not deliberate, most computer sites victimize themselves by not taking the time to include security into their computer operations. Often security is viewed the same as documentation, something that's needed, but you never seem to get around to doing it. Further many businesses will spend thousands of dollars in physical security to protect their building and assets, but very little on computer related security. However if you take the time to evaluate and implement a security system, you will significantly reduce the chance of your business becoming a victim and suffering the loss associated with it. By taking the offense to the computer security issue will provide you with the best defense.

BIOGRAPHY

Isaac Blake is currently a Technical Support Engineer working in Intel's Technical Support Group. This group is responsible for supporting Intel's HP and VAX systems on a corporate wide basis. Prior to Intel he worked for HP as a Senior Support Engineer at the Response Center. His experience covers 16 years in the computer industry. The last 11 years was working on HP3000 systems mainly as a system manager. He is also a reserve police officer who has dealt with investigation and prosecution of computer related crimes.

A USER-FRIENDLY VIEW OF VIEW

By Gail Bird

EG&G Idaho Inc.

P.O.Box 1625

Idaho Falls, Idaho 83415

Phone (208)526-0590

A USER-FRIENDLY VIEW OF VIEW

By Gail Bird
EG&G Idaho, Inc.
Idaho Falls, Idaho

Often users of screens laboriously enter information for a large number of fields within a screen, triumphantly smack the "enter" key, then wilt in their chairs as they watch their efforts disappear while a flashing window message at the bottom of the screen announces an error in the first field entered.

The VPLUS-3000 (View) father-son relationship for forms can be a very useful tool for assisting a user in entering only the minimum necessary amount of information on a screen. Father-son (family) relationships between screens involve "look-alike" screens with distinctly different attributes.

Two concepts which have proven very useful deserve further discussion. The first involves making nearly all fields within a screen unenhanced "display" fields. The only fields left accessible to the user are the "key" fields - those necessary to complete the first step of the transaction. As successive steps of the transaction are completed and approved by the editing program, these "accepted" fields are "locked up" and additional fields are opened up. Enhancements such as underline or inverse video are turned off in the "locked" fields and displayed only in the optional or required fields of a particular son. This same idea is useful for allowing multi-purpose screens with only those fields for a particular application being opened for input. It is also advantageous where looping is desirable on only a portion of a screen.

The second concept involves using the "family" relationships to cause not only protected fields, but also their related labels on the form to appear and disappear as the user progresses through a transaction. This is accomplished by creating label fields as protected, display-only fields having values added and/or removed on son forms. Corresponding enhancements are also changed from son to son.

"As a source data entry system, VPLUS/V provides easy forms design with data editing and validation built into the forms". On this positive note, the VPLUS/3000 User's Manual begins the introduction to VPLUS/V, better known as View. On an equally positive note, many eager screen designers and programmers begin to design some very exotic

data input screens using all the powers that have been placed at their disposal by H.P.

Since Formspec, the screen design program, allows many data edits to be included in the screen specifications, data items can be validated and/or modified without the use of program code. However, there will be times when items of data being entered must be validated or otherwise manipulated by comparison with additional information in a data base or data file stored somewhere else on the system. To further complicate the situation, the outcome of this particular edit or manipulation can become the determining factor in another edit or data manipulation in another field located somewhere else on the input screen. Now, remember Murphy's law. Rest assured that this second data field will be located not conveniently adjacent to the first field, but at the farthest possible point away on the screen. Many other fields of data will exist between the two and there will be no changing the screen because of some time-honored tradition, red tape, or the boss says so.

A tremendous reduction in the frustration factor for both programmers and final users can be achieved if data items can be edited or validated or otherwise modified at short, logical intervals during the process of completing a form. As these smaller sections or subsets of data are processed, they can be "locked in" on the form and restricted from access or further modification. This creates a commitment by the user and can often help accomplish the accurate editing or validation of data entered later on the form. Likewise, access can be restricted to only those fields which are necessary for a particular section.

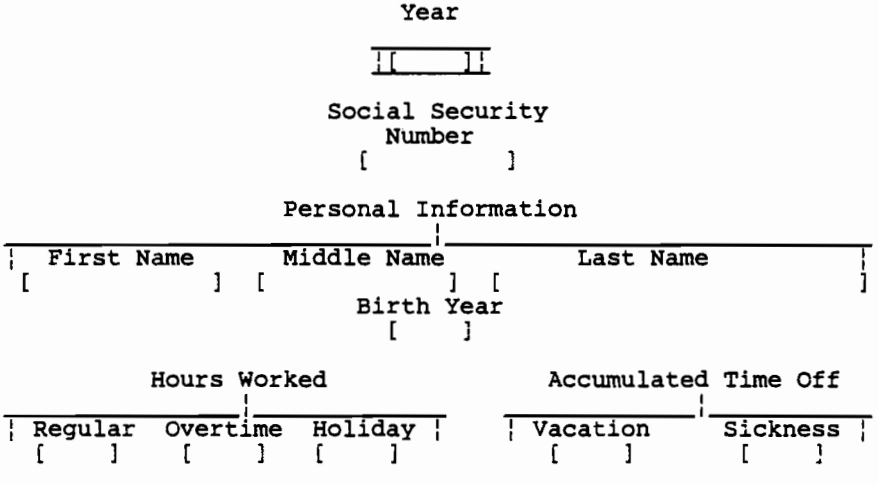
The form family concept of VPLUS/3000 is a very useful programming tool for achieving a solution to many of the problems that arise from situations such as those just described. According to the VPLUS/3000 User's Manual, "A family of forms is a collection of forms that share a common form layout but may have different field attributes, processing specifications, or form sequencing options." The "form family" consists of a "parent" or "father" form after which all other family members are modeled, and "child forms", also called "children". Each child has every field of the parent form and field lengths can not vary from form to form. However, field attributes and processing specifications can be significantly different and it is this capability which makes family forms so useful.

As an example, consider a data form having the following fields: Year, social security number, first name, middle

name, last name, birth year, regular hours worked, overtime hours worked, holiday hours worked, vacation hours accumulated, and sickness hours accumulated. With this form we wish to (1) add new employee names and hours, (2) update existing employees, (3) add additional work information about employees, (4) delete employee records, (5) browse employee records.

Since all records will more than likely be processed for the same year, the YEAR field will be defined as required on the father form and as display on all son forms. The father form for this family is shown in Figure 1. All unprotected fields are shown with the half-bright, inverse video enhancement which is depicted enclosed in a rectangle. The Formspec design characterisitcs are shown in Figure 2.

Figure 1
(FATHER)
EMPLOYEE INFORMATION ENTRY FORM



****

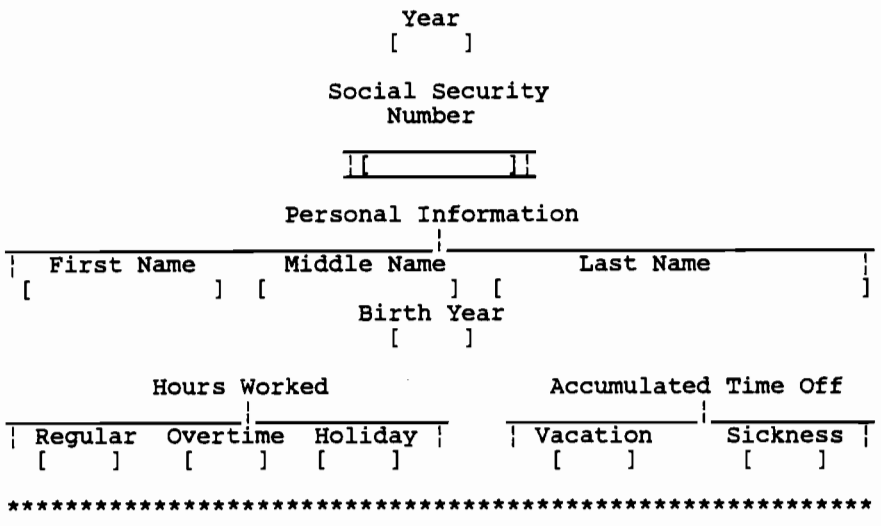
The first son form will be designed having the SOCIAL SECURITY NUMBER as its only user accessible field. The second son form will have the FIRST NAME, MIDDLE NAME, LAST NAME, and BIRTH YEAR as its only accessible fields, with all other fields defined as display field type. The third son form will have the HOURS information defined as either optional or required with all other field types display.

Figure 2
(FATHER)

```
Form: FATHER
   Repeat Option: N
   Next Form Option: C
   Next Form: SON1
   Reproduced from:

Comments:  Only the YEAR field is unprotected.
********* ********* ********* ********* ********* ********* ********* **********
Field: YEAR
   Num: 1    Len: 4     Name: YEAR             Enh: HI    FType: R  DType: DIG
   Init Value:

Field: SSN
   Num: 2    Len: 9     Name: SOCIAL_SEC_NO    Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: FNAME
   Num: 3    Len: 16    Name: FIRST_NAME       Enh: NONE  FType: D  DType: CHAR
   Init Value:

Field: MNAME
   Num: 4    Len: 16    Name: MIDDLE_NAME      Enh: NONE  FType: D  DType: CHAR
   Init Value:

   Num: 5    Len: 20    Name: LAST_NAME        Enh: NONE  FType: D  DType: CHAR
   Init Value:

Field: BYR
   Num: 6    Len: 4     Name: BIRTH_YEAR       Enh: NONE  FType: D  DType: CHAR
   Init Value:

Field: RHR
   Num: 7    Len: 4     Name: REGULAR_HOURS    Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: OHR
   Num: 8    Len: 4     Name: OVERTIME_HOURS   Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: HHR
   Num: 9    Len: 4     Name: HOLIDAY_HOURS    Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: VHR
   Num: 10   Len: 4     Name: VACATION_HOURS   Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: SHR
   Num: 11   Len: 4     Name: SICKNESS_HOURS   Enh: NONE  FType: D  DType: DIG
   Init Value:
```

After the year has been edited and approved, the first son form is displayed using the same program forms buffer. The cursor will be located in the first accessible field, which happens to be SOCIAL SECURITY NUMBER, so cursor location control occurs automatically (see Figures 3 and 4).

Figure 3
(SON1)
EMPLOYEE INFORMATION ENTRY FORM



After the SOCIAL SECURITY NUMBER has been entered by the user, the application program can validate it by searching the employee data set in the data base for the given number. If the record does not exist, then the second son form will be displayed, opening the FIRST NAME, MIDDLE NAME, LAST NAME, and BIRTH YEAR for access so they can be added(see figures 5 and 6). If the record exists in the data base, the name and birth year will be displayed on the third son form, bypassing the second son form and forcing the cursor to be located in the REGULAR HOURS WORKED field. Figures 7 and 8 show the third son form and its respective enhancements and field types.

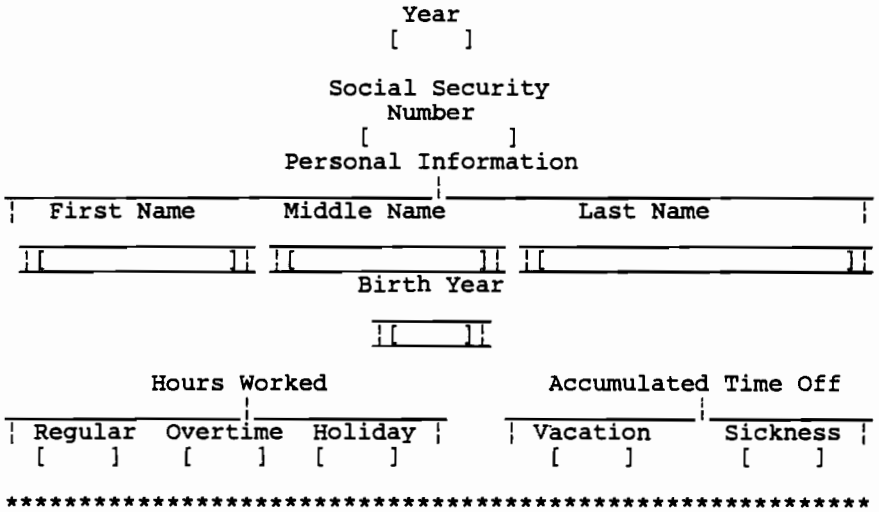
It now begins to appear that the computer is almost "smart", or I may be so bold as to say "user friendly". The cursor is positioned in the appropriate fields without requiring the user to "tab over" those fields which already contain information. This is because those "unnecessary" fields have been defined as display

Figure 4
(SON1)

```
Form: SON1
   Repeat Option: N
   Next Form Option: C
   Next Form: SON2
   Reproduced from: FATHER

Comments:  Only the Social Security Number is Unprotected
********* ********* ********* ********* ********* ********* ********* *********
Field: YEAR
   Num: 1    Len: 4     Name: YEAR             Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: SSN
   Num: 2    Len: 9     Name: SOCIAL_SEC_NO    Enh: HI    FType: R  DType: DIG
   Init Value:

Field: FNAME
   Num: 3    Len: 16    Name: FIRST_NAME       Enh: NONE  FType: D  DType: CHAR
   Init Value:

Field: MNAME
   Num: 4    Len: 16    Name: MIDDLE_NAME      Enh: NONE  FType: D  DType: CHAR
   Init Value:

Field: LNAME
   Num: 5    Len: 20    Name: LAST_NAME        Enh: NONE  FType: D  DType: CHAR
   Init Value:

Field: BYR
   Num: 6    Len: 4     Name: BIRTH_YEAR       Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: RHR
   Num: 7    Len: 4     Name: REGULAR_HOURS    Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: OHR
   Num: 8    Len: 4     Name: OVERTIME_HOURS   Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: HHR
   Num: 9    Len: 4     Name: HOLIDAY_HOURS    Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: VHR
   Num: 10   Len: 4     Name: VACATION_HOURS   Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: SHR
   Num: 11   Len: 4     Name: SICKNESS_HOURS   Enh: NONE  FType: D  DType: DIG
   Init Value:
```

field type and are "protected" from user access on this particular son. The screen changes have been very fast as

Figure 5
(SON2)
EMPLOYEE INFORMATION ENTRY FORM



the various son forms were presented because VIEW/3000 does not have to repaint each screen. Only new or changed information is repainted. Although access is not permitted to fields which do not pertain to the portion of the form with which our user is working, the application program can display any pertinent information in any of the fields we wish.

In addition, field enhancements can be changed from form to form. One good method is to make all DISPLAY type fields unenhanced and to use either an underline or inverse video enhancement for the unprotected fields. This makes the accessible area of a form very distinct from the rest of the form and as the user progresses through the various son forms, there is less opportunity for confusion.

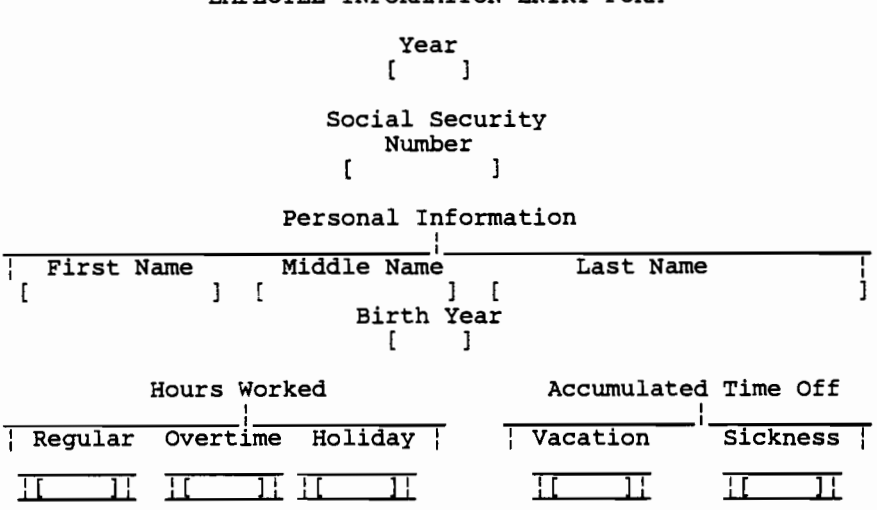
If any erroneous data is entered, (this actually does occasionally occur) only the "open", unprotected portion of the form is involved. All other fields retain their integrity and the errors can be more easily handled. On large forms with numerous data fields, this concept is especially useful because it helps break a large job into smaller units. It has been found that working with small

Figure 6
(SON2)

```
Form: SON2
   Repeat Option: N
   Next Form Option: C
   Next Form: SON3
   Reproduced from: FATHER

Comments:  Only Employee Personal Information is unprotected
********** ********** ********** ********** ********** ********** ********** **********
Field: YEAR
   Num: 1    Len: 4     Name: YEAR             Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: SSN
   Num: 2    Len: 9     Name: SOCIAL_SEC_NO    Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: FNAME
   Num: 3    Len: 16    Name: FIRST_NAME       Enh: HI    FType: R  DType: CHAR
   Init Value:

Field: MNAME
   Num: 4    Len: 16    Name: MIDDLE_NAME      Enh: HI    FType: O  DType: CHAR
   Init Value:

Field: LNAME
   Num: 5    Len: 20    Name: LAST_NAME        Enh: HI    FType: R  DType: CHAR
   Init Value:

Field: BYR
   Num: 6    Len: 4     Name: BIRTH_YEAR       Enh: HI    FType: R  DType: CHAR
   Init Value:

Field: RHR
   Num: 7    Len: 4     Name: REGULAR_HOURS    Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: OHR
   Num: 8    Len: 4     Name: OVERTIME_HOURS   Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: HHR
   Num: 9    Len: 4     Name: HOLIDAY_HOURS    Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: VHR
   Num: 10   Len: 4     Name: VACATION_HOURS   Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: SHR
   Num: 11   Len: 4     Name: SICKNESS_HOURS   Enh: NONE  FType: D  DType: DIG
   Init Value:
```

portions of the whole in this manner encourages more logical application program flow (structure). Therefore, the use of family forms can be programmer friendly too.

Figure 7
(SON3)
EMPLOYEE INFORMATION ENTRY FORM



After the first complete record has been processed, there may be a need for entering further information for the same employee for the same year. The form family concept very easily accomodates this need. Looping through the HOURS portion of the form is accomplished by looping through that portion of the application program which manipulates the third son form. Structured programming has provided for this ease of looping and the family of forms has assisted. Again the cursor is positioned in the correct field and no "tabbing" over unnecessary fields is required. When the user is finished entering data for any one particular employee, the application program can be signaled to return to the first son form to allow entry of another social security number. The year field will not be addressed unless the user indicates to do so. These logical steps back and forth through a form help make data entry less complicated and application program design becomes considerably more simple.

For applications which don't require additional edits beyond those performed by VIEW/3000 there is one more advantage worthy of mention. The program ENTRY.PUB.SYS can

Figure 8
(SON3)

```
Form: SON3
   Repeat Option: N
   Next Form Option: C
   Next Form: FATHER
   Reproduced from: FATHER

Comments:  Only the "HOURS" fields are unprotected
********* ********* ********* ********* ********* ********* ********* **********
Field: YEAR
   Num: 1    Len: 4     Name: YEAR             Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: SSN
   Num: 2    Len: 9     Name: SOCIAL_SEC_NO    Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: FNAME
   Num: 3    Len: 16    Name: FIRST_NAME       Enh: NONE  FType: D  DType: CHAR
   Init Value:

Field: MNAME
   Num: 4    Len: 16    Name: MIDDLE_NAME      Enh: NONE  FType: D  DType: CHAR
   Init Value:

Field: LNAME
   Num: 5    Len: 20    Name: LAST_NAME        Enh: NONE  FType: D  DType: CHAR
   Init Value:

Field: BYR
   Num: 6    Len: 4     Name: BIRTH_YEAR       Enh: NONE  FType: D  DType: DIG
   Init Value:

Field: RHR
   Num: 7    Len: 4     Name: REGULAR_HOURS    Enh: HI    FType: O  DType: DIG
   Init Value:

Field: OHR
   Num: 8    Len: 4     Name: OVERTIME_HOURS   Enh: HI    FType: O  DType: DIG
   Init Value:

Field: HHR
   Num: 9    Len: 4     Name: HOLIDAY_HOURS    Enh: HI    FType: O  DType: DIG
   Init Value:

Field: VHR
   Num: 10   Len: 4     Name: VACATION_HOURS   Enh: HI    FType: O  DType: DIG
   Init Value:

Field: SHR
   Num: 11   Len: 4     Name: SICKNESS_HOURS   Enh: HI    FType: O  DType: DIG
   Init Value:
```

be used and no applications program will need to be written. Logical traversal through the various son programs is accomplished through use of the NEXT FORM NAME portion of the FORM MENU in FORMSPEC. NEXT FORM NAME can be initially designated when the forms are designed and then programatically changed within the edits in the processing specifications portion of the FIELD MENU. The "CHANGE" processing statement can be used to alter the form name of the next form.

One of the more popular features of personal computers is "windowing". One aspect of windowing involves overlaying a current screen or a portion thereof with another screen. Although this second application of form families does not involve overlaying, it does accomplish the appearance and disappearance of entire portions of a screen from the monitor. In the previously discussed examples, any designer installed field labels will be present on all members of the family. In this example, even the field labels for fields not involved in a particular operation will be "disappeared" from the screen.

There are two kinds of fields on a form: protected and unprotected. Unprotected fields are available to the user for entering data. Protected fields cannot be accessed by the user. It is the protected fields which prove useful in this instance. Protected fields may be field titles, report headings, or "display only" fields to which the system sends data. Since neither users nor application programs can access the first two types of protected fields, the "display only" field is the only remaining tool. There is an old proverb which states, "What you don't know won't hurt you". This statement may hold true for screen users. Let us look further.

Having survived the form family concept as presented previously, we are now ready to build upon this foundation and become even more exotic. Users will be amazed (aren't they always?) and the boss will surely be properly impressed. Now, not only will the data fields appear and disappear at the touch of a key (f key or Enter) but also the corresponding labels. If the field is invisible and its label is invisible, then they must not exist, right? From the user's point of view, this is a true statement, but from a screen designer's view, having a thorough understanding of form families, who knows? This leads to another, more recent proverb: "Only her screen designer knows for sure".

This "form illusion" is accomplished by creating <u>all</u> fields on the father form as either "display only" protected fields or unprotected fields for data entry. This means

every character entered on the form layout must be part of a field delimited by brackets ([]), by escape followed by a bracket, or by a combination of these. The unprotected fields should have corresponding "label fields" which will be "display only" field types having an enhancement of "NONE". Although the field type for label fields probably won't be changed on any of the sons, there are a couple of good ways to cause labels to appear and disappear as the application program progresses through the form family.

The simpler, more straight forward way is to use the INITIAL VALUE area of the form design menu. Labels can be placed in only the appropriate son form and omitted in all others. These labels will appear on the form only when the son form is displayed and can be programmed to disappear from other forms or can be passed on to other forms. Application programs should use the "VINITFORM" intrinsic to activate the labels at the appropriate time.

For applications requiring even greater flexibility, the labeling capabilities are made almost limitless by the use of "VPUTBUFFER". A program can use its forms buffer area to send any desired values to the screen. These values can be hard coded into the program, read into memory from an outside file, or even input by the user if necessary. Blank spaces can be sent to the screen to cause labels to "disappear" and field enhancements can also be altered using "VCHANGEFIELD" to change from unprotected to protected field types and also from visible to invisible.

One last VPLUS/3000 intrinsic which has been valuable for both family and non-family forms is "VPLACECURSOR". This intrinsic allows the application program to designate which field the cursor is positioned in after "VSHOWFORM" has been called. This is useful for bypassing fields which have initial values which could be changed by the user but probably won't. These fields must be unprotected so they can be changed if necessary, but calling "VPLACECURSOR" avoids requiring the user to "tab over" them during the times when no changes are needed.

OSI: THE REALITIES OF MULTIVENDOR NETWORKING

ELAINE BONHAM
HEWLETT-PACKARD
CUPERTINO, CALIFORNIA

THE IMPORTANCE OF OSI

Rapid growth in the use of computers in business office, engineering and manufacturing environments has led to a wide choice of software solutions and hardware options from many different vendors.

Historically, this choice has been created by the fact that each of these environments has a completely different set of problems and needs that computers from many different vendors are addressing. Engineering design computers, for example, need sophisticated graphics interfaces, while sales and order processing requires multi-user transaction processing.

This diversity of needs and solutions leads to a problem, however, when one business area tries to interact with another. If the computers of one vendor can't "communicate" with those of a different vendor, business efficiency suffers. Without a common language, transfer of data can only be achieved by very costly protocol conversion.

In the early 1970s this problem was already widely acknowledged, and work began to define the "Esperanto" of computer communications -- a common language that would make it easy for the computers of different vendors to exchange data.

In 1977, after work on standardization in networking had been going on in different locales, the International Organization for Standardization coordinated the effort to devise a common computer language. This led to the development of ISO's Open Systems Interconnection (OSI) Reference Model, which outlined the structure of the language to be used for multivendor computer communication.

The OSI seven-layer model is thus a blueprint for multivendor communication. It was divided into seven layers to enable different standards groups to work on defining different

parts of the model, as well as to ensure that each layer could then be reassembled into the whole.

Within the model, protocols at each layer perform specific tasks, and protocols for all the layers together enable data transfer across a multivendor network. Work to define these protocols has been ongoing since the late '70s, and today the first full protocol stacks have been defined and implemented by major vendors.

Hewlett-Packard has been committed to networking standards, and has been an active participant in the work of standards groups, since 1972. The company's growth as a major computer manufacturer has led to our increased involvement in the standards arena.

Hewlett-Packard is committed to networking standards because HP believes that multivendor networking is and will continue to be of extreme importance to users of all kinds. Only when users are free to put together their optimal solutions, from the offerings of all vendors, will the potential of computers to make people more productive and efficient be achieved.

This paper gives a status report on the development of networking standards for the OSI Reference Model. It describes which services will be offered and looks realistically at when as-yet-unfinished standards are expected to be available.

STANDARDS: FROM PAPER TO PRODUCT

The OSI seven-layer model can be viewed as having two parts. Layers 1 through 4 are concerned with the transmission of data across networks, while layers 5 through 7 serve the users of the data.

To understand where each layer of the OSI model is in the standards process, it's important to consider how standards are defined and established.

Many different groups have input in defining a networking standard. In the United States, most standards definitions are generated by organizations such as the National Bureau of Standards (NBS) or the Institute of Electrical and Electronics Engineers (IEEE).

Each of the groups can work independently on defining a standard; ISO may then consider this standard for adoption, or will work with the group to develop the standard jointly. For instance, the X.25 spec came from CCITT in Europe, and the ISO 8802/3 from IEEE in the United States.

The initial form in which a standard is published is a draft proposal (DP). This is a working document that will be reviewed, reworked and revised until it becomes either a second draft proposal or a draft international standard (DIS).

This decision hinges on how complete the proposal is at the end of its first review cycle. If the draft proposal is comprehensive and the ensuing work has made it more complete, the document may be voted on by ISO to become a draft international standard. If further work is needed, the proposal will be given second draft status.

It is not until a standard has attained DIS status that it is stable enough for vendors to contemplate developing products based on that standard. After becoming a DIS, final reviews and minor changes still take place before the document becomes an international standard; these changes are easily incorporated into a product still under development.

Since each review cycle normally takes a year, the normal time it takes to go from a draft proposal to an international standard is roughly three years.

After a standard has been published, there may still be omissions or ambiguities in its defining documents. These differences are resolved by implementors' workshops, such as those run by the National Bureau of Standards, at which vendors agree to implement the standard in the same way. This is crucial to the ability of different vendors' products to work together, or interoperate.

Once a product has been built, there is a need for verification that the product in fact conforms to the standard. The Corporation for Open Systems (COS) is a consortium of vendors and users whose mission is to develop just such conformance test suites for OSI products. Until conformance testing, there is no guarantee that any product does conform to the established standard.

The final, and perhaps most crucial, stage of an OSI product's life cycle is validation that the product can interoperate with the products of other vendors. These interoperability tests provide a chance for vendors to work out any minor problems in their implementations.

With the OSI products currently on the market, conformance and interoperability testing have not been completed; therefore it's premature to regard these offerings as true OSI products.

Hewlett-Packard has recognized the importance of conformance testing for many years. The company was a founding member of COS, and has chaired the COS committee on test architecture. HP was also a co-founder of OSINET and EurOSInet, respectively the North American and European test and demonstration networks for multivendor interoperability.

HP: A LEADER IN NETWORKING STANDARDS

These long-standing commitments to standards-based networking reflect Hewlett-Packard's belief that OSI is the foundation for the multivendor networks of today and the future, and that standards being defined must be of value to users in a true multivendor environment.

Thus Hewlett-Packard has been an active leader in the development of international networking standards. We currently chair the IEEE committee on 802.3 Local Area Networks as well as the task force on Network Management. We are also contributing to the effort to develop many other standards, including those for Logical Link Control, Directory Services, and other application layer protocols.

HP is particularly active in the development of OSI standards for Network Management. There are currently four main parts of this effort: OSI Management Framework, Common Management Information Service Protocol (CMIS/P), Specific Management Information Services Protocol (SMIS/P), and the Management Information Base of the Management Operations Model.

Trudy Reusser of HP is the founder of ISO's X3T5.4 committee, which encompasses all of OSI management on the ANSI side. She is also a permanent member of the U.S. delegation to ISO for OSI Management, and she chairs the convergence committee between ANSI/OSI and IEEE 802.

Hewlett-Packard's commitment to OSI is an important part of its overall networking strategy, called HP AdvanceNet. HP AdvanceNet was implemented to replace the company's Distributed Systems Network (DSN) architecture.

HP AdvanceNet is designed to support the interconnection of both HP-to-HP and multivendor computer networks through support for the OSI model. HP has undertaken a major effort to develop networking software products that conform to the OSI model, while retaining the end-user network services originally offered under DSN and enhanced with the NS (Network Services) family of products.

In the early Eighties HP rewrote its proprietary networking software to conform with the structure of the OSI model, and implemented its first international standard products, based on CCITT X.25, in 1981. We followed this with the introduction of an ISO 8802/3-based LAN for all of our computer products. Subsequent products have included layers 4 and 5 protocol implementations: full MAP stacks and X.400.

Today, all major networks being installed are based on industry standards, such as X.25 and IEEE 802.3. HP's backbone for wide area networks is an X.25 network. Hewlett-Packard is

the only major computer vendor that will implement and support a total solution for wide area networking that encompasses both the computers and the network.

The HP AdvanceNet strategy is to migrate all proprietary services to OSI services as the OSI standards become available.

In today's multivendor environment, interoperability is crucial. Even when migration to OSI is achieved, it doesn't mean much if OSI products from different vendors can't work together, or interoperate. As an important part of its commitment to standards-based multivendor networking, HP has been a leader in major Manufacturing Automation Protocol (MAP) pilots and X.400 demonstrations. This has given Hewlett-Packard a lot of valuable experience in making OSI products interoperate in live environments, including the following:

--HP was one of seven companies to take part in the first public demonstration of OSI multivendor connectivity, transferring files over MAP version 1.0 on July 9, 1984 at the National Computer Conference in Las Vegas.

--HP completed the first operational MAP pilot at a General Motors plant.

--HP played a key role in the largest MAP pilot yet undertaken, for GM's Truck and Bus program.

--HP announced its first standard MAP products at the MAP Users' Group meeting in Michigan in September 1986. These are MAP 2.1 interface products for HP 1000 workcell control systems and HP 3000 business computer systems.

Hewlett-Packard is an OSI leader in the commercial environment as well. In the spring of 1987, HP demonstrated an X.400 product and tested it for interoperability at the Hannover Fair. The product was successfully tested with the products of thirteen other vendors, including DEC, Xerox, Olivetti, Data General, Honeywell-Bull, and Siemens.

In the multivendor environment, support is a vital consideration for users. For the last six years in a row, Hewlett-Packard has been ranked Number One in customer support in the annual Datapro survey. Now HP is enhancing its support offerings to assist users in both migrating to OSI and achieving multivendor interoperability.

Under HP Network Planning and Design, users call on HP's worldwide team of Network Consultants, who help prepare a customized proposal for planning and designing a complete network solution. The Consultants also draft an in-depth datacomm requirements analysis, a detailed network design, and plans for controlling user costs.

HP also offers Network Prepare, Network Startup and NetAssure -- contract services that help users map out and implement all the steps involved in building a network, from initial design alternatives to post-installation support.

Hewlett-Packard has made and followed through on its long-standing commitment to standards-based multivendor networking. As a leader in the development and implementation of the OSI model, HP looks forward to the day when OSI will enable users to put together their ideal networking solutions from the offerings of all vendors.

Issues in Artificial Intelligence:

The Social Impact of Artificial Intelligence Technology

Shawn Brayman

Brant Computer Services Limited
6303 Airport Road, Suite #201
Mississauga, Ontario
Canada
L4V 1S2

(416) 673-9417

The Social Impact of Artificial Intelligence Technology

by Shawn Brayman

Like any technology in its infancy, there is a high level of disagreement as to the eventual impact that AI technology will have. In an interesting article in Fortune magazine in June 1985, a dozen of the top AI researchers in the world were asked for their nut-shell observations about the technology. According to many, we are 50 years and several major breakthroughs away from anything that may be considered real computer intelligence.

According to Marvin Minsky, founder of MIT's Artificial Intelligence Laboratory, "computers need to develop such human traits as a sense of humour - which helps people dismiss absurd propositions quickly." He feels, as do many others, that to make truly intelligent machines will "require new insights into the nature of thought - and perhaps scientific breakthroughs even more brilliant than Einstein's".[1]

On the other side of the fence, many researchers believe that the basic principles of making intelligent machines are pretty well known, so that achieving a level of general intelligence up to or surpassing that of people is mainly a matter of building bigger and faster computers and stuffing them with huge quantities of knowledge.

The task of addressing the potential social impact of AI technology is far beyond anything that could be achieved in this paper. Rather, I will review a number of the areas in which the most active research is continuing and from the nature of the research and other sources discuss some general potential ramifications.

A place to start in attempting to determine the areas most likely to be substantially affected by AI technology is a quick review of the various areas of study in the field, and from each of these, assuming that they are each successful in bringing practical applications into the marketplace, looking briefly at what these applications and their impact may be.

A suitable manner in which to proceed may be to list each of the areas of concentration that were apparent at the two most recent AI conferences, the International Joint Conference on Artificial Intelligence held in Los Angeles in August of 1985 and the Fifth

National Conference on Artificial Intelligence held in Philadelphia in August 1986. At these conferences, leading researchers from around the world presented papers that we will assume represent the "state of the art". From this arbitrary grouping of papers, I will distill the subjects that are not technology focused but rather address a general area of application.

The subject areas from the conferences were:

- AI and Education
- AI Architectures
- Automated Reasoning
- Automatic Programming
- Cognitive Modelling
- Expert Systems
- Knowledge Representation
- Learning and Acquisition
- Logic Programming
- Natural Language
- Perception
- Philosophical Foundations
- Planning and Search
- Robotics
- Theorem Proving

The only interesting distinction between the two conferences was the addition of the area of "Applications" to the agenda of the 1986 conference, a fact that I feel is reflective of the pace of the industry at this time and the mushrooming size of application development.

The topic areas of AI Architectures, Automated Reasoning, Knowledge Representation, Learning and Acquisition, Logic Programming, the Philosophical Foundations, Theorem Proving, and Planning and Search, all refer to specific aspects of the implementation of the technology and mechanisms effecting the design and building of AI programs. Cognitive Modelling was discussed in the introduction of this paper.

The area of robotics is a subset that is often included under AI because of its heavy reliance on vision in particular, and depending on the role of the robot, other AI techniques to deal with ambiguity in the real world. At this time Japan is a world leader in developing and marketing robots for the factory, and we can expect that the impact of robots in the workplace will continue to increase over the balance of the decade. Although an important aspect of AI technology, I will not be addressing

it in detail in this paper since it is in many ways a hardware technology that is already significantly advanced.

The five remaining areas of study all appear to have significant commercial and technical impact so will form the main topics of discussion, these being: education, automated programming, expert systems, natural language, and perception. The major application area of process control will also be examined.

AI and Education

Education is beyond a doubt one of the most important areas of ongoing human endeavour, whether it is in the school room, the shop floor or the home. The concept of Computer Assisted Learning (CAL), (sometimes referred to as Computer Assisted Instruction - CAI), has been around for a number of years and is only just beginning to have any noticeable impact on the educational process.

As personal computers become more widely distributed, we see greater use of CAL packages in the schools, and we see organizations like TV Ontario offering adult education courses over the TV with floppy disks sent to the participants to be used at home in conjunction with the learning program. The concept of using computers to assist in the teaching process is not new, but it is still not widely implemented.

The effect that AI will have on this field is probably more qualitative than quantitative. In a traditional CAL program to tutor a student in mathematics, the program usually presents a series of questions to the student, who then calculates an answer and types it into the computer. The student scores points and proceeds on to more difficult examples if successful, or else repeats the lesson and tries to do better the next time.

With an intelligent tutor system (ITS) such as the "The Geometry Tutor" presented at IJCAI-85 by John Anderson, the computer is diagnostic as opposed to "drill and practice", so if the student provides the wrong answer to a problem, rather than a "sorry you are wrong - try again", the intelligent tutor will have rules by which it can look for the type of mistake the student made and correct the misunderstanding of the student, "correcting the student when behaviour deviates below a minimum threshold, and helping the student over hurdles".[2]

Because of this qualitative difference, we may expect that AI research and commercial application will result in more effective and personalized educational software and should enhance the educational process. An important question burried in this observation is "What is the role of education?", for without the answer to this question we may just be trying to replicate what is already being done rather than enhance the process.

> ...owing to a lack of human resources where one teacher is expected to look after the educational needs of some 30-odd pupils, most of the goals of education are neglected. As a result, the education system is not free to deal with the business of education but acts as a certification and placement agency.
>
> Therefore, it is in the areas of neglected goals of education where computers could play a major role, in contrast to the current situation where they are used to replace current activities of the teachers or the textbooks.[3]

At a recent meeting of the Ontario Association of Software Developers, a motion was entertained and accepted to prepare a proposal for the Chinese government, solicited by the Minister of Education of China, for an electronic school of the future to be based primarily around automated training aids. For a nation like China where the need to deliver quality education and training to a growing population is a massive undertaking, AI may be an important part of the solution.

For countries like Canada where a substantial education and training infrastructure is already in place, the impact of this technology may be seen as threatening or disruptive by some, although from a more optimistic perspective, since many educators are frustrated with the high teacher/student ratio, the acceptance of this technology may be very quick.

According to a market study by Frost & Sullivan on Artificial Intelligence Products, "the surge in use of AI with personal computers, which appears to have begun ... should continue through the end of the decade, though the initial applications will be limited to education and training".[4] It appears as if the field of education may be one of the first areas substantially impacted by this technology.

A secondary impact on training may become evident through software that is not written specifically for education, but is rather a production

expert system, a program with rules and facts that is emulating the expertise of a human expert. It is a basic function of almost all good expert systems that they explain the nature of the reasoning that they followed in arriving at a solution to a problem. As such they can be used to simulate real-world decision making and train employees in the limited domain with which they deal.

In this way, corporations that develop expert systems to capture the expertise of the top technical or professional people have already developed the basis for training systems for new or junior personnel. In a world where the problems of career changes and job retraining are growing daily, the positive aspects of AI technology may play an important role.

It is important to understand though, that the development of commercial expert systems will not result in a cornucopia of training software. Where the expert system has the "rules" of the expert decompiled to their most basic elements, the novice or trainee still requires an understanding of the basics or fundamentals to even understand what the underlying rules mean. The expert system will only demonstrate how and where they are applied.

Automated Programming

Everyone at one point or another has heard some version of the observation that if cars had evolved at the same rate as computers today's automobiles would get hundreds of thousands of miles per gallon and be sold for some amount under ten dollars.

Although this may be true for the computer hardware, most people in the computer industry will testify that computer software must be more like a car than a computer. Once created, the software may run millions of times faster and be far more complex than it was in the past, but to create the software still requires man-hours or man-years of effort by human programmers.

Some of the tools for computer programming have altered as we moved from switch registers to assembly level programming and in the fifties on to what are now termed "Third Generation" languages like Fortran, COBOL, Basic and others. By the early eighties a number of program generators which enhanced programmer productivity by a factor of 10 or more were

recognized and referred to as, not suprisingly, "Fourth Generation" languages.

Still, with all of the innovations and the move to higher level languages that could achieve far more functionality with a small number of programming commands, the production of software still is reliant on the hours and efforts of large numbers of programmers.

In the October 1984 issue of Fortune magazine, in an article called "The Next Revolution in Computer Programming", Tom Alexander discussed certain companies that used LISP machines and AI tools "to write software for everything from personal computers to mainframes (and) claim to have improved programming productivity by factors from two to 100".

> At the top of the Lisp line are the so-called "development" machines, used for designing and debugging new software. These are equipped with elaborate programming aids that help automate the programming process, keep track of innumerable details, and discover "bugs" - logical contradictions, for example. All the vendors supply extensive libraries of Lisp program modules that can be linked together quickly, like Tinkertoy components, to create complex programs, These modules include, for example, AI based software that can translate ordinary English words and phrases into computer language. Programmers are beginning to use these aids and AI's exploratory programing techniques to test new ideas and develop rough prototype programs in only a few days or weeks. Once the basic ideas have been proved out in this fashion, they can be modified and refined.[5]

Once again, it would appear as if this technology is not so much a threat to the jobs of programmers, (as with educators), but rather will enhance the ability of these people to deliver their respective goods or services. We can expect the price to decline and the availability of software to increase as the tools of the programmer start to have their effect. In the same manner that hardware prices have taken a dramatic downturn and PCs have permeated the market, software will become a less costly and more available product.

The net effect of this will be to expect software to keep better pace with micro-electronics allowing the momentum already gathered for the computer industry to continue.

Expert Systems

Probably the single most well known aspect of AI technology in respect to commercial acceptance and viability is that of expert systems. Breakthroughs in expert systems have led the way for the commercial acceptance of AI, and it will in all likelihood be expert systems that impact society most dramatically over the next few years.

The major hurdle in developing expert systems is the process of "knowledge engineering" wherein a computer analyst must try to decompile the rules that the human expert uses in making decisions, and then encode these in the computer. In early expert systems as much as 50 man-years of effort was required to follow this process through to the point where the computer system was as expert as the human in its specific field.

The true impact of this technology in early years (right now for instance), will not be in large expert systems that are being developed with years of man-effort. Rather, small "knowledge based systems" will be developed in short periods of time in a broad range of applications. A knowledge based system is another way of saying "an expert system that isn't an expert". Examples are loan assesment programs, advisory programs and decision support systems of all shapes and sizes.

The reason for this approach is the fact that for any computer program that is highly dependent on logical constructs, (IF THENS, GOTOs, DO WHILE etc), where the major effort is the analysis of information rather than the collection and reporting of data, Prolog and Lisp tools offer substantive advantages. Further, these smaller systems can be developed without the large investment in research and development dollars, can prove themselves early on, and allow firms to learn about the technology at less risk.

It is important to understand that in the same way that any Expert System could be written in FORTRAN, C, PASCAL or another language of this sort, many traditional decision support tools may be better built using PROLOG or LISP.

In respect to expert systems then, I expect we will begin to see the same types of arguments as "can a machine think ?" begin to show themselves in questions like "is this computer program an expert ?". In the same way that this line of questioning could prove pointless when applied to human "experts" it may prove to be a red herring in respect

to many computer programs.

What will be important is that this new computer programming technology will allow computer tools of increasing complexity to be developed at substaintially reduced costs. This in turn will continue to feed the general growth and use of computers in the workplace and the home.

Natural Language Understanding

The field of natural language study has several ways in which it manifests itself, including speech recognition, natural language understanding, speech generation, and translation. Most basic of all of these is natural language understanding which is the ability to enter a sentence in a human language, have the computer decipher what the sentence means and carry out the instructions.

> Intellect, the first such commercial system, is already used by several hundred of the nation's leading corporations. The impact that these systems are having on how information is disseminated within these organizations is quite profound. With Intellect, business professionals are able to use English queries to get at the vital corporate information base themselves.[6]

This will lead eventually to a truly user-friendly interface between an individual and a computer, the effect of which could be substantial. The impact of computers on the lives of disabled individuals is already manifesting itself. To couple natural language and possible speech recognition into a computer that may be connected to telephone systems, home shopping systems and the like could open up whole new worlds.

The use of voice command systems in cars, homes and all manner of equipment is on the horizon. The Department of Defense in the United States has already incorporated into advanced jet fighters some voice response systems that are attuned to specific pilots. We are not that far away from the day you jump in your car and ask "Where is the nearest gas station?" and you get an answer like "Left at the second light and its on the right !"

> Simple implementations of speech understanding systems will begin appearing within the next two years... Almost all speech recognition systems available today are 'speaker-dependent,

> discrete-speech' systems. This means that each speaker must train the system to recognize his or her voice speaking individual words. Once the system is trained, the user must speak in a halting manner to ensure that discernable pauses exist between each word.[7]

Because the pattern recognition process of speech recognition has become quite formalized, this aspect of the technology has removed itself somewhat from the rest of the AI field which is usually far more dependent on non-algorithmic solutions. Other developments in speech recognition, however, are closely tied to AI.

The Defense Advanced Research Agency (DARPA) in the United States has established an "overall goal ... to provide the U.S. with (a) broad line of machine intelligence technology and to demonstrate applications of the technology to critical problems in defense".[8] DARPA's budget of $600 million towards its strategic computing program is but one of the funding initiatives of the U.S. government. In the area of natural language understanding in particular:

> Unlike commercially available speech-recognition systems that typically must be trained to recognize the speech of users or to which the user must speak only one word at a time. DARPA's objective is to develop a system that can accept continuous speech and recognize a very large vocabulary of words without training.
>
> Carnegie-Mellon University is developing a system that is expected to be able to recognize a vocabulary of 1,000 spoken words. Ultimately the goal is 10,000 words. Texas Instruments is under contract to focus on continuous-speech understanding in a noisy environment.[9]

In the commercial arena a number of companies are taking aggressive strides to gain a leadership position in this new marketplace.

> One prototype system, designed to understand 10,000 words and to have the capability to operate as a "voice-activated word processor," consists of a variety of "expert" modules that use different methods to identify spoken words. One module, for example, can disambiguate homonyms based on their context. The system also uses natural language parsing to show the parts of speech for for each word and their relationship to one another. These techniques, combined with advanced acoustic resolution, have reportedly resulted in a recognition accuracy rate of over 97 percent in the experimental system.[10]

Perception or Vision Recognition

The problems of developing vision recognition systems currently lie in two areas: the actual cameras necessary to "view" something and the software necessary to identify and react to whatever it is that appears on the television screen.

A common application for vision recognition products is in robotics, where the "bin picking" problem is often sighted. In this example, a bin may be filled with large numbers of assorted items and the robot is instructed to select a specific piece and remove it from the bin.

It is necessary for systems to identify the various pieces in the bin in a 3-D context, and the position of its own "gripper" in relation to the object. This requires the ability to store the three dimensional profile of every object in the bin and being able to match the aspects of the part that the system has been asked to retrieve with any perspective of the same part in the bin. The robot must then close in on the desired object, grasp it and remove it from the bin.

The computations for this process are significant, coupled with limitations in camera technology that make 3-D differentiation difficult. Although some systems are available they are restricted in use, with no real breakthroughs expected until the next decade. At this time the Defense Advanced Research Projects (DARPA) is supporting research to develop a self-propelled vehicle able to drive down unknown roads at between 5 and 10 miles per hour. This is obviously a long way away from major commercial application but the ramifications are considerable.

Process Control

Probably one of the single biggest uses of the technology will be in the area of fault diagnostics and process control. In the past, procedural programs were hard pressed if not totally unable to address these problems since in a real world situation one never new which input would change, and it would be impossible to make a complete pass of the program for each unit of time. Likewise in fault diagnosis, the human user may not always approach the problem from a "straight line" perspective. With descriptive programming this problem is overcome since

the system only brings into play those rules or facts that match the real-world situation. An example of this is the effort being applied by NASA in respect to the American space station program.

> The NASA budget bill produced by the Senate Appropriations Committee ... required that NASA set aside 10% of the total space station costs for development of advanced automation and robotics not in use in existing spacecraft.
>
> The report stresses there must be expert systems on board the space station when it first becomes operational. This requirement can serve two purposes: it can be a catalyst for research and it can also ensure that the electronic architecture of the space station will have the capability of accepting artificial intelligence programs in the future.
>
> The committee found, however, that much greater use of automation, largely resulting from use of expert systems (because they mimic human expertise) could be realized if 13% of total space station costs were spent on development. Therefore, the committee wants the funding level for expert systems, robotics and other forms of automation raised from $800 million to $1 billion.[11]

The space station's initial expert systems are expected to be:

> Electrical Power Expert Systems
> Guidance, Navigation and Control Expert Systems
> Communications and Tracking Systems
> Information and Data Management Systems
> Environmental Control and Life Support.

In short, nearly all aspects of the day to day operation of the station will be handled by expert systems with appropriate backup, right down to the robotic inspection and exchange of replaceable parts. By 2010 they expect that the crew members will talk to the various expert systems in natural language.

Although the concept of HAL9000 from Arthur C. Clarke's 2001 is more encompassing, the basic process of managing system loads, monitoring sensors and providing fault diagnosis are standard to almost every manufacturing and processing industry in the country. At the recent International Joint Conference on Artificial Intelligence in Los Angeles, there were dozens of papers presented relating to working expert systems used in process control in factories, power stations, at

Kennedy Space Center and dozens of other practical applications. It is a technology that has arrived.

As with most statements of this nature the first and foremost concern tends to be the impact on jobs and society's ability to reorganize its methods of distributing of the wealth when more and more of the 'work tasks' are being taken over by machines. The belief that this will allow all individuals to move on to more challenging occupations is probably utopian in nature. Aside from concerns of this nature, other potential problems will inevitably arise - some already apparent and others not yet even anticipated.

Conclusion

The fact is that AI is here and is here to stay. This statement is far from controversial, considering that research in this field actually began back in the 1950's.

Like any other new and potentially life-changing technology (remember the microwave oven?), AI must overcome the obstacles to general acceptance by proving its significant value to society.

The growing interest and investment in expert systems on the part of the commercial marketplace is one positive direction for AI, and the massive AI commitment - in people and in dollars - from the US government is also most encouraging.

It is to our own advantage to educate ourselves as to the nature of Artificial Intelligence and to be aware of the latest developments and changing goals in this currently volatile field. Your interest in papers such as this can provide you with important background to the "AI issuse", and your continued efforts to read about, discuss, and (hopefully) participate in AI development will ensure that you will not be left behind when this not-so-new technology becomes mainstream.

Footnotes

1 "The Brains Behind Artificial Intelligence", Fortune (June 10, 1985), p. 97.

2 John R. Anderson, "The Geometry Tutor", Proceedings of the Ninth International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers Inc., Los Altos, CA (August 1985).

3 L. Ford, "Intelligent computer aided instruction", Artificial Intelligence: human effects, Ellis Horwood Ltd., West Sussex, England (1984).

4 "AI showing commercial potential", Computing Canada (January 9, 1985), p. 17.

5 Tom Alexander, "The Next Revolution in Computer Programming", Fortune (October 29, 1984), p. 81.

6 Larry R. Harris & Dwight B. Davis, Artificial Intelligence Enters the Marketplace, Bantam Books Toronto (June 1986), p. 167.

7 Ibid., p. 170.

8 Aviation Week & Space Technology, (April 22, 1985), p. 46.

9 Ibid., p. 54.

10 Larry R. Harris & Dwight B. Davis, Artificial Intelligence Enters the Marketplace, Bantam Books Toronto (June 1986), p. 170.

11 Aviation Week & Space Technology (April 22, 1985), p. 63.

A DEVELOPMENT METHODOLOGY FOR EXPERT SYSTEM PROGRAMMING

Shawn Brayman and Ross Hopmans

Brant Computer Services Limited
6303 Airport Road, Suite 201
Mississauga, Ontario
Canada

(416) 673-9417

Contents

1.0 Introduction

It is necessary in any software development project to have a pre-defined methodology established that allows the development team and management to measure the progress and success of the project.

Over the past 18 months Brant Computer Services has undertaken three expert system development projects using the MPROLOG language, a modular implementation of PROLOG that is widely installed on a number of different computer architectures. Our development work has been done on the HP Vectra (an IBM-PC/AT clone) and the source code is compatible with the implementation of MPROLOG for the HP3000 which Brant currently has in beta test.

In late 1985, we began working on our first expert system project. This was a personal financial planning system for a life insurance company in Canada and involved a standard 4GL component with an MPROLOG based advisor. Brant worked with one of the top financial planners in Canada on the knowledge engineering phase, and since that time has entered into an agreement to continue the development of the system and market it as a product on the HP3000 and on IBM-PCs.

The second system was an internal project from the start and involved the development of a diagnostic expert system for the "system management" of an HP3000 facility. Our first iteration involved the development of a simple expert system shell which we subsequently found incapable of handling the representation of the problem. We are currently developing a more specific diagnostic shell which we will then populate with the "knowledge" of the HP3000.

The third project was the development of a prototype system for a California-based agricultural management company. The system is designed to schedule irrigation events for farmers growing specific crops. The system takes into account climatic information, knowledge about the type of crop, specific information on the fields (soil type, salinity etc.) and management constraints such as limits to the minimum and maximum amount of water available, days of the week unacceptable for irrigation, and others to schedule a season's irrigation requirements.

Through our experience in the development of these three systems,

combined with our studies of literature on AI development methodologies, we are developing a set of guidelines to be followed by our staff in any AI development endeavour.

In the body of this paper we will first introduce some of the concepts of AI and Logic Programming, then review some observations about the knowledge engineering process, and then suggest what to look for in selecting a project that you intend to address with a tool like MPROLOG. Finally, we will outline in more general terms some of the major aspects in the deveolpment of an expert or knowledge-based system.

2.0 Overview of Artificial Intelligence

Artificial Intelligence is not a new field - it has been around since the 1950's when people like Dr. Marvin Minsky helped found the first Artificial Intelligence Laboratory at MIT. Since that time thousands of researchers in dozens of universities have added to the research effort. Although the field is not new, what is new is the perception of commercial readiness of certain aspects of the technology - namely expert systems.

For the first decade, AI research headed down what proved to be a blind alley. This effort attempted to create a hardware/software "thinking machine". It was felt that once we discovered how a person thinks, we could put this "thinking algorithm" into a computer. We could then provide it with information and it could "think" out the answer. The search was on for a general problem solver or thinking machine. The search was unsuccessful.

By the mid 1960's, recognizing that the attempt to create a general problem solver was not going to be successful in the short term, researchers tried a new tack - to try and create a program that could emulate a human expert in a limited domain of knowledge. The first major expert systems were born in the late 1960's.

DENDRAL was an expert system designed to help determine the structure of chemical compounds based upon analysis of the components of the molecule. MACSYMA was a second expert system started shortly afterwards that was designed to solve symbolic mathematical problems. In both cases the difficulty of the problems solved is signifigant and both of these expert systems are in routine use today.

When people discuss AI, confusion tends to arise as a result of the various aspects of the field - in the same way that the question "What is computer programming?" could be met with answers ranging from machine-level programs to applications programs, with a world of possibilities in between. In the field of AI, answers are further complicated by the fact that the linguists, the psychologists, the philosophers and the computer scientists working in AI all have different perspectives. Some of the AI application areas under study are:

- expert systems
- natural language understanding
- automatic programming
- learning systems
- perception and vision recognition
- robotics

When we discuss whether or not any of the existing AI systems are truly "intelligent", the AI experts tend to disagree. Some feel that we can already emulate intuition and human intelligence in some areas. Others, like Minsky, feel we are probably fifty years from machine intelligence and must first teach machines things like common sense and a sense of humour.

I will not take sides in the argument, nor do I feel it is that important. What is important is that whether intelligent or not, we are developing a new style of software that in some ways emulates human "thinking". There appears to be little doubt that the impact of this new type of software will be substantial.

Now let's take a look at some specifics about AI programming.

3.0 AI Programming

Although it is obvious that there must be differences in AI programming, there are clearly as many myths about AI. In this section of the paper, we will first discuss several basic issues concerning AI. From there we will go on to discuss some specifics of AI programming and how it is advantageous.

Research has told us that only about 20% of the problems that we encounter are numeric. The remaining 80% are symbolic problems - qualitative rather than quantitative. Traditional programming

techniques help us solve the numeric or quantitative problems. AI does not replace traditional techniques but does add power to them. AI adds a second dimension to problem solving by giving us techniques to solve symbolic or qualitative problems.

First, let me emphasize that AI programming is, most importantly, programming. You have a language, usually either LISP or PROLOG, and you develop programs. The programs we will be discussing are intended to solve certain types of problems - problems that are usually solved by someone in your organization with specific types of expertise; an expert in his field.

Second, although AI is often called "fifth generation computing" and PROLOG and LISP specifically called fifth generation languages, this is a bit of a misnomer. LISP was invented in the 1950's, around the same time as FORTRAN. It processes symbols rather than data, but other than this qualitative difference, Fortran and LISP are both of the same basic "generation" of programming tools. PROLOG was developed in the 1970's and may be more equivalent to a 4GL like PowerHouse or SPEEDWARE. Although fourth generation languages were or are supposed to replace third generation languages, symbolic processing is not designed to replace 4GL's. Our Expert Financial Planning System combines the fourth generation language SPEEDWARE with MPROLOG to apply the strong suits of a 4GL to those aspects of the problem which are algorithmic in nature, and using Brant's MPROLOG product for the expertise-based aspects of the system.

Third, let me point out that any program that can be written in MPROLOG or a 5GL <u>can</u> be written in a traditional language like COBOL. In fact, MPROLOG itself is written in the "C" language on the HP3000. With this in mind it stands to reason that anything written in MPROLOG on the HP3000 could in fact be written in "C".

Most AI programs are a series of rules and facts expressing the relationship between any number of things in a problem domain. The rules express these relationships in simple forms:

Rule 1: The computer won't work if
there is no power.

Rule 2: The computer won't work if
the power supply is cut off.

Rule 3: There is no power if
the on/off switch is turned off.

Rule 4: There is no power if
the cord is not plugged in.

Rule 5: There is no power if
the building's power is out.

In creating a diagnostic program to determine why our computer won't work, we put in a series of rules that outline relationships. MPROLOG will then sort and order these relationships, much as you might structure them in an ordinary program.

Let's take a look at some specific examples.

3.1 Dealing with Complex Relationships

In an effort to outline a few of the areas where AI techniques and tools are advantageous, we will look at a few examples. First let us consider a situation where a relationship exists between several conditions. The example we will use is that of a program to determine if an individual is eligible to receive a personal loan. Let us assume that the factors that are considered in a loan application are the amount of the loan, monthly payments, income of the individual, security of the loan and stability of the individual.

Expressed in MPROLOG we may have:

```
eligible (name, amount, payment, income, security,
          stability) if
capable_payments (amount, payment, income) and
loan_size_ok (amount, security) and
stability_ok (stability).
```

Each of the relationships to determine stability and the capability to make payments may in turn be a function of other factors or rules. We may have a query on our program of the sort:

```
?eligible(Shawn, 5000, 300, 2000, 0, none).
```

This query is asking if Shawn is eligible for a loan of $5000.00

with payments of $300.00 per month if he makes $2000.00 per month, has no security for the loan and is an unstable sort.

At this stage, we have seen very little that couldn't be easily accomplished with any other languages. Now let's try a few other inquiries.

```
?eligible(WHO, 20000, __, __, __, __).
```

This inquiry would give us a response of anyone who is eligible for a $20000.00 loan. Another example query might be:

```
?eligible(Shawn, HOWMUCH, 300, 2000, 0, none).
```

This inquiry would tell us how much of a loan Shawn would be eligible for, given that he only wanted to pay $300.00 per month and the other personal information is the same.

The important thing to recognize is that once the relationship has been defined, each of the different types of inquiries all use the same code - instead of having to write a routine to calculate the size of loans people are eligible for and a routine to list groups of people eligible for certain types of loans, or just to confirm if someone is eligible or not.

In effect, every argument in a relationship may or may not be defined. This means that achieving the same degree of flexibility in a traditional program would require 2^N routines. Obviously not every real world situation requires this degree of flexibility, but given a complex situation, the advantages become clear. In our example above, 64 separate routines or procedures would be required to accomplish the same effect as our one relationship in MPROLOG. In other words, if you programmed this in a traditional language, you must think out all 64 routines.

3.2 Dealing with Complex Inter-Relationships

A second aspect of the strength of AI languages is the way in which we can handle a series of complex inter-relationships. Let's take a look at another example.

Consider the following relationships:

C is true if A is true and B is true or
C is true if D is true and E is true
A is true if E is true and G is true
B is true if C is true and D is true
J is true if B is true and K is true
H is true if J is true and E is true
I is true if H is true and B is true

Given that D, E and K are true, is I true?

In a traditional problem-solving style, we would work out the relationships and develop some form of "tree-like" structure.

We would then encode this "solution structure" into our program in a series of structured IF/THEN's, LOOPS or whatever appeared necessary to best address our problem.

Now consider the problems we may encounter if we increase the number of relationships (we only have seven listed), if a relationship needs to be removed (say the relationship of H to J and E) or we need to be able to solve to find any of the ten conditions at any time. As we can perceive, with any of these scenarios, the complexity of our program would increase dramatically as would problems of maintenance.

To write the same program in MPROLOG, we would have:

```
true (c) if true (a) and true (b)
true (c) if true (d) and true (e)
true (a) if true (e) and true (g)
true (b) if true (c) and true (d)
true (j) if true (b) and true (k)
true (h) if true (j) and true (e)
true (i) if true (h) and true (b)

true (d)
true (e)
true (k)
```

Our inquiry would take the form:

```
?true (i)
YES
?true (g)
NO
```

To add new relationships or remove them, we simply add or delete that line of code. The order of the relationships does not matter, nor which condition we are searching for. In effect, we define the problem as opposed to the solution. We let MPROLOG automatically create our "tree-structure" and work out the inter-relationships. We do not need to worry as we add and delete new relationships.

3.3 Knowledge Trees and Heuristic Search

We seem to have described a system that will automatically generate a tree structure according to the relationships and allow us to traverse it, but what does this mean? There are other forms of decision support tools that use binary trees and there is no "magic" to them.

The qualitative difference with our Prolog system is not in the underlying tree but in how we traverse it. Picture a tree that becomes so large, with so many possibilities and permutations that passing through takes substantial time and resources. It is at this point that we introduce the concept of search techniques with backward chaining, forward chaining, breadth first search, depth first search, applying weights or probabilities to events and any number of other methods of searching a set of facts.

We also apply "rules" or heuristics - much as a human would - to cut down the search space, rather than try every possibility. For example, if we are trying to leave a room, we do not try the floor, the ceiling and the walls; we locate a door or window, locate a latch or knob, etc. Humans have rules that permit shortcuts rather than undergoing an exhaustive search for possibilities.

3.4 How do I know what is happening?

One of the obvious things about an AI program is that it is designed to automate an area of expertise that is not "obvious" in

content, or we wouldn't call the person who does that job an expert. With this in mind and given the examples we outlined above where a small piece of code can be used 64 different ways, how does a programmer know what his or her program will do? The answer is, he doesn't.

AI programs are designed to address real-world problems with non-obvious answers, so we must expect that once an AI program reaches a certain level of complexity, your programmers will not know what the system will do. Experience with some of our own projects has demonstrated that with as few as half a dozen relationships and a few variations of each, you will be surprised at some of the ways that your system will react.

This raises two key issues concerning AI programming. First, if you don't understand a conclusion reached, you can ask your program "WHY?" Unlike traditional programs where you know it is doing a process or procedure because you told it exactly what to do, in AI systems we don't know. Your expert system can be coded in a manner so as to allow you to ask WHY at any stage, and the system will outline the rules that it used to reach its conclusion.

In our previous rule example using the computer for instance, you may tell the system that your HP3000 won't work. It may respond with: "Check to see if the on/off switch is turned off". If you ask it "WHY?" it would respond:

```
Because - Rule 1:  The computer won't work if
                   there is no power, and

          Rule 3:  There is no power if
                   the on/off switch is turned off.
```

A second benefit of this ability is that it exemplifies the value of AI as a training and educational tool. The fact is that when your company encodes the expertise of someone in an AI system, this expert system can be used both in production and in training less experienced people.

3.5 LISP versus MPROLOG

In Brant we still have "discussions" among our people as to the advantages and disadvantages of COBOL versus fourth-generation languages. A similar argument exists in the AI world between LISP

and PROLOG users.

People wishing to develop applications in AI have three choices: one of the two base languages, (LISP or PROLOG), or what are called expert system shells. Expert system shells are partial expert systems already in place, where you drop in the specific rules you want and have a working expert system in less time and at less cost for man hours than with the base language.

According to a newsletter entitled Artificial Intelligence Markets (AIM) published in the United States, "several high end language vendors reported the return of customers who went off to buy expert system development tools to AI language products because they found the tools inadequate for their needs. Since sixty to seventy percent of high end AI language customers are Fortune 1000 firms or the federal government (as opposed to universities and AI companies), this is not a trend to sneeze at."

Accordingly, we will leave this discussion focussed on the two AI languages and leave discussions of the tools to others. In a tutorial on PROLOG presented at the International Joint Conference on Artificial Intelligence in August 1985, Dr. William Kornfeld pointed out several "cultural" and practical differences between LISP and PROLOG, namely that LISP is all-inclusive, has complex semantics and hence a large manual that can be confusing. PROLOG strives for simplicity, doing a number of things very well, such as symbolic structure manipulation and processing facts, and consists of a few well-chosen constructs. In a practical sense, LISP has a few features PROLOG does not, like destructive assignment and structure manipulation, iteration with DO's or LOOP's and global variables. PROLOG has logical variables and efficient procedure calls that make the programs clearer and easier to understand, at the expense of some generality.

Again, according to the AIM newsletter, "We think LISP will evolve into a systems manager language - system management functions will be in LISP, and they will be layered between the operating system and the applications. PROLOG and other languages (including conventional languages) will serve as secondary, application languages which will be called from LISP."

By no means is this intended to resolve any debate, but rather it is intended to outline some of the issues and where things appear to be heading. Brant at this time is only involved in application

work with MPROLOG.

4.0 Knowledge Engineering

A critical aspect of developing an expert system is the process of knowledge engineering. A knowledge engineer plays the same role as a traditional systems analyst, with a few important differences.

A knowledge engineer is an individual who attempts to discover the rules or heuristics that a human expert uses to reach conclusions in problem solving. A traditional systems analyst may ask a user specific questions about an application and hope to get a reasonable answer. Knowledge engineers are warned on the other hand not to expect a reasonable answer and in fact not to take what the expert says at face value.

It appears that experts use what can be called "compiled expertise", where they may take dozens of specific rules or conditions and compile them all down to one specific rule. It is the knowledge engineer's task to "decompile" this expertise to a form that is usable in our automated systems. The problem is that when asked, your expert will not recognize that some of his "rules" are actually compilations and so will provide the knowledge engineers with inaccurate or misleading information. In other words the expert may make up false rules to try and explain why he or she made a decision, because they do not know themselves.

Research has uncovered a phenomenon known as the "Paradox of Expertise" - the more competent domain experts become, the less able they are to describe the knowledge they use to solve problems!

A spin-off recommendation that quickly becomes evident from this Paradox of Expertise is that you should never be your own expert in a knowledge engineering exercise. The process of "decompiling" the knowledge is not a process that can be effectively undertaken by one individual on his own.

There are two main approaches to knowledge engineering - the observational and the intuitive method:

> Investigators applying the observational method don't interrupt the expert with questions or comments during problem solving. Instead, they analyze a transcript of the session

after the fact, possibly with the expert's help. AI researchers have used this approach to study problem solving by nonexperts, calling it protocol analysis. Here the subject talks while solving a simple problem or puzzle, the verbalizations are transcribed, and the underlying problem-solving processes are inferred from the resulting trace.[1]

In the case of the intuitive method the knowledge engineer gains a much deeper understanding of the problem domain through readings in the field and interaction with experts. The knowledge engineer then develops a set of rules that he feels adequately represents the "knowledge" and verifies these against the opinions or judgments of the experts. A reversal of this process is when the expert tries to develop the rules himself and provides them to the knowledge engineer. The knowledge engineer then enters them into the system and validates their effectiveness through other experts and the success of the system.

In most real world cases the knowledge engineering process is a combination of both the observational and the intuitive approach.

5.0 Qualifying the Candiate

Not all problems lend themselves to expert system technology, and a number of pitfalls and observations have been accumulated in almost every book on the subject. In this section we will outline a number of the most often quoted reasons for selecting a problem, along with some observations on potential pitfalls.

Traditional Technology doesn't work

If you can solve the problem in a reasonably cost effective manner using traditional languages like "C" or "Fortran" - do it! Do not waste your time and money making an AI tool do what can already be done effectively with a traditional language. The converse of this is a problem that many people get into where they decide to develop in a traditional language because of portability or performance. Develop your system in a high-level expert system tool so that the solution can be demonstrated in a reasonable length of time. One complete, you can consider re-implementing in a traditional language if required.

Narrow domain of application

For a successful expert system, it is important that the domain of application be as narrow as possible. For this reason most systems work in domains where there is a finite number of rules and relationships to ensure that the solution does not rely on facts that lie outside of its knowledge base. A knowledge base on fault diagnosis for a disk drive on a computer system is limited but detailed enough to warrant the use of an expert system. On the other extreme, an automated psychotherapist would be a poor choice for an expert system as it would need to know something about the whole world to be effective. The more constrained the knowledge domain, the more successful the project.

There are recognized experts

The experts in a field must be able to perform reasonably well in solving the type of problems being considered. If they do not, replicating poor performance on a computer may not prove fruitful. The experts should use heuristics or "rules of thumb" to solve their problems and will probably have a vocabulary of abstract terms and concepts. The experts must be better than amateurs.

The task is primarily cognitive

Problems that are undertaken should be cognitive in nature as opposed to numeric, and lead to non-numeric solutions.

The task is combinatoric

The task should have lots of choices to make it interesting and to take advantage of the descriptive as opposed to procedural nature of the program. The process of arriving at a solution should involve a chain of reasoning where one rule or fact leads to the firing of a second rule or fact and so on. If the process is actually a shallow line of reasoning on a large number of related but unconnected factors then AI technology does not add a lot to the process.

The problem is at the right stage of knowledge formalization

If an algorithm exists for the solving of a problem, then it is not necessary to apply an AI methodology. If, on the other extreme, no rules at all apply, then there will be no basis on which the

problem can be solved. AI can be considered if algorithms are too slow (ie: some scheduling or planning systems) or if they do not provide sufficient information.

Specialists agree on the knowledge

If there is no agreement on the underlying knowledge base and rules that the system uses, then the results of the process will be of little value as no one will accept the recommendations except that group of specialists who input the rules in the first place.

Other criteria that should be considered in the selection of a problem are:

-The skill is routinely taught to neophytes
-Data and case studies are available
-Incremental progress is possible
-Not a time critical application
-Freedom to fail
-Resulting system would have a high payoff
-Payoff is easily measurable
-Potential users are enthusiastic.

It should be recognized that many problems may meet these criteria without being a full-fledged expert system development project. These smaller systems (better referred to as knowledge-based systems) reflect many of the strengths of the technology without having the same level of investment by the company. They can prove to be very useful learning exercises with a valuable system as the result, and can set the stage for larger projects down the line.

6.0 A General Design Methodology

Up until recently, the knowledge engineer was himself an expert and an expert in short supply. The actual term was coined at a point in time when the process of refining the knowledge of a domain expert was of such a magnitude that the term "engineer" seemed appropriate. Many of the observations that have led to the outline of design methodologies for expert systems are themselves the results of refining the years of experience, the heuristics, of knowledge engineers.

> ... acquiring knowledge from experts resists linear, one-pass techniques. Instead, knowledge acquisition and system-building interact inseparably. Choices regarding the desired initial capabilities determine what knowledge to acquire first and how to engineer it for use. Over time the knowledge base expands to support additional capabilities, and this expansion often strains the capacity of the initial knowledge formulation. Thus the knowledge engineer frequently reaches a point where future progress depends on improved conceptualization and related reformulation of that knowledge.[2]

One of the basic facts of the knowledge engineering process is that it is iterative in nature; in 4GL terms, we apply a prototyping methodology. Since Brant has been involved in 4GL consulting for several years with Powerhouse, Speedware and Rapid we felt we could "wing it". We did rely on experts from Logicware Inc., the developers of MPROLOG, for consulting assistance during our startup - something we strongly recommend to others. Although we can all "go it on our own", the insight of an experienced knowledge engineer can be worth its weight in gold.

6.1 Task Suitability

As outlined in Section 5, the selection of the appropriate task is essential to a successful project. Without limiting what we discussed earlier, the main points to consider are:

- Focus on a problem area that does not require "common sense" to solve;
- The task should be very clearly defined, and one that is solved reasonably effectively by experts;
- Get a solid commitment from your expert, whether inhouse or not.

6.2 Building the First Prototype

From our experience it is first necessary to familiarize yourself with the problem domain by reading books, literature or articles on the area, or else through extensive dialogue with your expert.

Once you are feeling comfortable with the domain, clearly identify and characterize the important aspects of the problem. When the

problem has been clearly circumscribed, use the observational method to record the expert solving one or two cases that he feels are typical of the problem area. We addressed the area of knowledge engineering in Section 4.

Now comes a crucial stage in the project: the selection of the appropriate tool. Many projects falter or fail because the tool selected is not appropriate for representing the domain concepts and control structure. In such an event, change tools! Do not try to make the tool do something it is not designed to do. A second aspect of this problem can be that since the field itself is young, many tools are new to the market or immature. Try to pick a tool that has a good track record so that you don't end up spending more time debugging someone else's product than working on your own project.

When the tool has been selected and the first example is well understood, start building the first prototype. Try to have this working within a couple of months.

Focus on a limited number of representative problems. Make sure your system accurately handles these before branching out. Avoid getting bogged down with problems that the whole AI world is still struggling with (time and space relationships, natural language, etc).

Try to keep the domain specific knowledge seperate from general problem solving rules. Keep your inference engine simple and don't worry about efficiency in the first effort.

Try to develop or purchase tools that will help to assist in the rule-writing process. Many shells have such modules included and language vendors may provide you with examples of similar implementations.

Document the system as you would any other system. When testing the system consider the impact of errors in input/output and how they will impact rules or control strategies.

6.3 Extending the First Prototype

Once you have your system working and effectively solving your cross-section of representative problems, you are going to want to

let a number of people start to work with it. A few design considerations can go a long way.

Be sure you have an explanation facility so that users can review the reasoning that leads to a conclusion and the facts that are in the knowledge base.

Some very simple "front-end" interface features will help your users make better use of the system. Don't expect to train everyone on the "vocabulary" of your system.

Provide a mechanism for the users to record their comments or complaints if the system dosn't work. Quite often observations about the accuracy of the rules of the effectiveness of the system will not be recorded unless you are standing over their shoulder or such a facility exists.

Keep a test library of cases so that you can test against them as you continue to refine the system.

It is important to keep the expert "onside" during the process. Keep him sheltered from the technical problems but involved in such aspects as designing the interface.

Be careful that in all you have learned you don't start acting like the expert yourself and become challenging. Quite often, having built in a line of reasoning the knowledge engineer may become "protective".

6.4 Building the Second Generation of the System

Throw away the prototype. The purpose of the prototype was to refine the rules into machine utilizable form and gain a better understanding of the domain. This has been accomplished so don't saddle yourself with the mistakes from the first design.

Begin to consider questions of generality of problems and the ability of the system to respond effectively. Performance should become more of an issue with larger production systems.

Decide who the real users of the final system will be and make sure that the system I/O will feel natural to them. What may have been sufficient for yourself and the expert may scare others away.

6.5 Evaluating the System

Early in the process you should confirm with the expert how he would evaluate the system to determine whether it was useful. Does it match with what you are try to have the system accomplish? Are you solving a problem that will be of value to your users, or will it become self-evident after the first few times they use the system?

As an aside, in one documented knowledge engineering exercise, it was discovered that the expertise of one company's expert could be refined down to about twenty rules. Although an exception, in that case the expert was devastated and quit.

The user interface is crucial to the acceptance of the system.

7.0 Summary

Expert systems provide us with a permanence of expertise, and allow us to combine expertise and make it portable and more readily available. That leads to consistent, reproducable results with an explanation facility. There is no question of the value of expert systems in decision support to be advisors or colleagues in any number of fields.

Rather than attempt to design a flowchart that shows arrows going from knowledge engineers to experts to prototypes and so on, we have tried to outline many of the general concerns that are part of the "methodology" of developing an expert system. If your firm already has standards for documentation and reporting, they will probably be applicable with little modification.

The most important fact to remember is that although expert system development is not "research lab material", it is inherently experimental at this point. Take the time to familiarize yourself with several other systems and projects before you get underway. This is one of those occasions where front-end study and reading will pay dividends overall.

May 1987

Footnotes

[1] Donald A. Waterman, A Guide to Expert Systems (Menlo Park, 1985), p. 157.

[2] Hayes-Roth, Waterman & Lenat, Building Expert Systems (Menlo Park, 1983), p. 127.

Bibliography

Hayes-Roth, Waterman & Lenat. Building Expert Systems. Menlo Park (California): Addison-Wesley Publishing Company, 1983.

Waterman, Donald A. Building Expert Systems. Menlo Park: Addison-Wesley Publishing Company, 1985.

QUICK PERFORMANCE ANALYSIS

Chris Brayman and Gil Harrison

Brant Computer Services Limited
9637A-45 Avenue
Edmonton, Alberta
Canada

(403) 438-9123

Contents

1.0 Introduction

In traditional or third generation language environments, languages have little knowledge about the Operating and File Management Systems. As such, the application programmer must instruct the language to communicate with MPE, and interface with IMAGE, KSAM and MPE File Management Systems. Programmers forced to work at this level often develop an in-depth understanding of the HP3000 environment. Performance optimization becomes a routine exercise.

In today's fourth generation environments, Powerhouse provides an intelligent interface to the File Management Systems and handles the link to the MPE Operating System. The language provides numerous other functions automatically including screen handling routines, file input/output, locking strategies, stack control, and much more. The nature of Powerhouse's powerful default decisions and interface mechanisms has allowed application programmers to build systems without the in-depth understanding of the HP3000 environment and the result is that performance optimization is often ignored.

Ensuring good performance of your application systems developed with Powerhouse Quick - or any 4th GL language on the HP3000 - does require some knowledge of the 3000 environment in the context of the development language. Quick features such as QKGO parameters, procedural code, external links to 3rd GL languages and database structure should be analyzed for their utilization of system resources such as CPU, memory and Input/Output. This analysis becomes especially important with large systems that draw extensively on machine resources.

This paper presents performance topics that should be considered by application developers using Powerhouse Quick. The scope of this analysis focuses specifically on the effects of tuning Quick QKGO parameters (run-time variables) and the development guidelines we have adopted in our procedural code sections of Quick.

The content of this paper is presented as a discussion of the mechanisms involved, and a summary of our performance analysis. More detailed benchmark specifics will be presented at the Las Vegas IUG as part of an application case study of performance optimization.

2.0 Powerhouse Quick by Cognos

Powerhouse is a family of fourth generation language products authored by Cognos of Ottawa, Canada. The software is comprised of a dictionary, a report writer (Quiz), a menu and screen generator (Quick) and a transaction processor (QTP). In addition to these development products, Cognos offers the following applications: Powerhouse Graphics, an end-user report writer (The Expert), a financial accounting package (Multiview), a spread sheet program (Powerplan), and a development and documentation tool (The Architect).

In the development of large, on-line systems, a fourth generation environment must be capable of building sophisticated screens that perform extensive edits and calculations, and update multiple records in multiple files simultaneously. Quick provides the power and flexibility through the following:

1. Dictionary: An automatic interface to the Powerhouse Dictionary that provides a logical view of files and elements. More than a dozen element attributes (such as format picture clauses, date formats, value editing, pattern matching, screen labels, input and output scaling, and more) can be stored. All this information is automatically extracted from the dictionary (unless overridden by the screen program) when the screen is compiled.

2. Procedural and Non-Procedural Code: Although much of the programming is accomplished with non-procedural design statements, procedural sections allow control of more complex functions. In addition, programmers can explicitly issue file and data base locks, perform complicated edit routines and data manipulation, and much more.

3. Default Override: The programmer is allowed to alter default assumptions made by Quick in all pre-programmed routines such as entry, path, find, delete, and update procedures. The decisions made by Quick that some 4th GL's don't permit access to, are completely accessible for modification. The ability to override these decisions makes Quick extremely powerful and flexible as a 4th GL.

4. File Types: Eight different file types are permitted including Primary, Master, Reference, Delete, Audit, Secondary, Detail and Designer. All Quick screens include at least a primary file to which the user activities of entry, find, change, and delete are directed. Complex applications may involve multiple Master, Secondary, Reference, Audit, and Designer files logically related to the activities of the primary file.

5. QKGO: QKGO file establishes a wide range of run time parameters for the Quick screens. Default values are set for all parameters with the ability to increase or decrease values according to the application requirements. System programmers must analyze the machine environment for available memory, input /output limitations, and CPU power. Then depending on the size and complexity of your Powerhouse application(s) parameters can be tuned to ensure optimal use of stack space, internal buffers, extra data segments and more.

2.1 Screen Hierarchial Structure

The design of a screen hierarchy must balance the logical requirements of the application with the size of the stack required. Some logical structures may encourage a very deep hierarchy, but conflict with the stack limitations of the machine environment.

If the logical structure does not demand a deep hierarchy we tend to build shallow structures. The savings realized by this design approach can be used for increased work areas in stack and improve system performance.

Other considerations that come into play in the design of our system of screens include:

1. Adopt the KIS principal (keep it simple) in the design of individual screen programs. We try to spread the processing of multiple files over multiple screens where possible. Screen programs with a primary file and several secondary, designer and alias files may perform poorly depending on record sizes and the volume of processing logic.

2. Avoid accessing files sequentially or performing chained reads of many records in real time environments.

3. Use terminal memory for stacking screens (2.2.3).

4. Use default screen processing where possible.

2.2 QKGO Parameters and Performance

QKGO parameters are maintained by application programmers through the system of QKGO construction and maintenance screens provided by Powerhouse products. These parameters afford the program extensive control over the size and nature of the Quick environment according to the values specified.

To appreciate how some of the QKGO parameters affect a system of Quick screens, we should discuss briefly the process of running programs under MPE and activating Quick screens. The following should not be considered a comprehensive discussion on the mechanisms involved. Only those activities relevant to this paper will be discussed.

2.2.1 Processes

Under the control of the MPE Operating System programs are run as "processes". A process is the basic executable entity and consists of a Process Control Block, a set of code segments and a data segment (Stack) on which the code segments operate.

Code segments are shareable between processes and contain a set of machine instructions. Each process can have many code segments (eg: Quick).

A data segment (Stack) contains variable information

including data elements, control tables, file buffers, etc. The data segment is not shareable between processes and therefore cannot be accessed by others. If a process attempts to load more data on the stack than it can hold, it will abort with a stack overflow.

A process can define an extra data segment for temporary storage of data that would normally be in the stack. On multi-processing systems like the HP3000, inactive data segments can be moved to disc temporarily if more real memory is required than is available. These inactive data segments are moved back to real memory to become active data segments as required by the process. This allows more users to access the machine since each of the executing processes spend some portion of their time waiting for disc I/O and segments can be inactiviated during these idle times.

When memory becomes very full, the CPU can spend a great deal of time swapping data segments in and out of memory, resulting in overall performance degradation.

In the typical third generation environment such as COBOL, individual screens are executed as "program files". These files are shareable between processes as code segments. If multiple users access the same screen program, only one copy must reside in memory.

In the typical 4GL environment such as Quick, screen programs are not executed as "program files" and must be maintained separately in all user stacks. Therefore, screens accessed by multiple users must be maintained redundantly in memory for each user. In application systems with many users an increased memory requirement results.

This and other differences between 3rd and 4th GL environments emphasize the need to optimize utilization of machine resources. Increased demand by 4th GL systems for CPU, memory and I/O should be kept to a minimum.

2.2.2 Loading Quick Screens

In Powerhouse on-line environments, the Quick program controls the activities of the processes. The code

segments contained in this program are sharable between different users, and operate to control the environment of user sessions.

When a user selects a menu option to move lower in the screen hierarchy, the Quick program causes the following activities to occur as the next menu or screen is activated:

1. A copy of the compiled screen file is read from disc and is loaded at the top of available stack.

2. A copy of the stack data is copied to an extra data segment if space is available. If extra data segment space is not available the original compiled screen file on disc must be accessed if the screen becomes inactive.

3. The static screen display including the terminal programming and literal display fields are copied to a temporary disc file (called application lines).

4. The screen is displayed on the terminal using data contained in the application lines.

Some other considerations are:

A. New screens are loaded following the previous screens in the stack. Additional screens are also copied to extra data segments as space permits.

B. If screen stacking specifications are used by the Quick programmer, it is possible to maintain several screens in the application lines at once. As such, the terminal can switch from one screen to another without retransmitting data from the extra data segment or original compiled screen file. This feature is available only on terminals with more than one page of memory. This mechanism is controlled by the screen stacking commands of the SCREEN statement, the application lines QKGO parameter, and available terminal memory.

C. If screen stacking is not specified, the screen layout data for the active screen in the application lines is replaced by the next screen when the screen is inactivated.

D. When Quick screens become inactive (the user exits the screen and moves higher up in the hierarchy) the stack space is released. The screen copy in the extra data segment (as space permits) and the layout data in application lines are maintained.

An understanding of the environment just described (see 2.2.1 and 2.2.2) is important to the application programmer of complex Quick applications. System performance can be improved by considering those internal mechanisms that affect stack and extra data segment usage.

As a general rule the programmer should analyze each user group in an application for their resource requirements and establish separate QKGO files. Different users will access only a few screens and others may access many. Different screens may have very different requirements for work areas in stack.

2.2.3 QKGO Parameters

For the purpose of our performance analysis of QKGO parameters we will focus on the execution time parameters manipulated in our large production environments. The following is a general description of each parameter with our results of turning in different application or machine configurations:

1. Application Lines

This parameter involves the stacking options of the SCREEN statement, the application lines QKGO parameter and available terminal memory. The parameter controls the number of lines of simulated terminal memory used for stacking screens. A maximum of 240 screen lines can be specified by a value which is a multiple of 24.

The system of Quick screens are first mapped onto application lines according to screen stacking commands specified by the programmer. When displaying screens on the terminal, application lines are then mappped into terminal memory. Therefore, when an inactive screen becomes active, and the screen has been maintained in the temporary disc file of application lines, a screen flip occurs.

In small application systems we have realized the maximum benefit of this feature by stacking all the screens on different 24 line blocks of the application lines. The user enjoys an instantaneous flip between all the screens in his/her system after the first access and mapping to application lines and terminal memory.

In our large application systems we typically adopt a different approach to balance performance benefits against easy maintenance of the application. The master menu is mapped onto application lines 1 to 24. All menu options or screens at the second level are mapped to application lines 25 through 48. Again menus or screens at level three in the screen hierarchy are mapped to page three. This is repeated to the deepest level in our system. Maximum benefits can be seen in shallow hierarchies with many menu options where limited terminal memory is available.

2. Procedure Code Pages

This parameter sets the number of 256 byte pages reserved for procedure code in memory. The procedure code is required only when the screen is active or when files declared on the screen are updated from a lower level screen. As such, the procedure code is not maintained permanently in stack. Quick will read the original screen file or the allocated procedure code areas as necessary. Increasing this parameter increases stack usage but decreases the number of procedure code retrievals.

Depending on the volumes of procedure code in your system this parameter can be adjusted upward from the default of 8 pages to avoid swapping between the screen file and stack. As well, the order recommended for placing procedure code sections in your screen (page 9.11 of the Quick manual) should be adhered to strictly.

Example: When the field processing cycle is initiated (Accept Verb Processing) for a specific field the INPUT, EDIT, PROCESS, and OUTPUT procedures are all required in stack. If these procedures are located randomly in your screen, multiple reads to the screen file may result with

large code volumes.

3. Screen Table

This parameter determines the number of entries in a table used to track whether a Quick screen is available for reloading from an extra data segment or not.

As outlined in 2.2.2, Quick saves the screen in an extra data segment after using the screen the first time. Since it is much faster to reload a screen from an extra data segment than from the screen file on disc, this parameter should be set equal to the number of screens usually accessed by the user. The minimum value must be the number of screens in the longest arm of the hierarchy since Quick must be able to track all active screens.

If sufficient extra data segment space is available, it is highly recommended that this parameter be set high enough to hold all screens accessed by the user. Analysis shows a savings of 60 to 70 percent of CPU when accessing from extra data segments. Each user group should be analyzed separately and the value set according to need.

4. Terminal Buffer

This parameter determines the size of the buffer used for data read from data displayed on the terminal. In applications where users use rapid-fire entry, a low value will cause performance to suffer. If stack space is available the value should be raised to the maximum.

5. Expression Size

This parameter determines the site of the global work area reserved for evaluation of expressions and function results. Since the space allocated uses stack we set the value to the minumum and adjust upward until Quick does not issue an error message.

6. Files Opened

This parameter sets the number of table entries used to maintain information about open files. Since the space

allocated uses stack we set the value to the minimum and adjust upward until Quick does not issue an error message.

Since the number of files opened by different users in our systems will vary, we analyze each user group separately. As such, this parameter typically will be quite different for various user group QKGO files.

7. Screen Levels

This parameter should be set to the deepest number of levels in the screen hierarchy. Using a value too high wastes stack.

8. Selection Sizes

This parameter determines the work area size for selection buffers in each screen. The buffer must be larger than the total length of the values stored. If your users do not employ SELECT mode the value can be set slightly larger than the longest key field. If SELECT mode is used the value must be set according to the selection requirements of the screen.

9. Segment Size

This parameter determines the size of extra data segments used to store blocks of rollback and screen information. The Quick manual recommends a minimum value of 8192 words or the size allowed for extra data segments in the system configuration.

If large numbers of screens are accessed by a user group this parameter should be raised to the maximum. However, in machine environments with a shortage of memory, finding space for 16K word segments may require excessive memory management. As such, the value of this parameter may be reduced if a user group accesses only a few screens or your system has critical memory shortage.

10. Secondary Blocks

This parameter establishes the number of table entries used to maintain information about blocks of rollback and

screen information. The blocks are the size of the rollback buffer, and will reside in extra data segments as space permits, or in temporary disc files.

When the "Segment Size" has been increased from the default 8192, the multiple of this parameter and "Rollback Buffer" should be made equal to the "Segment Size".

Since it takes stack space to keep track of multiple blocks we typically increase the "Rollback Buffer" before "Secondary Blocks".

11. Rollback Buffer

This parameter determines the size of buffer space allocated for rolling back updates if an error occurs during the UPDATE procedure. The Quick manual recommends a minimum value of the largest record in your system rounded up to the nearest 128 words. If stack is available the value can be increased to avoid the use of the secondary areas extra data segments and temporary disc files for rolling back data.

The size of this parameter also works in conjunction with the "Secondary Blocks" parameter to determine the size of the secondary areas (extra data segments and temporary files) used for storing spillover from the "Rollback Buffer" and screen information.

The size of this parameter in words multiplied by the value of the "Secondary Blocks" should at least equal the size of the extra data segment established by the "Segment Size" parameter. If it is expected that multiple data segments will be required by a user, the value of this or the "Secondary Blocks" may be increased.

Analysis of each user group for storage requirements in these secondary areas is necessary to ensure optimal use of stack and extra data segment space. If users access many screens in the system and are constantly changing screens, it is beneficial to maintain all the screens in extra data segments. As such, a large "Paging System" (defined by "Rollback Buffer" multiplied by "Secondary Blocks") should

be maintained.

Increasing the size of this "Paging System" works well in machine environments with available main memory. Systems with limited memory will not acommodate large "Paging Systems" easily. As such, more accesses to the original screen file on disc will occur when limited extra data segment space is available in the "Paging System". If many users access a large number of screens frequently with limited memory and smaller "Paging Systems", then performance degradation is severe.

2.3 Quick Procedural Code or 3GL Subroutine?

Most standard functions in even our most complex applications can be handled effectively and efficiently with Quick. However, in certain circumstances very involved procedural sections of Quick that involve hundreds of lines of procedural code and multiple file processing can be more efficiently handled with a 3rd GL routine written in SPL or COBOL. All standard IO and concurrency controls are always left under the control of Quick.

As a general guideline we consider the use of 3rd GL subroutines in the following circumstances:

1. When on-line edits and/or procedural processing involves large volumes of procedural code infrequently executed. Procedural code must be loaded into the stack when the screen is activated. Unnecessarily loading procedure code that may not be executed simply wastes stack space.

2. When on-line edits and/or procedural processing involves (chained or serial) reads processing more than 500 records.

3. Where a standard routine is accessed by many users concurrently and requires immediate response.

4. When a Quick screen exceeds 1000 lines of code.

Subroutines are typically put in the segmented library and allocated to main memory.

3.0 Conclusions

1. On-line performance may be improved by using screen stacking in terminal memory. Your strategy for mapping will depend on where your user moves in the application. If the user accesses a small number of screens the greatest benefit can be made by stacking all the screens in available memory. If he or she accesses a wide variety of screens in a large system stacking can be done according to levels.

2. Accessing extra data segments for screen copies is much faster than the original compiled screen file on disc. If a user class accesses a large volume of screens, and the machine environment has sufficient memory, increase the size of the "Paging System" by adjusting upward the values of "Rollback Buffer", "Secondary Blocks", and "Segment Size" as required.

3. Work areas and table areas for QKGO parameters such as expression size, selection size and screen levels should be set as low as possible to avoid wasting stack.

4. If stack space is available, the parameter for "Procedure Code Pages" should be increased to the maximum to reduce procedure code paging. Procedure code sections should be organized properly in your screen.

5. External calls to 3rd GL routines should be considered in very large complex sections of Quick procedural code.

6. Adopt the KIS principal (keep it simple) in the design of individual screen programs. Spread processing of multiple files over multiple screens where possible.

Optimize your HP 3000 Applications by Utilizing RINs, MPE Message Files, and a Real-Time Terminal Monitor

Benedict G. Bruno
S.T.R. Software Company
P. O. Box 12506
Arlington, VA 22209

Introduction

Several articles have been written discussing the merits of the HP 3000 Computer System as a transaction processing oriented machine. Topics have included process handling, interactive on-line systems, etc. Excellent resources available to the standard MPE user/programmer for process handling control are the Resource Identification Number (better known as a RIN), MPE message files, and the real-time terminal monitor.

The concept of RINs has not been adequately described. This paper will discuss the benefits of local versus global RINs in a multi-processing environment, i.e., how may RINs be utilized for the control and monitoring of user applications executing on the HP 3000. RINs are extremely effective in the process handling environment. Several programming examples will be provided.

Although MPE provides several system intrinsics and commands for creation and access of local and global RINs, no software tool is provided for display of this information. Thus it is difficult, almost impossible, to display the RIN value, password, creating user/account, information of the locking process, and information of waiting process. This information is maintained by MPE in the System RIN table.

The contents of the System RIN table have been formatted for the user in the supplied SHOWRIN program. The author has developed this program for both MPE IV and MPE V based systems. Depending upon capability, portions of the RIN table are formatted in displays similar to the HP On-line Performance Tool (OPT) program. Using this SHOWRIN program and the discussion of RIN usage, you can start integrating local and global RINs into your HP 3000 applications right away!

In addition to the local and global RINs, MPE message files may be utilized to create unattended real-time applications on the HP 3000. Among these is the application monitor whereby an HP display terminal can control and maintain software systems by using a combination of RINs and message files. This paper will conclude with a complete example of how a real time terminal emulator may be developed.

Local RINs

MPE provides local and global RINs for process control. In order to begin our discussion, let us define local and global RINs.

Local RINs can be described having the following attributes:

* Created, accessed, and released on a session or job basis only. (intra-job)

* Programmatic MPE intrinsic access only.

* User maintains control of RINs by specifying which task each local RIN value is associated.

* The MPE LOCRINOWNER intrinsic returns the process identification number (PIN) of the process currently locking the local rin.

A local RIN is a resource allocated on a job or session basis for use by any process executed during this job or session. Local RINs are created and released only using the GETLOCRIN and FREELOCRIN intrinsics programmatically. The GETLOCRIN intrinsic allocates the requested number of local RINs for the current job or session. The FREELOCRIN intrinsic releases all local RINs previously obtained with the GETLOCRIN intrinsic.

Local RINs are accessed using the LOCKLOCRIN and UNLOCKLOCRIN intrinsics. Local RINs are referenced with the integer value assigned within range of the number allocated with the GETLOCRIN intrinsic. Thus, if the user requests 24 local RINs with the GETLOCRIN intrinsic, then the LOCKLOCRIN intrinsic may lock RIN numbers one through 24. The UNLOCKLOCRIN intrinsic performs the same way.

The MPE LOCRINOWNER intrinsic provides the PIN of the locking process for the supplied local RIN. This capability prevents deadlocks and informs any process in a process tree of the currently locking process. Thus, there is no need for a display of the local RIN entries from the system RIN table.

Local RIN Example

Probably the best usage of a local RIN that I have ever used is to implement a priority based queue. Suppose that we have a process X which must accept input from any of three external processes A, B, and C with an assigned priority of one (1), two (2), or three (3). We wish to control the inputs to process X such that all priority one messages are executed first, then all priority two messages, and finally all priority three messages. We must also assume that the

current request must be completed in order to start the next, i.e., we cannot abort the current request and place it back in its assigned queue in order to process this higher priority queue entry.

How might we do this? Well, if we obtain two local RINs with values one (1) and two (2) using the GETLOCRIN intrinsic within the input process, the three processes handling priorities one, two, and three must attempt to lock and unlock these local RINs with some defined scheme. Here is the scheme:

Pri 1: Must attempt to lock (LOCKLOCRIN) on the RIN assigned value one (1). He suspends until the lock is complete. Once locked this process continues to hold this lock until all his priority requests are written to the primary executing process X.

Since each of the priority two and three processes must lock the RIN value of one, as long as priority one process holds this RIN value, then no other priority request may be submitted.

Pri 2: Must attempt to lock (LOCKLOCRIN) on the RIN assigned value two (2). He suspends until the lock is complete. Once locked this process repeats this lock procedure on RIN assigned value one (1), i.e., he attempts to lock the RIN value one. If so, then the RIN is released and his priority request is written to the primary executing process X. This is repeated until there are no more input requests at his priority level, i.e., for each request, he must lock RIN value one, unlock RIN value one, and write the record. When no requests remain then lock of RIN value two is also released.

The reader should note that the priority two process will send as many of his requests until the first request is made at priority one. If no requests are found at priority one, then all requests of priority two are submitted. While at least one priority two request remains, then no priority three requests may be submitted.

Pri 3: Must attempt to lock on the RIN assigned value two. He suspends until the lock is complete. Once locked this process immediately releases this lock. Then he must attempt to lock on the RIN assigned value one. He suspends until the lock is complete. Once locked this process immediately releases this lock. At this point the request may be written to the executing priority process X.

The reader should note that in order for priority 3 process C to write his request, he must lock each of the local RIN values independently. Thus if any of the higher priority

processes have it locked, it allows that higher priority process to complete its requests before this priority three process can issue his.

Global RINs

Global RINs can be described having the following attributes:

* Created and released on a system wide (global) basis for any session or job on the system. (inter-job)

* User assigned RIN password is associated with a unique entry in the system RIN table using the GETRIN MPE command. eleased only by creating owner with the FREERIN MPE command.

* System RIN table is system disc resident on logical device number 1. Modified only during reload.

* Any job or session may access the global RIN using the LOCKGLORIN and UNLOCKGLORIN intrinsics by supplying the RIN password and RIN value.

* The HP supplied SYSDUMP program displays the global RIN number with the creating user and account. The user specified password is NOT displayed. Hence, the need for the SHOWRIN utility program.

Global RINs are available to any session or job on the system provided the value and password are known. Global RINs are assigned to a creating user and account with the GETRIN command. The RIN associated with this password is permanently maintained in the System RIN table which is system disc resident. An entry may be removed using the FREERIN command. Note that the RIN entry may only be removed if the creating user and account match the logged on user and account names!

The size of the global RIN table residing on the system disc is configured with the SYSDUMP dialogue. Changing this size can only occur during a RELOAD. The standard MPE configuration is delivered with a table size of 48 entries. In order to utilize the global RIN feature, you should increase the size of the table appropriately.

The following commands can be executed to create and free the global RIN assigned using the password of GRIN.

```
:HELLO user.account
:GETRIN GRIN
RIN: nnn

:FREERIN nnn
```

MPE responds with the integer value assigned to the RIN password with the GETRIN command. It is important that you always remember this association. Currently, no HP supplied utility displays the RIN entry value, password, and creating user and account names.

You should note that the FREELOCRIN intrinsic removes all created local RINs for the current session or job. The FREERIN MPE command releases ONE global RIN from the system RIN table if currently signed on as the creating user and account.

In order to continue, the reader should now note that the local RINs may be very effective during process control within an individual process tree, i.e., intra-job level processes. Global RINs are very effective during process control within several processes executing from different jobs, sessions, and accounts, i.e., inter-job level.

Global RINs are valuable in the monitor and control of several programs comprising an application. By assigning a global RIN to each program in the application and requiring each program in the application to lock its associated RIN, we are guaranteed that only one copy of the program is executing. Some programmers implement this feature by opening a file for exclusive access. Since the global RIN is locked during the entire execution of this program and additional data structures may need to be locked, i.e., IMAGE/3000 data bases, KSAM and MPE files, etc., the program file must be :PREPed with MR (multiple resource or better known as multiple RIN) capability in order to hold multiple locks. The user, group, and account capabilities must also be modified for MR capability.

Once each of the programs in the application is initiated and the lock is maintained on their RINs, another monitor program may control the execution of these processes even further. Suppose that this monitor program is responsible for the initiation, control, and termination of all programs within this application. Once each process is started, the RIN is held for the duration of this process. The monitor need only attempt a conditional lock in order to determine if the process is still executing. If the conditional lock succeeds, then the process is not executing since the monitor process was able to lock the RIN; otherwise, the lock fails and the process is executing since the RIN is unavailable.

This technique is further enhanced by utilizing the writers id feature with the open, close, and data record types of the MPE message file system. The requirement is to have each program in the application open the monitor message file for write access. If the monitor program opens this message file and enables the writers id feature, then as each process opens, writes, or closes this file, the monitor program receives a record containing the writers id, the file access, and data if it is a write. The writers id is an integer assigned by the file system in ascending order to each program opening the file.

If a close record is detected unexpectedly, then the monitor program need only identify which program has failed and respond in a controlled manner. The monitor may restart the failing program or may request all other programs in the application to terminate immediately.

In addition to the initiation and termination of programs within the application, the monitor program may also control the execution of each of these programs. Specifically, the monitor program may suspend or resume execution of these programs. Utilizing the message file techniques above, the monitor program issues a conditional lock on a secondary RIN for each process requested to suspend. Once locked, the monitor requests that the process suspend itself by issuing an unconditional lock on this secondary RIN. The process is resumed execution when the monitor releases the secondary RIN. Hence, an excellent method of global process control!

The SHOWRIN program was developed in order to determine which RINs were currently locked and by which process. Another useful feature is that the password could be associated with the RIN entry itself. This feature saved documenting the RIN entries and their passwords into a journal for later reference.

Another useful utility would be one that would display which of the System local and global RINs are locked, have been locked (and when), by whom, etc. I have also had requests to actually determine the RIN value for a given user/account and/or RIN password. Also, if given the RIN value, then who is the creating user/account and the RIN password. This can all be displayed with the SHOWRIN program and/or SHOWRIN procedure.

Global RIN Example

Now let's begin with an example to better explain these features. An application of one monitor and two programs will be used to demonstrate the global RIN concepts discussed earlier. Sample code will be provided in both SPL and COBOLII.

In Figure 1 below, the monitor program is diagrammed to use the parm value of the :RUN command as the value of the global RIN named 'MONITOR'. The monitor program issues a conditional lock on this global RIN. If successful, the program continues; otherwise, an error message is displayed since some other process currently has this RIN locked (this prevents two copies of the monitor program from executing because another copy is already executing). The monitor program opens three message files: the primary message file named 'M' is opened for read access enabling the writers id feature; the additional message files named 'A' and 'B' are opened for write access in order to communicate with process A and process B.

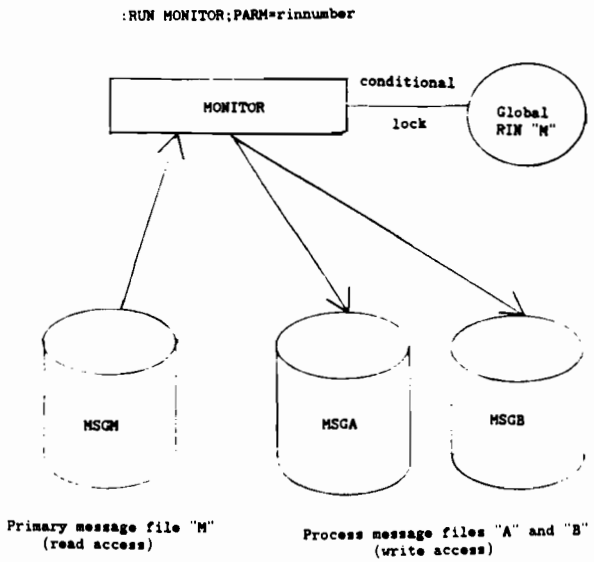


Figure 1: Monitor Program "M" Execution Flow Diagram

The example has been coded in SPL below in Figure 2. The program variable declarations are first displayed. Using the indentation and blank lines provide for easy reading.

Figure 2: Monitor program in SPL

```
1     $CONTROL USLINIT,MAP
2
3     <<*****************************************************************>>
4     <<** MONITOR Program Version Information.                         **>>
5     <<**                                                             **>>
6     <<** Version    Date    Who  Comments / Description              **>>
7     <<** ------- -------- --- ---------------------------------- **>>
8     <<** B.01.00   5/31/86  BGB  Initial program release.           **>>
9     <<**                                                             **>>
10    <<** This program will lock the global RIN name of "MONITOR"     **>>
11    <<** with the RIN number passed in the ;PARM= parameter.         **>>
12    <<*****************************************************************>>
13    $PAGE "*****  MONITOR PROGRAM  *****"
14    BEGIN
15
16    COMMENT  EQUATES AND DEFINES
17    ****************************************************************;
18
19    DEFINE    IR'CONTROL'CODE          = LINPUT'RECORD#,
20              IR'PROGRAM'ID            = INPUT'RECORD(2)#;
21
22    EQUATE    RS                       = %036,
23              BEL                      = %007,
24              ESC                      = %033;
25    $PAGE
26    COMMENT   LOCAL VARIABLES
27    ****************************************************************;
28
29    LOGICAL ARRAY           LTERMLINE(0:79);
30    BYTE ARRAY              TERMLINE(*)=LTERMLINE;
31
32    LOGICAL ARRAY           LBA1(0:79);
33    BYTE ARRAY              BA1(*)=LBA1;
34
35    LOGICAL ARRAY           LINFO'STRING(0:127);
36    BYTE ARRAY              INFO'STRING(*)=LINFO'STRING;
37
38    LOGICAL ARRAY           LINPUT'RECORD(0:119);
39    BYTE ARRAY              INPUT'RECORD(*)=LINPUT'RECORD;
40
41    DOUBLE                  RUN'PARM;
42
43    LOGICAL                 DONE,
44                            LOCK'COND,
45                            OK,
46                            TIME'2'QUIT,
47                            TRUE'COND;
48
49    INTEGER                 CERROR,
50                            CPARM,
51                            FERROR,
52                            I,
53                            IN'LENGTH,
54                            INFO'LENGTH,
55                            LENGTH,
56                            MSGA'FILE,
57                            MSGB'FILE,
58                            MSGM'FILE,
59                            NUM'CHAR,
60                            PAUSE'RIN'A,
```

Continuing the program below, the intrinsic declarations are listed in alphabetical order. The GET'INFO procedure will locate the ;PARM= and ;INFO= parameters of the :RUN statement by locating the terminate stack marker.

Figure 2: Monitor program in SPL (Cont.)

```
61                          PAUSE'RIN'B,
62                          MONITOR'RIN;
63
64      INTRINSIC           ASCII,
65                          BINARY,
66                          COMMAND,
67                          DATELINE,
68                          FCHECK,
69                          FCLOSE,
70                          FCONTROL,
71                          FERRMSG,
72                          FOPEN,
73                          FREAD,
74                          FWRITE,
75                          LOCKGLORIN,
76                          PRINT,
77                          PRINTFILEINFO,
78                          QUIT,
79                          TERMINATE,
80                          UNLOCKGLORIN;
81      $PAGE
82           PROCEDURE  GET'INFO (PARM, INFOL, INFOSTR);
83              DOUBLE  PARM;
84             INTEGER  INFOL;
85       LOGICAL ARRAY  INFOSTR;
86
87           BEGIN
88           <<*****************************************************>>
89           <<** The GET'INFO procedure will locate the ;PARM= and **>>
90           <<** ;INFO= parameters from the :RUN statement.  This  **>>
91           <<** procedure will correctly execute on any version of**>>
92           <<** MPE IV and MPE V.  Note the Delta-P value change  **>>
93           <<** from %0 of MPE IV to %40000 of MPE V.  By reading **>>
94           <<** stack markers till these values, we eventually    **>>
95           <<** locate the terminate marker and the parameters are**>>
96           <<** then at SM-4, SM-5, and SM-6.                     **>>
97           <<*****************************************************>>
98
99
100          INTEGER          QREG = Q;
101
102          INTEGER POINTER  SM;
103
104          BYTE POINTER     INFO,
105                           INFOT;
106
107          <<************************>>
108          <<** Start of main code **>>
109          <<************************>>
110
111          @SM:=@QREG;                  <<** Point to current marker **>>
112
113          I:=0;
114          DONE:=FALSE;
115
116          DO
117            BEGIN
118            IF SM(-2) = 0 THEN         <<** MPE IV terminate marker **>>
119              DONE:=TRUE
120            ELSE
```

The FS'ERROR procedure formats a standard error message for any file system error detected while accessing any of the MPE files. This includes the text from FERRMSG and the tombstone from PRINTFILEINFO.

Figure 2: Monitor program in SPL (Cont.)

```
121             BEGIN
122             IF SM(-2) = %40000 THEN <<** MPE V terminate marker **>>
123               DONE:=TRUE
124             ELSE
125               BEGIN
126               @SM:=@SM-SM;          <<** Walk down stack marker  **>>
127               I:=I+1;
128               IF I = 100 THEN
129                 QUIT (-1);          <<** Cannot find terminate   **>>
130               END;                  <<** marker so abort!        **>>
131             END;
132           END
133         UNTIL (DONE);
134
135         PARM:=DOUBLE (SM(-4));      <<** SM-4 = value of ;PARM=  **>>
136         INFOL:=SM (-6);             <<** SM-6 = length of ;INFO= **>>
137         @INFO:=SM(-5);              <<** Byte ptr to info string **>>
138         @INFOT:=@INFOSTR&ASL(1);    <<** Byte ptr of target      **>>
139         MOVE INFOT:=INFO,(INFOL);   <<** Move to target          **>>
140         END;
141     $PAGE
142         PROCEDURE  FS'ERROR (FILE'INDEX, FILE'NUMBER, FS'INTRINSIC);
143             VALUE  FILE'INDEX, FS'INTRINSIC;
144           INTEGER  FILE'INDEX, FILE'NUMBER, FS'INTRINSIC;
145
146         BEGIN
147         FCHECK (FILE'NUMBER, FERROR);
148         MOVE TERMLINE:="** Unexpected FS error ",2;
149         NUM'CHAR:=ASCII (FERROR,10,BA1);
150         MOVE *:=BA1,(NUM'CHAR),2;
151         MOVE *:=" has occurred on ",2;
152         CASE FILE'INDEX OF
153           BEGIN
154       <<0>> ;
155       <<1>> MOVE *:="MSGM",2;
156       <<2>> MOVE *:="MSGA",2;
157       <<3>> MOVE *:="MSGB",2;
158           END;  <<** END OF CASE **>>
159         MOVE *:=" during ",2;
160         CASE FS'INTRINSIC OF
161           BEGIN
162       <<0>> ;
163       <<1>> MOVE *:="FOPEN",2;
164       <<2>> MOVE *:="FCLOSE",2;
165       <<3>> MOVE *:="FREAD",2;
166       <<4>> MOVE *:="FWRITE",2;
167       <<5>> MOVE *:="FCONTROL",2;
168           END;  <<** END OF CASE **>>
169         MOVE *:=". **",2;
170         LENGTH:=TOS-LOGICAL(@TERMLINE);
171         PRINT (LTERMLINE,-LENGTH,0);
172         FERRMSG (FERROR,LTERMLINE,LENGTH);
173         PRINT (LTERMLINE,0,0);
174         PRINT (LTERMLINE,-LENGTH,0);
175         PRINTFILEINFO (FILE'NUMBER);
176         TERMINATE;
177         END;
178     $PAGE
179         BEGIN
180
```

Execution of the program begins on this page. The program banner and system time are displayed. The global RINs for the MONITOR program, program A, and program B are retrieved from the ;INFO= parameter with the GET'INFO procedure. The MONITOR global RIN is locked to restrict only one execution of this program.

Figure 2: Monitor program in SPL (Cont.)

```
181         <<**********************************>>
182         <<** Start of main program code. **>>
183         <<**********************************>>
184
185         TRUE'COND:=TRUE;
186
187         MOVE TERMLINE:=("*** MONITOR *** B.01.00 Copyr. 1986, ",
188                         "Benge Bruno.  All rights reserved."),2;
189         LENGTH:=TOS-LOGICAL(@TERMLINE);
190         PRINT (LTERMLINE,-LENGTH,0);
191         DATELINE (TERMLINE);
192         PRINT (LTERMLINE,-27,0);
193         PRINT (LTERMLINE,0,0);
194
195         GET'INFO (RUN'PARM, INFO'LENGTH, LINFO'STRING);
196
197         MONITOR'RIN:=BINARY (INFO'STRING,3);
198         IF <> THEN
199           BEGIN
200           MOVE TERMLINE:="** Invalid monitor RIN passed. **",2;
201           LENGTH:=TOS-LOGICAL(@TERMLINE);
202           PRINT (LTERMLINE,-LENGTH,0);
203           TERMINATE;
204           END;
205
206         PAUSE'RIN'A:=BINARY (INFO'STRING(3),3);
207         IF <> THEN
208           BEGIN
209           MOVE TERMLINE:="** Invalid pause RIN for A passed. **",2;
210           LENGTH:=TOS-LOGICAL(@TERMLINE);
211           PRINT (LTERMLINE,-LENGTH,0);
212           TERMINATE;
213           END;
214
215         PAUSE'RIN'B:=BINARY (INFO'STRING(6),3);
216         IF <> THEN
217           BEGIN
218           MOVE TERMLINE:="** Invalid pause RIN for B passed. **",2;
219           LENGTH:=TOS-LOGICAL(@TERMLINE);
220           PRINT (LTERMLINE,-LENGTH,0);
221           TERMINATE;
222           END;
223     $PAGE
224         <<************************************************>>
225         <<** Lock RIN 'MONITOR' to ensure that only one  **>>
226         <<** copy of this program is active.  Locate the **>>
227         <<** RIN number from the ;PARM= parameter.       **>>
228         <<************************************************>>
229
230         LOCK'COND.(15:1):=0;
231         MOVE BA1:="MONITOR ";
232
233         LOCKGLORIN (MONITOR'RIN,LOCK'COND,BA1);
234         IF <> THEN
235           BEGIN
236           IF > THEN
237             BEGIN
238             MOVE TERMLINE:=("** Monitor program is currently ",
239                             "executing. **"),2;
240             LENGTH:=TOS-LOGICAL(@TERMLINE);
```

Now that the global RIN is locked, continue by opening the message files and enable the message file wait facility. Note the file and access options of the FOPEN intrinsic. This allows for multi-processing and inter-job access.

Figure 2: Monitor program in SPL (Cont.)

```
241             PRINT (LTERMLINE,-LENGTH,0);
242             END
243           ELSE
244             BEGIN
245             MOVE TERMLINE:=("** LOCKGLORIN intrinsic error: RIN# = "),2;
246             NUM'CHAR:=ASCII (MONITOR'RIN,10,BA1);
247             MOVE *:=BA1,(NUM'CHAR),2;
248             MOVE *:=".  RIN password = MONITOR. **",2;
249             LENGTH:=TOS-LOGICAL(@TERMLINE);
250             PRINT (LTERMLINE,-LENGTH,0);
251             END;
252           MOVE TERMLINE:="** This process cannot continue. **",2;
253           LENGTH:=TOS-LOGICAL(@TERMLINE);
254           PRINT (LTERMLINE,0,0);
255           PRINT (LTERMLINE,-LENGTH,0);
256           TERMINATE;
257           END;
258     $PAGE
259         <<*************************************************>>
260         <<** Now open the MONITOR, PROGRAMA, and PROGRAMB     **>>
261         <<** message files for read and write access.        **>>
262         <<*************************************************>>
263
264         MOVE TERMLINE:="MSGM ";
265         MSGM'FILE:=FOPEN (TERMLINE, %30105, %2300);
266         IF <> THEN
267           FS'ERROR (1,MSGM'FILE,1);
268
269         MOVE TERMLINE:="MSGA ";
270         MSGA'FILE:=FOPEN (TERMLINE, %30105, %2303);
271         IF <> THEN
272           FS'ERROR (2,MSGA'FILE,1);
273
274         MOVE TERMLINE:="MSGB ";
275         MSGB'FILE:=FOPEN (TERMLINE, %30105, %2303);
276         IF <> THEN
277           FS'ERROR (3,MSGB'FILE,1);
278
279         <<*************************************************>>
280         <<** Now enable extended waits on empty files for   **>>
281         <<** read access and full files for write access.   **>>
282         <<*************************************************>>
283
284         FCONTROL (MSGM'FILE, 45, TRUE'COND);
285         IF <> THEN
286           FS'ERROR (1,MSGM'FILE,5);
287
288         FCONTROL (MSGA'FILE, 45, TRUE'COND);
289         IF <> THEN
290           FS'ERROR (2,MSGA'FILE,5);
291
292         FCONTROL (MSGB'FILE, 45, TRUE'COND);
293         IF <> THEN
294           FS'ERROR (3,MSGB'FILE,5);
295     $PAGE
296         <<*************************************************>>
297         <<** Now execute the main body of the loop waiting **>>
298         <<** for the control codes to process.             **>>
299         <<*************************************************>>
300
```

The primary MSGM message file is read for incoming process requests. The external job streams for programs A and B may be started, stopped, suspended, or resumed.

Figure 2: Monitor program in SPL (Cont.)

```
301         OK:=TRUE;
302         TIME'2'QUIT:=FALSE;
303
304         DO
305           BEGIN
306           IN'LENGTH:=FREAD (MSGM'FILE, LINPUT'RECORD, -240);
307           IF <> THEN
308             FS'ERROR (1,MSGM'FILE,3);
309
310           IF IR'CONTROL'CODE < -4 LOR IR'CONTROL'CODE > -1 THEN
311             BEGIN
312             MOVE TERMLINE:="** Invalid control code of ",2;
313             NUM'CHAR:=ASCII (IR'CONTROL'CODE,10,BA1);
314             MOVE *:=BA1,(NUM'CHAR),2;
315             MOVE *:=" received.  Ignored. **",2;
316             LENGTH:=TOS-LOGICAL(@TERMLINE);
317             PRINT (LTERMLINE,-LENGTH,0);
318             END
319           ELSE
320             BEGIN
321             I:=IR'CONTROL'CODE * -1;
322             CASE I OF
323               BEGIN
324         <<0>> ;
325
326         <<1>> BEGIN  <<** Startup. **>>
327               MOVE TERMLINE:=("STREAM PROGRAMA.STREAMS",%15);
328               COMMAND (TERMLINE,CERROR,CPARM);
329               IF <> OR CERROR > 0 OR CPARM > 0 THEN
330                 BEGIN
331                 MOVE TERMLINE:=("** Unable to stream PROGRAMA. ",
332                                 "Cerror="),2;
333                 NUM'CHAR:=ASCII (CERROR,10,BA1);
334                 MOVE *:=BA1,(NUM'CHAR),2;
335                 MOVE *:=". Cparm=",2;
336                 NUM'CHAR:=ASCII (CPARM,10,BA1);
337                 MOVE *:=BA1,(NUM'CHAR),2;
338                 MOVE *:=". **",2;
339                 LENGTH:=TOS-LOGICAL(@TERMLINE);
340                 PRINT (LTERMLINE,-LENGTH,0);
341                 OK:=FALSE;
342                 END
343               ELSE
344                 BEGIN
345                 MOVE TERMLINE:=("STREAM PROGRAMB.STREAMS",%15);
346                 COMMAND (TERMLINE,CERROR,CPARM);
347                 IF <> OR CERROR > 0 OR CPARM > 0 THEN
348                   BEGIN
349                   MOVE TERMLINE:=("** Unable to stream PROGRAMA. ",
350                                   "Cerror="),2;
351                   NUM'CHAR:=ASCII (CERROR,10,BA1);
352                   MOVE *:=BA1,(NUM'CHAR),2;
353                   MOVE *:=". Cparm=",2;
354                   NUM'CHAR:=ASCII (CPARM,10,BA1);
355                   MOVE *:=BA1,(NUM'CHAR),2;
356                   MOVE *:=". **",2;
357                   LENGTH:=TOS-LOGICAL(@TERMLINE);
358                   PRINT (LTERMLINE,-LENGTH,0);
359                   OK:=FALSE;
360                   END
```

In order to suspend each process, the secondary global RIN is locked by the monitor program. Once locked, program A or B is informed to unconditionally lock this RIN. This will be demonstrated later in program A.

Figure 2: Monitor program in SPL (Cont.)

```
361               ELSE
362                 BEGIN
363                 MOVE TERMLINE:=("Programs A and B have been ",
364                                 "started."),2;
365                 LENGTH:=TOS-LOGICAL(@TERMLINE);
366                 PRINT (LTERMLINE,-LENGTH,0);
367                 END;
368               END;
369             END;
370
371         <<2>> BEGIN  <<** Shutdown. **>>
372             LTERMLINE:=-1;
373             FWRITE (MSGA'FILE,LTERMLINE,-2,0);
374             IF <> THEN
375               FS'ERROR (2,MSGA'FILE,4);
376
377             FWRITE (MSGB'FILE,LTERMLINE,-2,0);
378             IF <> THEN
379               FS'ERROR (3,MSGB'FILE,4);
380
381             MOVE TERMLINE:="Programs A and B have been stopped. ",2;
382             LENGTH:=TOS-LOGICAL(@TERMLINE);
383             PRINT (LTERMLINE,-LENGTH,0);
384
385             TIME'2'QUIT:=TRUE;
386             END;
387
388         <<3>> BEGIN  <<** Suspend A or B. **>>
389             IF IR'PROGRAM'ID = "A" OR IR'PROGRAM'ID = "a" THEN
390               I:=1
391             ELSE
392               IF IR'PROGRAM'ID = "B" OR IR'PROGRAM'ID = "b" THEN
393                 I:=2
394               ELSE
395                 I:=0;
396
397             CASE I OF
398               BEGIN
399
400           <<0>> BEGIN
401             MOVE TERMLINE:=("** Unknown process to suspend; must ",
402                             "be 'A' or 'B'. **"),2;
403             LENGTH:=TOS-LOGICAL(@TERMLINE);
404             PRINT (LTERMLINE,-LENGTH,0);
405             END;
406
407           <<1>> BEGIN
408             LOCK'COND.(15:1):=1;
409             MOVE BA1:="PROGRAMA ";
410
411             LOCKGLORIN (PAUSE'RIN'A,LOCK'COND,BA1);
412             IF < THEN
413               BEGIN
414               MOVE TERMLINE:=("** LOCKGLORIN intrinsic error: ",
415                               "RIN# = "),2;
416               NUM'CHAR:=ASCII (PAUSE'RIN'A,10,BA1);
417               MOVE *:=BA1,(NUM'CHAR),2;
418               MOVE *:=".  RIN password = PROGRAMA. **",2;
419               LENGTH:=TOS-LOGICAL(@TERMLINE);
420               PRINT (LTERMLINE,-LENGTH,0);
```

Program B is suspended by executing the code below. This completes the suspend code.

Figure 2: Monitor program in SPL (Cont.)

```
421                 OK:=FALSE;
422                 END;
423
424               LTERMLINE:=-2;
425               FWRITE (MSGA'FILE,LTERMLINE,-2,0);
426               IF <> THEN
427                 FS'ERROR (2,MSGA'FILE,4);
428
429               MOVE TERMLINE:="Program A is suspended.",2;
430               LENGTH:=TOS-LOGICAL(@TERMLINE);
431               PRINT (LTERMLINE,-LENGTH,0);
432               END;
433
434          <<2>> BEGIN
435               LOCK'COND.(15:1):=1;
436               MOVE BA1:="PROGRAMB ";
437
438               LOCKGLORIN (PAUSE'RIN'B,LOCK'COND,BA1);
439               IF < THEN
440                 BEGIN
441                 MOVE TERMLINE:=("** LOCKGLORIN intrinsic error: ",
442                                 "RIN# = "),2;
443                 NUM'CHAR:=ASCII (PAUSE'RIN'B,10,BA1);
444                 MOVE *:=BA1,(NUM'CHAR),2;
445                 MOVE *:=".  RIN password = PROGRAMB. **",2;
446                 LENGTH:=TOS-LOGICAL(@TERMLINE);
447                 PRINT (LTERMLINE,-LENGTH,0);
448                 OK:=FALSE;
449                 END;
450
451               LTERMLINE:=-2;
452               FWRITE (MSGB'FILE,LTERMLINE,-2,0);
453               IF <> THEN
454                 FS'ERROR (2,MSGB'FILE,4);
455
456               MOVE TERMLINE:="Program B is suspended.",2;
457               LENGTH:=TOS-LOGICAL(@TERMLINE);
458               PRINT (LTERMLINE,-LENGTH,0);
459               END;
460
461               END;  <<** END OF CASE **>>
462             END;
463
464        <<4>> BEGIN  <<** Resume A or B. **>>
465             IF IR'PROGRAM'ID = "A" OR IR'PROGRAM'ID = "a" THEN
466               I:=1
467             ELSE
468               IF IR'PROGRAM'ID = "B" OR IR'PROGRAM'ID = "b" THEN
469                 I:=2
470               ELSE
471                 I:=0;
472
473             CASE I OF
474               BEGIN
475
476          <<0>> BEGIN
477               MOVE TERMLINE:=("** Unknown process to resume; must ",
478                               "be 'A' or 'B'. **"),2;
479               LENGTH:=TOS-LOGICAL(@TERMLINE);
480               PRINT (LTERMLINE,-LENGTH,0);
```

Programs A and B are resumed by simply unlocking the secondary global RIN. This will allow programs A and B to successfully lock the RIN. The RIN is then immediately unlocked and the process continues as before. Once the monitor program completes, the MONITOR global RIN is unlocked and the message files are closed.

Figure 2: Monitor program in SPL (Cont.)

```
481                 END;
482
483           <<1>> BEGIN
484                 UNLOCKGLORIN (PAUSE'RIN'A);
485
486                 MOVE TERMLINE:="Program A has been resumed.",2;
487                 LENGTH:=TOS-LOGICAL(@TERMLINE);
488                 PRINT (LTERMLINE,-LENGTH,0);
489                 END;
490
491           <<2>> BEGIN
492                 UNLOCKGLORIN (PAUSE'RIN'B);
493
494                 MOVE TERMLINE:="Program B has been resumed.",2;
495                 LENGTH:=TOS-LOGICAL(@TERMLINE);
496                 PRINT (LTERMLINE,-LENGTH,0);
497                 END;
498
499                 END;  <<"" END OF CASE "">>
500               END;
501
502               END;  <<"" END OF CASE "">>
503             END;
504           END
505         UNTIL (TIME'2'QUIT OR NOT OK);
506     $PAGE
507         <<**************************************************>>
508         <<"" Now close the files and unlock the global RIN. "">>
509         <<**************************************************>>
510
511         UNLOCKGLORIN (INTEGER(RUN'PARM));
512
513         FCLOSE (MSGM'FILE,0,0);
514         IF <> THEN
515           FS'ERROR (1,MSGM'FILE,2);
516
517         FCLOSE (MSGA'FILE,0,0);
518         IF <> THEN
519           FS'ERROR (2,MSGA'FILE,2);
520
521         FCLOSE (MSGB'FILE,0,0);
522         IF <> THEN
523           FS'ERROR (3,MSGB'FILE,2);
524         END;
525     $PAGE
526     END.
```

In Figure 3 below, Process A is diagrammed to use the ;INFO= parameter of the :RUN command to locate the program suffix of A or B. It also contains the primary and secondary global RINs for the password of PROGRAMx, where x is the program suffix.

The program issues a conditional lock on the primary global RIN. If successful, the program continues; otherwise, an error message is displayed since some other process currently has this RIN locked (this prevents two copies of the program from executing because another copy is already executing).

Program A opens two message files: the primary message file named 'A' is opened for read access; the monitor message file named 'M' is opened for write access in order to communicate with the monitor process M.

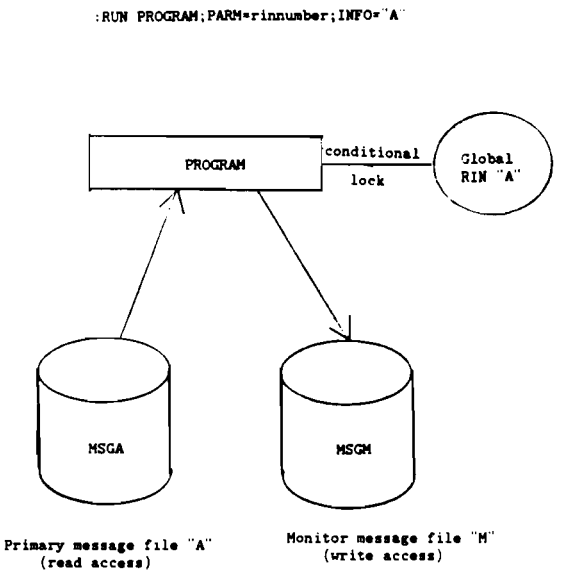


Figure 2: Program "A" Execution Flow Diagram

The second program example has been coded in COBOLII for comparison to that in SPL mentioned earlier. The variable names correspond closely to those used in the SPL version of the monitor program.

Figure 4: Program 'x' in COBOLII

```
1     $CONTROL SOURCE,MAP,CROSSREF,VERBS
1.1    IDENTIFICATION DIVISION.
1.2    PROGRAM-ID. PROGRAM.
1.3    AUTHOR. Benedict G. Bruno.
1.4    DATE-WRITTEN. June, 1986.
1.5    DATE-COMPILED.
1.6    REMARKS. The PROGRAM program will lock the global RIN name of
1.7             'PROGRAMx' with the RIN number passed in the ;INFO=
1.8             string and the 'x' replaced from the ;INFO=.
1.9
2      ENVIRONMENT DIVISION.
2.1    CONFIGURATION SECTION.
2.2
2.3    SOURCE-COMPUTER. HP3000.
2.4    OBJECT-COMPUTER. HP3000.
2.5
2.6    SPECIAL-NAMES. CONDITION-CODE IS COND-CODE.
2.7
2.8    DATA DIVISION.
2.9   $PAGE " "
3      WORKING-STORAGE SECTION.
3.1
3.2    01 TIME-2-QUIT          PIC X(1) VALUE "N".
3.3    01 OK                   PIC X(1) VALUE "Y".
3.4
3.5    01 INPUT-REC.
3.6       03 CONTROL-CODE       PIC S9(4) COMP.
3.7
3.8    01 DOUBLE-VARIABLES.
3.9       03 RUN-PARMD         PIC S9(9) COMP.
4         03 RUN-PARMI REDEFINES RUN-PARMD.
4.1          05 RUN-PARM1      PIC S9(4) COMP.
4.2          05 RUN-PARM2      PIC S9(4) COMP.
4.3
4.4    01 LOGICAL-VARIABLES.
4.5       03 DONE               PIC S9(4) COMP.
4.6       03 LOCK-COND          PIC S9(4) COMP.
4.7       03 TRUE-COND          PIC S9(4) COMP VALUE -1.
4.8
4.9    01 INTEGER-VARIABLES.
5         03 CERROR             PIC S9(4) COMP.
5.1       03 CPARM              PIC S9(4) COMP.
5.2       03 FERROR             PIC S9(4) COMP.
5.3       03 I                  PIC S9(4) COMP.
5.4       03 IN-LENGTH          PIC S9(4) COMP.
5.5       03 INFO-LENGTH        PIC S9(4) COMP.
5.6       03 LEN                PIC S9(4) COMP.
5.7       03 MSGM-FILE          PIC S9(4) COMP.
5.8       03 MSGX-FILE          PIC S9(4) COMP.
5.9       03 NUM-CHAR           PIC S9(4) COMP.
6         03 PAUSE-RIN          PIC S9(4) COMP.
6.1       03 PROCESS-RIN        PIC S9(4) COMP.
6.2
6.3    01 CHARACTER-VARIABLES.
6.4       03 TERMLINE           PIC X(79).
6.5       03 BA1                PIC X(160).
6.6       03 CERROR-A           PIC X(5).
6.7       03 CPARM-A            PIC X(5).
6.8
6.9    01 INFO-STRING.
```

The procedure division follows below with the main driver section. The START-UP paragraph performs the initialization tasks, namely the retrieval of the primary and secondary global RINs as below.

Figure 4: Program 'x' in COBOLII (Cont.)

```
7         03 PROGRAM-NAME       PIC X(1).
7.1       03 PROCESS-RIN-A      PIC X(3).
7.2       03 PAUSE-RIN-A        PIC X(3).
7.3   $PAGE " "
7.4    PROCEDURE DIVISION.
7.5
7.6   ****************************************************************
7.7   **** Start of main driver section.                          ****
7.8   ****************************************************************
7.9
8      MAIN SECTION.
8.1        PERFORM START-UP.
8.2        PERFORM PROCESS-INPUT UNTIL TIME-2-QUIT = "Y" OR OK = "N".
8.3        PERFORM FINISH-UP.
8.4        STOP RUN.
8.5   $PAGE
8.6   ****************************************************************
8.7   **** The initialization section displays the program banner, ****
8.8   **** locate the ;INFO= parameters, and open files.          ****
8.9   ****************************************************************
9
9.1    START-UP.
9.2
9.3        MOVE SPACES TO TERMLINE.
9.4        STRING "*** PROGRAM *** B.01.00 Copyr. 1986, "
9.5               DELIMITED BY SIZE
9.6               "Benge Bruno.  All rights reserved."
9.7               DELIMITED BY SIZE INTO TERMLINE.
9.8        DISPLAY TERMLINE.
9.9        MOVE SPACES TO TERMLINE.
10         CALL INTRINSIC "DATELINE" USING TERMLINE.
10.1       DISPLAY TERMLINE.
10.2       MOVE SPACES TO TERMLINE.
10.3       DISPLAY TERMLINE.
10.4
10.5       CALL "GET'INFO" USING RUN-PARMD INFO-LENGTH INFO-STRING.
10.6
10.7       IF PROGRAM-NAME NOT = "A" AND PROGRAM-NAME NOT = "a" AND
10.8          PROGRAM-NAME NOT = "B" AND PROGRAM-NAME NOT = "b" THEN
10.9           MOVE SPACES TO TERMLINE
11             MOVE "** Invalid program suffix passed. **" TO TERMLINE
11.1           DISPLAY TERMLINE
11.2           STOP RUN
11.3       .
11.4
11.5       IF PROCESS-RIN-A NOT NUMERIC THEN
11.6           MOVE SPACES TO TERMLINE
11.7           MOVE "** Invalid process RIN value. **" TO TERMLINE
11.8           DISPLAY TERMLINE
11.9           STOP RUN
12         .
12.1
12.2       IF PAUSE-RIN-A NOT NUMERIC THEN
12.3           MOVE SPACES TO TERMLINE
12.4           MOVE "** Invalid pause RIN value. **" TO TERMLINE
12.5           DISPLAY TERMLINE
12.6           STOP RUN
12.7       .
12.8  $PAGE
12.9  ****************************************************************
```

A conditional lock is executed for the primary global RIN as below. Once completed, the process and monitor message files are opened.

Figure 4: Program 'x' in COBOLII (Cont.)

```
13      **** Lock RIN 'PROGRAM' to ensure that only one copy of this ****
13.1    **** program is active.  Locate the RIN number from the      ****
13.2    **** ;INFO= parameter.                                       ****
13.3    ****************************************************************
13.4
13.5        MOVE SPACES TO BA1.
13.6        STRING "PROGRAM" DELIMITED BY SIZE
13.7               PROGRAM-NAME DELIMITED BY SIZE INTO BA1.
13.8        MOVE PROCESS-RIN-A TO PROCESS-RIN.
13.9        MOVE 0 TO LOCK-COND.
14
14.1        CALL INTRINSIC "LOCKGLORIN" USING PROCESS-RIN LOCK-COND BA1.
14.2
14.3        IF COND-CODE > 0 THEN
14.4            MOVE SPACES TO TERMLINE
14.5            MOVE "** Program is currently executing. **" TO TERMLINE
14.6            DISPLAY TERMLINE
14.7            MOVE "** This process cannot continue. **" TO TERMLINE
14.8            DISPLAY TERMLINE
14.9            STOP RUN
15          ELSE
15.1            IF COND-CODE < 0 THEN
15.2                MOVE SPACES TO TERMLINE
15.3                STRING "** LOCKGLORIN intrinsic error: RIN# = "
15.4                       DELIMITED BY SIZE
15.5                       PROCESS-RIN-A DELIMITED BY SIZE
15.6                       ".  RIN password = PROGRAM. **"
15.7                       DELIMITED BY SIZE INTO TERMLINE
15.8                DISPLAY TERMLINE
15.9                MOVE "** This process cannot continue. **" TO
16                     TERMLINE
16.1                STOP RUN
16.2            .
16.3    $PAGE
16.4    ****************************************************************
16.5    **** Now open the MONITOR and PROGRAMx message files for      ****
16.6    **** read and write access.                                   ****
16.7    ****************************************************************
16.8
16.9        MOVE "MSGM " TO TERMLINE.
17          CALL INTRINSIC "FOPEN" USING  TERMLINE %30105 %2303
17.1                                      GIVING MSGM-FILE.
17.2        IF COND-CODE NOT = 0 THEN
17.3            CALL "FS'ERROR" USING 1 MSGM-FILE 1
17.4        .
17.5
17.6        MOVE SPACES TO TERMLINE.
17.7        STRING "MSG" DELIMITED BY SIZE
17.8               PROGRAM-NAME DELIMITED BY SIZE INTO TERMLINE.
17.9        CALL INTRINSIC "FOPEN" USING  TERMLINE %30105 %2300
18                                        GIVING MSGX-FILE.
18.1        IF COND-CODE NOT = 0 THEN
18.2            CALL "FS'ERROR" USING 2 MSGX-FILE 1
18.3        .
18.4
18.5    ****************************************************************
18.6    **** Now enable extended waits on empty files for read        ****
18.7    **** access and full files for write access.                  ****
18.8    ****************************************************************
18.9
```

The extended wait facility is enabled as below. This completes the START-UP paragraph and the initialization function. The PROCESS-INPUT paragraph is called from the main driver to process input requests until a shutdown is received. The SUSPEND-PROCESS paragraph is executed when the monitor program requests us to suspend. This is accomplished by unconditionally locking the secondary global RIN as below.

Figure 4: Program 'x' in COBOLII (Cont.)

```
19          CALL INTRINSIC "FCONTROL" USING MSGM-FILE 45 TRUE-COND.
19.1        IF COND-CODE NOT = 0 THEN
19.2            CALL "FS'ERROR" USING 1 MSGM-FILE 5
19.3        .
19.4
19.5        CALL INTRINSIC "FCONTROL" USING MSGX-FILE 45 TRUE-COND.
19.6        IF COND-CODE NOT = 0 THEN
19.7            CALL "FS'ERROR" USING 2 MSGX-FILE 5
19.8        .
19.9    $PAGE
20       PROCESS-INPUT.
20.1
20.2    ****************************************************************
20.3    **** Now execute the main body of the loop waiting for the   ****
20.4    **** control codes to process.                               ****
20.5    ****************************************************************
20.6
20.7        CALL INTRINSIC "FREAD" USING  MSGX-FILE INPUT-REC -240
20.8                               GIVING IN-LENGTH.
20.9        IF COND-CODE NOT = 0 THEN
21              CALL "FS'ERROR" USING 1 MSGM-FILE 3
21.1        .
21.2
21.3        IF CONTROL-CODE < -2 OR CONTROL-CODE > -1 THEN
21.4            MOVE "** Invalid control code received.  Ignored. **" TO
21.5                TERMLINE
21.6            DISPLAY TERMLINE
21.7        ELSE
21.8            IF CONTROL-CODE = -1 THEN
21.9                MOVE "Y" TO TIME-2-QUIT
22              ELSE
22.1                IF CONTROL-CODE = -2 THEN
22.2                    PERFORM SUSPEND-PROCESS
22.3        .
22.4    $PAGE
22.5     SUSPEND-PROCESS.
22.6
22.7    ****************************************************************
22.8    **** Now suspend the program by unconditionally locking the  ****
22.9    **** pause RIN.  When the RIN lock completes, we release     ****
23      **** it immediately, thus resuming execution.                ****
23.1    ****************************************************************
23.2
23.3        MOVE 1 TO LOCK-COND.
23.4        MOVE PAUSE-RIN-A TO PAUSE-RIN.
23.5        MOVE SPACES TO BA1.
23.6        STRING "PROGRAM" DELIMITED BY SIZE
23.7               PROGRAM-NAME DELIMITED BY SIZE INTO BA1.
23.8
23.9        CALL INTRINSIC "LOCKGLORIN" USING PAUSE-RIN LOCK-COND BA1.
24          IF COND-CODE < 0 THEN
24.1            MOVE SPACES TO TERMLINE
24.2            MOVE "** LOCKGLORIN intrinsic error: RIN#=" TO TERMLINE
24.3            DISPLAY TERMLINE
24.4            MOVE "N" TO OK
24.5        ELSE
24.6            CALL INTRINSIC "UNLOCKGLORIN" USING PAUSE-RIN
24.7        .
24.8    $PAGE
24.9     FINISH-UP.
```

The FINISH-UP paragraph below completes program execution by unlocking the primary global RIN and closing the message files.

Figure 4: Program 'x' in COBOLII (Cont.)

```
25
25.1  ****************************************************************
25.2  **** Now close the files and unlock the global RIN.          ****
25.3  ****************************************************************
25.4
25.5        CALL INTRINSIC "UNLOCKGLORIN" USING PROCESS-RIN.
25.6
25.7        CALL INTRINSIC "FCLOSE" USING MSGM-FILE 0 0.
25.8        IF COND-CODE NOT = 0 THEN
25.9            CALL "FS'ERROR" USING 1 MSGM-FILE 2.
26
26.1        CALL INTRINSIC "FCLOSE" USING MSGX-FILE 0 0.
26.2        IF COND-CODE NOT = 0 THEN
26.3            CALL "FS'ERROR" USING 2 MSGX-FILE 2.
```

In order to execute the monitor, A, and B programs, the global RINs must be located. The GETRIN MPE command is executed for each of the program passwords. Figure 5 below displays the logon and GETRIN commands. The bold face type denotes input user input.

Figure 5: Executing the GETRIN command

```
:HELLO BENGE.BRUNO (RETURN)
HP3000 / MPE V G.01.01 (BASE G.01.01). SUN, MAY 25, 1986, 10:10 AM
:GETRIN MONITOR (RETURN)
RIN: 122
:GETRIN PROGRAMA (RETURN)
RIN: B7
:GETRIN PROGRAMA (RETURN)
RIN: 139
:GETRIN PROGRAMB (RETURN)
RIN: 126
:GETRIN PROGRAMB (RETURN)
RIN: 129
:BYE (RETURN)

CPU=1. CONNECT=3. SUN, MAY 25, 1986, 10:13 AM
```

The SHOWRIN Program

The SHOWRIN program provides a formatted display of the system global RIN table. The program is written using privileged mode for execution on any MPE IV and MPE V based systems. The program may only execute in session mode and will immediately terminate in batch mode.

The group that SHOWRIN executes must have PM capability. For this reason, you may wish to place it in PUB.SYS or UTIL.SYS where PM is already in the group. The capability list of the user is checked when the SHOWRIN program is executed. If the user has system manager (SM) or privileged mode (PM) capability, then the SHOWRIN program may display the RIN table for all accounts; otherwise, only the user logon account may be used.

The SHOWRIN program may simply be executed using the :RUN statement. The alternate entry point of 'NOHELP' may be used to disable the initial help information as in Figure 6 below.

```
:HELLO BENGE,MANAGER.SYS (RETURN)
:RUN SHOWRIN (RETURN)

*** SHOWRIN *** B.01.00 Copyr. 1986, Benge Bruno. All rights reserved.
SUN, MAY 25, 1986, 7:30 PM

The SHOWRIN program provides a display of global RINs on your HP 3000.
Each of the commands are entered as single characters as in OPT/3000.
The following commands are available.

A -> Prompt for a specific account name; search on this account only.
B -> Display both locked and unlocked RIN entries found.
E -> Exit the program.
H -> Display this help information.
L -> Display locked RIN entries only.
P -> Enable/disable printing to RINLIST on device LP.
R -> Prompt for RIN numbers for additional information.
S -> Display RINs for all accounts; Requires PM or SM capability.
U -> Display unlocked RIN entries only.
W -> Display waiting processes and the holding processes (deadlock?).

To disable this initial help dialogue, use the entry point of 'NOHELP'.

Please press any key to continue.
```

Figure 6: Running the SHOWRIN program

After pressing the return key, the system RIN table will be displayed with format in Figure 7 below.

```
System RIN Table        LP    System-wide, LOCK ONLY                    6:05 PM

RIN      RIN          Creating
 #     password   Username.Acctname     PIN   Jobnum          Job name
----   --------   ----------------     ----  ------  --------------------------
```

Figure 7: SHOWRIN header display

Note that the top line contains the current system time, LP, and RIN selection. Since the logged on user has PM or SM capability, the RIN table will be checked for all accounts. This is noted with the 'System-wide' literal. If PM or SM is not detected, then only the logged on account will be searched. This is noted with the 'Account: xxxxxxxx' literal.

Several dispositions of the global RINs may be selected. The 'LOCK ONLY' option displays only those RINs that are currently locked by a process. The 'UNLOCK ONLY' option displays only those RINs that are currently free. The 'LOCK w/WAIT' option displays only those RINs that are currently locked by a process and the processes that are waiting on these RINs. The 'LOCK, UNLOCK, WAIT' option displays all entries in the table.

When the print option is enabled, an asterisk (*) is placed to the left of the LP designator in the header. The asterisk is removed when the LP option is disabled.

Entries found in the RIN table are displayed in ascending sorted order. For each entry found, the four digit RIN value, password, and creating user and account are displayed. If the RIN is currently locked, then the PIN, job or session number, and the job/session, user, and account of the locking process is displayed. An additional line is displayed with the PIN, job or session number, and the job/session, user, and account of any waiting processes.

The SHOWRIN program has single character terminal reads enabled every 20 seconds. Should no data be received, then a timeout occurs and the table is displayed using the same selection criteria.

To continue with our simple example of the monitor program and two external programs, we must first acquire the global RINs. This requires that the GETRIN MPE command be executed for the values of

'MONITOR', 'PROGRAMA', and 'PROGRAMB' with the program values executed twice; once for the primary process RIN and second for the secondary pause RIN.

Since the RIN values assigned by MPE are system dependent, I do not want to even suggest what they may be. However, using the SHOWRIN program, we may locate their values by using the 'U' for unlock or 'B' for all entries within the account which you have created these RINs. Having performed this in my own account, I may use the SHOWRIN program as in Figure 8 below to locate the (currently) unlocked RIN values and the associated passwords.

```
System RIN Table        LP    Account: BRUNO, LOCK,UNLOCK,WAIT          6:56 PM

RIN      RIN          Creating
 #     password   Username.Acctname     PIN   Jobnum          Job name
----   --------   ----------------     ----  ------  -------------------------
  87   PROGRAMA   BENGE.BRUNO
 122   MONITOR    BENGE.BRUNO
 126   PROGRAMB   BENGE.BRUNO
 129   PROGRAMB   BENGE.BRUNO
 139   PROGRAMA   BENGE.BRUNO
```

Figure 8: Locating all RIN entries for a specific account

If the monitor, program A, and program B processes are initiated, then each program will lock its corresponding global RIN. Each process will await input by reading its associated message file. Figure 9 below shows this initial state, whereby each RIN value is displayed with the locking process.

```
System RIN Table        LP    Account: BRUNO, LOCK ONLY                 7:11 PM

RIN      RIN          Creating
 #     password   Username.Acctname     PIN   Jobnum          Job name
----   --------   ----------------     ----  ------  -------------------------
  87   PROGRAMA   BENGE.BRUNO          153   #J397   A,BENGE.BRUNO .
 122   MONITOR    BENGE.BRUNO          251   #S445   BENGE.BRUNO
 126   PROGRAMB   BENGE.BRUNO          121   #J398   B,BENGE.BRUNO
```

Figure 9: Monitor, program A, program B normal execution

A useful feature of the global RINs discussed earlier is to use them for process control. If the monitor process were to lock the secondary pause RIN conditionally, programs A or B may be suspended if they attempt to lock the secondary pause RIN unconditionally. Figure 10 below displays this situation. You will note the previous entries as in Figure 9 for each of the processes primary RIN. The additional entries are showing that the secondary pause RINs are owned by the monitor process executed by session number 445. Programs A and B executing from job streams are then executing an unconditional lock for the pause RIN.

```
System RIN Table        LP    Account: BRUNO, LOCK ONLY                7:22 PM

RIN      RIN          Creating
 #     password    Username.Acctname     PIN   Jobnum        Job name
----   --------   -----------------     ----   ------   -------------------------
 87    PROGRAMA    BENGE.BRUNO           157   #J402    A,BENGE.BRUNO
122    MONITOR     BENGE.BRUNO           159   #S445    BENGE.BRUNO
126    PROGRAMB    BENGE.BRUNO           122   #J403    B,BENGE.BRUNO
129    PROGRAMB    BENGE.BRUNO           159   #S445    BENGE.BRUNO
                                         122   #J403    B,BENGE.BRUNO
139    PROGRAMA    BENGE.BRUNO           159   #S445    BENGE.BRUNO
                                         157   #J402    A,BENGE.BRUNO
```

Figure 10: Suspending programs A and B

Figure 11 below displays only the pertinent information from Figure 10 above in that only those RINs with processes waiting are displayed. This is effective in reducing the entries in the display to detect potential deadlocks. Looking closely at the entries in Figure 11 below, we see that the primary RIN values are not included.

```
System RIN Table        LP    Account: BRUNO, LOCK w/WAIT              7:22 PM

RIN      RIN          Creating
 #     password    Username.Acctname     PIN   Jobnum        Job name
----   --------   -----------------     ----   ------   -------------------------
129    PROGRAMB    BENGE.BRUNO           159   #S445    BENGE.BRUNO
                                         122   #J403    B,BENGE.BRUNO
139    PROGRAMA    BENGE.BRUNO           159   #S445    BENGE.BRUNO
                                         157   #J402    A,BENGE.BRUNO
```

Figure 11: Avoid deadlocks, display locked RINs w/waiting processes

Real-time Terminal Monitor

In the global RIN example discussed earlier, the MONITOR program controlled two external programs. This example can be further extended whereby the MONITOR program can become a real-time terminal monitor effectively displaying the status of virtually any process within the application which it is monitoring. How might this be done? By using a combination of message files, global RINs, and a separate process to read the terminal keyboard.

The idea is to split the MONITOR program into virtually two separate processes. The process executed from the :RUN statement will open the $STDLIST of the specific logical device number either passed within the ;PARM= or ;INFO= parameters or by using the FGETINFO (or FFILEINFO) intrinsics. This father process we will call MONITORF for MONITOR FATHER. His job is to monitor all external processes and respond to the application operator requests. The information is passed to the MONITORF process via a message file. All external processes write messages into this message file such as 'start, stop, suspend, resume, etc'. (From this strategy it is obvious that the external processes open the message file for write (FOPTIONS=%30105, AOPTIONS=%2303) whereas the MONITORF process opens the message file for read (FOPTIONS=%30105,AOPTIONS=%2300).

By enabling the writers identification feature of this message file, the MONITORF program receives two additional words for each FREAD of the message file. Within these two words, are the file access of open, close, or data (for external write) and the writers id. The writers id is an ascending unique number which MPE assigns to the process starting from one (1) for each process as the process opens the file. The only difficulty is to associate the MPE assigned writers id with the process opening, writing, and closing the file.

The open file access can really be ignored, since we really do not care who opens the file. It is during the write and close that we really should care about. Why? Because if a close record is detected unexpectedly, then the MONITORF process can quickly and accurately respond. When would we detect an unexpected close record? Recall, that when a processes terminates unexpectedly, MPE closes all files within the TERMINATE procedure. Message files are no different but a close record (including the writers id!) is sent to the MONITORF process. So the MONITORF process knows exactly when processes abort! Maybe the reaction is for the MONITORF process to automatically restart the individual failing process or shutdown all of the remaining (active) processes. This is a decision to be made by the applications programmer.

How does one associate the writers id with the process? I suggest that as soon as each external process opens all its files, data bases, spool files, etc. and reaches a steady idle state, i.e., the process

itself may be ready to read its own message file, then send some kind of acknowledgement record to the MONITORF message file with a predefined unique value assigned to this process. The MONITORF process receives this data record (from the external process write) with the unique id in it, knows of this id, and assigns it within a table of all values being received. Thus, when a close record is detected, the MONITORF program simply searches this internal table to find the unique predefined value of the process that was previously stored.

But if the MONITORF program is to process incoming messages from the external processes and also monitor terminal input from the user, how does it read both this message file and the terminal? Good question. The way this can be implemented is really rather neat. An additional process that we will call MONITORR for monitor read is responsible for reading the terminal keyboard. How, when, and what to look for will be the discretion of the MONITORF process. This can be conveyed with message files (what else!).

Before going any further let us draw a diagram as in Figure 12 below of the situation:

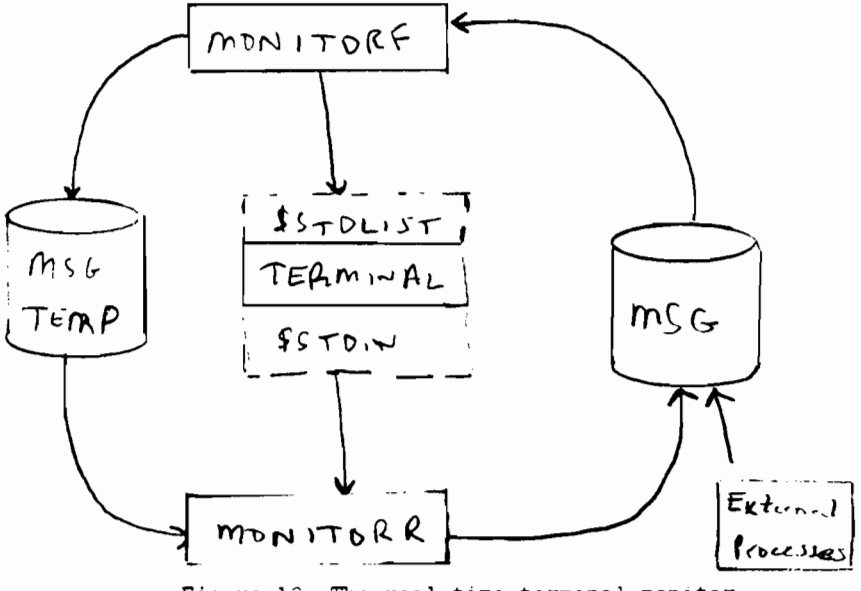


Figure 12: The real-time terminal monitor

Notice that the permanent message file named MSG is being read by the MONITORF process. This message file is the one opened for write access by all of the external processes. The additional temporary message file named MSGTEMP is being written by the MONITORF process. (For all practical purposes the name may be MSG as the permanent since

the files reside in different domains, namely the permanent and temporary file domains.) This temporary file is being read by the MONITORR process. Since the instructions of the terminal read are only to be specified by the MONITORF program and accepted by the MONITORR program, a temporary message file is used so that no other external process can interfere. That is why I think that message files are so powerful.

All that is left is to determine the format of the terminal requests with which the MONITORR process should respond. Things to think about are the following:

* Timed versus non-timed (infinite) terminal reads.

* Specific length reads.

* Function keys, ENTER, or carriage return (CR) terminated reads.

* Control-Y terminated reads.

* Lower versus upshifted data, i.e., should the data received be upshifted before return.

Now that we know what the MONITORR process is to look for, how is he to relay the information back to the MONITORF process? With a message file of course. Which one? Obviously the only one which the MONITORF program is reading which is the permanent message file named MSG! So, another set of codes must be used to denote the type of read completed. I suggest that you use negative integer values so that no one can cause strange problems using FCOPY to load any of the message files.

As an example, whenever the MONITORF process wishes a response to a prompt that he placed on the terminal screen, a special message is written to the MSGTEMP message file. Let's suppose we want a three character authorization code with a default value that is displayed. The user may enter three characters of data, the carriage return (in order to accept the default), or a timeout may occur. Why the timeout? Because we do not want to wait forever here on this terminal read, because we may want to continue on with other prompts and if the timeout does occur, we will just suppose that the user cannot make up his mind within a relatively short period of time. The MONITORR process completes the MSGTEMP read, and issues the appropriate parameters within the FREAD of the terminal. When the read completes (and it will!), the MONITORR process simply writes the read information with the appropriate code in to the permanent MSG file. Then since we must await another read request, the MONITORR process returns to reading the temporary MSGTEMP message file. The MONITORF process reads the terminal input from the permanent MSG file and reacts accordingly, i.e., whether a timeout occurred, a carriage

return was encountered, or the three characters of data he may expect. Then simply repeat the whole concept all over again.

Probably now is a good time to discuss some of the inherent problems with this approach. I have noticed that it is possible to obtain file system errors 27 and 28 during the read executed by the MONITORR process. I simply ignored them and reissued the read. If two occurred back to back, then I aborted the MONITORR process which causes the MONITORF process to bring the whole process tree down and then is simply started up once again from the beginning.

The only other problem is how can you issue a write to a terminal by the MONITORF program without terminating the read to the terminal by the MONITORR program? Why would the MONITORF program wish to write while he has told the MONITORR program to read? How about defining a complete 24 by 80 character display whereby specific rows, columns, lengths, etc. denote actions by all processes on the system? Now that is what I mean by a monitor. The monitor is really a predefined display for each of the processes on the system. By highlighting the display with different display enhancements, it is quite easy to devise a strategy whereby the status of the entire application can be made quickly and accurately.

In order to understand why a process would want to write to a terminal when another is reading an example might help. Recall that while you are logged on and MPE is awaiting your next command, i.e., the command interpreter has issued the colon (:) followed by the DC1 to enable the keyboard, some other MPE user wants to write to our terminal. How about the MPE :TELL command, the console request, etc? Each of these writes will what is called preempt the pending read, allow MPE to write the message, and then reissue the read. However the next read causes a CR/LF/DC1 to be issued. That is why when the console is logged on, that for each message a blank line is encountered where while no session is logged onto the console, then no blank line is found. It is for this reason, that the preemptive write is used to update information on the display screen by the MONITORF process (really an FWRITE to $STDLIST) which aborts and reissues the pending read of the MONITORR process (really an FREAD of $STDINX). How is this done? Unfortunately the preemptive write requires calling the privileged procedure known as ATTACHIO. This procedure is responsible for doing the real I/O to virtually any device on the HP3000. By simply defining a new procedure called file write preemptive (FWRITEP for short), we may issue this write to the display screen at any time and be assured that the read of the MONITORR process is reissued. This took some understanding of the ATTACHIO internals and how the parameters are passed. The FWRITEP procedure used the same procedure parameters as the FWRITE procedure and used the parameters to form the call to ATTACHIO.

As you can see, this real-time terminal monitor is really quite neat and not too bad to implement. There is really nothing wrong with using the ATTACHIO procedure. You must realize that all of MPE uses it, and it remains the same as executed in compatibility mode on the new MPE/XL systems. All that is left is to define two procedures on the MONITORF process which will request a terminal read from the MONITORR process (which I will call READ'TERMINAL) and another to process any input from the permanent message file named MSG (which I will call PROCESS'INPUT). Thus within READ'TERMINAL will be an FWRITE to the temporary message file and within PROCESS'INPUT will be an FREAD of the permanent message file.

I will now offer sample code in SPL that actually is used in a working application. Following the coding examples of the FWRITEP, READ'TERMINAL, PROCESS'INPUT, and MONITORR process, I offer a conclusion so please read on.

The code for the FWRITEP procedure follows below. You will notice that the call is nearly identical to the standard FWRITE except that the logical device number of the terminal NOT THE FILE NUMBER FROM THE FOPEN is required since the ATTACHIO procedure uses it. Notice that this code can be compiled with the SPL compiler and placed in an SL. There is no need to PREP your own program with PM in order to call it. All that is necessary is that GROUP and ACCOUNT where this resides must have PM capability such as SL.PUB.SYS. As with any external procedure, it will be resolved at :RUN time and off it goes!

```
$CONTROL SUBPROGRAM,MAP,SEGMENT=FWRITEP
BEGIN

INTEGER PROCEDURE  FWRITEP (LDEV,BUFFER,COUNT,CCTL);
           VALUE  LDEV,COUNT,CCTL;
         INTEGER  LDEV,COUNT,CCTL;
   LOGICAL ARRAY  BUFFER;
          OPTION  PRIVILEGED;

  BEGIN

<<*****************************************************************>>
<<** This procedure will perform a preemptive write to the **>>
<<** logical device number passed as LDEV.  Usually used   **>>
<<** to write to the system console as in the MPE :TELL     **>>
<<** command.  If the user has not begun to enter any data,**>>
<<** this procedure will abort the pending read, write the **>>
<<** data, and inform MPE to re-issue the next read.  This **>>
<<** will cause an EXTRA CR/LF to be sent to the terminal. **>>
<<** If the user has begun to key at least one (1) data    **>>
<<** character, the write will be performed once the user  **>>
<<** hits CR.                                              **>>
<<*****************************************************************>>
```

```
$PAGE
COMMENT  LOCAL VARIABLES
*************************************************************;

INTEGER       ERROR'CODE;
LOGICAL       CONDITION'CODE,
              STATUS=Q-1;

 DOUBLE PROCEDURE  ATTACHIO (LDEV,V1,V2,V3,V4,V5,V6,V7,V8);
            VALUE  LDEV,V1,V2,V3,V4,V5,V6,V7,V8;
          INTEGER  LDEV,V1,V2,V3,V4,V5,V6,V7,V8;
           OPTION  EXTERNAL;

INTEGER PROCEDURE  IOSTAT (ERROR'CODE);
            VALUE  ERROR'CODE;
          INTEGER  ERROR'CODE;
           OPTION  EXTERNAL;
$PAGE
     <<*********************************************************>>
     <<                Main Procedure Code                      >>
     <<*********************************************************>>

     BEGIN
     PUSH (STATUS);
     TOS.(2:1):=0; <<** Disable Traps **>>
     SET (STATUS);

     TOS:=ATTACHIO (LDEV,0,0,@BUFFER,1,COUNT,CCTL,0,%201);
     DEL;
     ERROR'CODE:=TOS;

     IF ERROR'CODE=1 THEN
       CONDITION'CODE:=2
     ELSE
       BEGIN
       CONDITION'CODE:=1;
       FWRITEP:=IOSTAT (ERROR'CODE);
       END;

     STATUS.(6:2):=CONDITION'CODE;
     END;
$PAGE
   END;
END.
```

The code for the READ'TERMINAL procedure follows below:

```
<<*************************************************************>>
<<** PROCEDURE  READ'TERMINAL (CODE,LENGTH)                  **>>
<<**                                                         **>>
<<** This procedure primes the terminal read son named       **>>
<<** MONITORR to return input to the permanent message       **>>
<<** file named MSG.  Two parameters are passed:             **>>
<<**                                                         **>>
<<** Code      -1 Close terminal if open.                    **>>
<<**           0 perform infinite read.                      **>>
<<**          >0 perform timed read for (code) seconds.      **>>
<<**                                                         **>>
<<**  Length  >0 perform read for (length) bytes.            **>>
<<**           0 return only function key input plus CR.     **>>
<<*************************************************************>>

PROCEDURE READ'TERMINAL (CODE,LENGTH);
    VALUE CODE,LENGTH;
  INTEGER CODE,LENGTH;

BEGIN
LTERMLINE:=CODE;
LTERMLINE(1):=LENGTH;
FWRITE (MSGTEMP'FILE, LTERMLINE, -4, 0);
IF <> THEN
  FS'ERROR (1, MSGTEMP'FILE, 4);
END;
```

The code for the PROCESS'INPUT procedure follows below:

```
<<*************************************************************>>
<<** PROCEDURE  PROCESS'INPUT (TYPE)                         **>>
<<**                                                         **>>
<<** This procedure processes the input from the permanent   **>>
<<** message file named MSG.  The input could be from many   **>>
<<** different sources, such as the MONITORR son process,    **>>
<<** or any of the external or satellite processes.          **>>
<<**                                                         **>>
<<** 1. If the input is terminal input and the passed type   **>>
<<**    is a one (1), then exit to the calling procedure.    **>>
<<**                                                         **>>
<<** 2. If the input contains an integer value >0 and <100   **>>
<<**    then the input is from an external process.  Use     **>>
<<**    this input to update the display screen but do not   **>>
<<**    return to the caller.                                **>>
<<**                                                         **>>
<<** 3. If the input does not contain an integer value in    **>>
<<**    in the first word and the type is >1, then it is     **>>
<<**    from an external process during its initialization   **>>
<<**    which requires an exit to the caller.                **>>
<<**                                                         **>>
<<** 4. If the read timeout occurs (as set with the type     **>>
<<**    value if type is >1), then exit to the calling       **>>
<<**    procedure.                                           **>>
<<**                                                         **>>
<<** Keep track of the writers ids for external processes    **>>
<<** so the screen update display can be called when close   **>>
<<** records are detected to display the aborting info on    **>>
<<** the screen during real time.                            **>>
<<*************************************************************>>

PROCEDURE  PROCESS'INPUT (TYPE);
    VALUE  TYPE;
  INTEGER  TYPE;

BEGIN
WHILE TRUE DO
  BEGIN
  IF TYPE > 1
    THEN L:=TYPE
    ELSE L:=0;

  FCONTROL (MSG'FILE,2,L);
  IF <> THEN
    FS'ERROR (2,MSG'FILE,5);

  IN'LENGTH:=FREAD (MSG'FILE,LRESPONSE'BUFFER,-4096);
  IF <> THEN
```

```
      BEGIN
      FCHECK (MSG'FILE,FERROR);
      IF FERROR = 22 THEN
        BEGIN
        LRESPONSE'BUFFER:= -3;       <<** Show Timeout **>>
        RETURN;
        END
      ELSE
        BEGIN
        FS'ERROR (3,MSG'FILE,3);
        RETURN;
        END;
      END;

    <<****************************************>>
    <<** Process according to record type. **>>
    <<****************************************>>

    CASE LRESPONSE'BUFFER.(14:2) OF
      BEGIN

<<0>> BEGIN <<** DATA RECORD **>>
      I:=INTEGER(LRESPONSE'BUFFER(2));

      <<*********************************>>
      <<** Check for terminal input **>>
      <<*********************************>>

      IF I < 0 LAND I > -4 LAND TYPE = 1 THEN
        BEGIN
        J:=INTEGER(LRESPONSE'BUFFER(3));
        MOVE RESPONSE'BUFFER:=RESPONSE'BUFFER(4),(J+4);
        READ'REQUEST:=FALSE;
        RETURN;
        END
      ELSE
        BEGIN
        <<***********************************>>
        <<** Check for a process startup. **>>
        <<** Verify that the unique       **>>
        <<** process id is within range   **>>
        <<** and then save this MPE id    **>>
        <<** in the array at this offset. **>>
        <<***********************************>>

        IF (I.(8:8) < LASTPROG LAND I.(8:8) > -1) LAND
          (I.(0:8) = 0 LOR I.(0:8) = %100) THEN
          BEGIN
          IF I.(1:1) = 0 THEN
```

```
            WRITERS'ARRAY(I.(8:8)):=LRESPONSE'BUFFER(1);
          END;

        <<**********************************>>
        <<** This should be a start or fail **>>
        <<** of a process startup.          **>>
        <<**********************************>>

        IF TYPE > 1 THEN
          BEGIN
          MOVE LRESPONSE'BUFFER:=LRESPONSE'BUFFER(2),(40);
          IF RESPONSE'BUFFER = "PASS" LOR
             RESPONSE'BUFFER = "FAIL" THEN
            RETURN;
          END;
        END;
      END;

<<1>> ;     <<**  Open Records **>>

<<2>> BEGIN <<** Close Records **>>
      I:=LRESPONSE'BUFFER(1);   <<** Get Writers ID **>>

      FOR X:=0 UNTIL LASTPROG DO
        BEGIN
        IF I = WRITERS'ARRAY(X) THEN  <<** Find ID in table **>>
          BEGIN
          <<**********************************************>>
          <<** External process has aborted.  Issue    **>>
          <<** log message and highlight display screen**>>
          <<** with an abort for this process.        **>>
          <<**********************************************>>
          END;
        END;
      END;

      END;  <<** END OF CASE **>>
    END;
  END;
```

The entire code of the MONITORR process follows below:

```
$CONTROL USLINIT,MAP

<<**********************************************************************>>
<<** MONITORR -- Version Information.                                  **>>
<<**                                                                   **>>
<<** Version    Date    Who  Comments / Description                    **>>
<<** -------  --------  ---  ---------------------------------------- **>>
<<** B.01.00   x/xx/xx  BGB  Initial program release.                  **>>
<<**                                                                   **>>
<<** This process is CREATEd/ACTIVATEd by the MONITORF father          **>>
<<** process to read the terminal on the logical device number         **>>
<<** specified in the ;PARM= parameter of the :RUN statement.          **>>
<<** Communication is established to this process from the             **>>
<<** temporary MSGTEMP message file.  Two parameters are passed        **>>
<<** in this file: an INTEGER control code, and an INTEGER length**>>
<<** of the read request.  Control codes written by the father         **>>
<<** have the following meaning:                                       **>>
<<**                                                                   **>>
<<**   -2 = Send start message with Version number to MSG file.        **>>
<<**                                                                   **>>
<<**   -1 = If TERMINAL file open, then close the TERMINAL file.       **>>
<<**                                                                   **>>
<<**    0 = Perform an infinite (no timeout) read on TERMINAL.         **>>
<<**                                                                   **>>
<<**   >0 = Perform a finite (timed read) on TERMINAL from value.**>>
<<**                                                                   **>>
<<** If the requested length = 0, then only look for function          **>>
<<** keys regardless of the timeout values.                            **>>
<<**                                                                   **>>
<<** Once the desired action above is completed, this son process**>>
<<** will deposit the following control codes and data values in **>>
<<** the permanent MSG message file.  This provides the                **>>
<<** MONITORF program real time updating of the main status            **>>
<<** screen.  The son process deposits an INTEGER control code,        **>>
<<** an INTEGER function key value or string length, and if data **>>
<<** found a BYTE ARRAY containing the data read from TERMINAL.        **>>
<<** The first parameter contains the following control codes:         **>>
<<**                                                                   **>>
<<**   -1 = Function key (f1 thru f8) or [RETURN] key pressed          **>>
<<**                                                                   **>>
<<**   -2 = Read completed with data to fill buffer or terminated**>>
<<**        with a [RETURN] key.  (above return is w/o data!)          **>>
<<**                                                                   **>>
<<**   -3 = Read timeout occurred from finite read request.            **>>
<<**                                                                   **>>
<<**   -4 = Control Y trap detected.                                   **>>
<<**                                                                   **>>
<<** The second parameter will have the following meanings:            **>>
```

```
<<**                                                              **>>
<<**   If control code = -1 then PARM2 = 0 for [RETURN] or        **>>
<<**                                     1 thru 8 for f1 thru f8 **>>
<<**                                                              **>>
<<**   If control code = -2 then PARM2 = length of string AND     **>>
<<**                               PARM3 = byte string itself     **>>
<<**                                                              **>>
<<**   If control code = -3 then PARM2 and PARM3 are ignored.     **>>
<<**                                                              **>>
<<**   If control code = -4 then PARM2 and PARM3 are ignored      **>>
<<******************************************************************>>

$PAGE "*****  MONITORR -- Reads terminal keyboard  *****"
BEGIN

COMMENT  EQUATES AND DEFINES
******************************************************************;

DEFINE    MONITORR                = 7#;  <<** Unique process id **>>

EQUATE    RS                      = %036,
          BEL                     = %007,
          ESC                     = %033;
$PAGE
COMMENT   LOCAL VARIABLES
******************************************************************;

LOGICAL ARRAY          LTERMLINE(0:79);
BYTE ARRAY             TERMLINE(*)=LTERMLINE;

LOGICAL ARRAY          LBA1(0:79);
BYTE ARRAY             BA1(*)=LBA1;

LOGICAL ARRAY          LBA2(0:79);
BYTE ARRAY             BA2(*)=LBA2;

LOGICAL ARRAY          LBA3(0:79);
BYTE ARRAY             BA3(*)=LBA3;

LOGICAL ARRAY          LINFO'STRING(0:129);
BYTE ARRAY             INFO'STRING(*)=LINFO'STRING;

LOGICAL ARRAY          LINPUT'RECORD(0:79);
BYTE ARRAY             INPUT'RECORD(*)=LINPUT'RECORD;

LOGICAL ARRAY          LINPUT'RECORD2(0:79);
BYTE ARRAY             INPUT'RECORD2(*)=LINPUT'RECORD2;

LOGICAL ARRAY          LOUTPUT'RECORD(0:79);
BYTE ARRAY             OUTPUT'RECORD(*)=LOUTPUT'RECORD;
```

```
LOGICAL ARRAY           LBLANKS(0:39):=40("  ");
BYTE ARRAY              BLANKS(*)=LBLANKS;

BYTE ARRAY              DEVICE(0:9);

DOUBLE                  RUN'PARM;

LOGICAL                 L,
                        OK,
                        DONE,
                        TERM'OPEN,
                        DEBUG'MODE,
                        BENGE'MODE,
                        LOCK'COND,
                        TRUE'COND,
                        CHECK'F'KEYS'ONLY,
                        FIRST'FS'ERROR'27'28,
                        PROCESS'BEING'INITIATED;

INTEGER                 I,
                        J,
                        K,
                        LEN,
                        DEV,
                        CERROR,
                        CPARM,
                        FERROR,
                        LENGTH,
                        LENGTH1,
                        LENGTH2,
                        LENGTH3,
                        NUM'CHAR,
                        IN'LENGTH,
                        IN'LENGTH2,
                        INFO'LENGTH,
                        READ'LENGTH,
                        TERMINAL,
                        MONITORR'RIN,
                        MSG'FILE,
                        MSGTEMP'FILE;

INTRINSIC               ACTIVATE,
                        ASCII,
                        BINARY,
                        DATELINE,
                        FCHECK,
                        FCLOSE,
                        FCONTROL,
                        FERRMSG,
                        FLOCK,
```

```
                        FOPEN,
                        FREAD,
                        FREADDIR,
                        FWRITE,
                        FWRITEDIR,
                        FUNLOCK,
                        FUPDATE,
                        LOCKGLORIN,
                        PRINT,
                        PRINTFILEINFO,
                        PRINTOP,
                        READ,
                        SUSPEND,
                        TERMINATE,
                        UNLOCKGLORIN;
$PAGE
COMMENT    PROCEDURE DECLARATIONS
*****************************************************************;

PROCEDURE  GET'INFO (RUN'PARM, INFO'LENGTH, LINFO'STRING);
   DOUBLE  RUN'PARM;
  INTEGER  INFO'LENGTH;
    ARRAY  LINFO'STRING;
   OPTION  EXTERNAL;
$PAGE
      PROCEDURE  WRITE'MSG (CONTROL'CODE,PARM1,PARM2);
          VALUE  CONTROL'CODE,PARM1;
        INTEGER  CONTROL'CODE,PARM1;
     BYTE ARRAY  PARM2;

      BEGIN
      IF DEBUG'MODE THEN
        BEGIN
        MOVE BA1:=("MONITORR: Posting write to permanent ",
                   "MSG message file."),2;
        LEN:=TOS-LOGICAL(@BA1);
        PRINT ( LBA1,-LEN,0);
        END;

      LOUTPUT'RECORD:=CONTROL'CODE;
      LOUTPUT'RECORD(1):=PARM1;
      IF PARM2 <> BLANKS,(1) THEN
        BEGIN
        IF CONTROL'CODE > -90 THEN  <<** Upshift **>>
          BEGIN
          PARM2 (PARM1):=RS;
          FOR I:=0 UNTIL PARM1-1 DO
            BEGIN
            IF PARM2(I) >= %141 LAND  <<** a **>>
               PARM2(I) <= %172       <<** z **>>
```

```
            THEN PARM2(I):=PARM2(I)-%040;
          END;
        MOVE OUTPUT'RECORD(4):=PARM2,(PARM1),2;
        LENGTH:=TOS-LOGICAL(@OUTPUT'RECORD);
        END
      ELSE
        BEGIN
        MOVE OUTPUT'RECORD(4):=PARM2,(PARM1),2;
        LENGTH:=TOS-LOGICAL(@OUTPUT'RECORD);
        END;
      END
    ELSE
      LENGTH:=4;

    FWRITE (MSG'FILE,LOUTPUT'RECORD,-LENGTH,0);
    IF <> THEN
      BEGIN
      FCHECK (MSG'FILE,FERROR);
      MOVE TERMLINE:=("** MONITORR: Write of MSG ",
                      "failed. (FS error "),2;
      NUM'CHAR:=ASCII (FERROR,10,BA1);
      MOVE *:=BA1,(NUM'CHAR),2;
      MOVE *:="). **",2;
      LENGTH:=TOS-LOGICAL(@TERMLINE);
      PRINTOP (LTERMLINE,-LENGTH,0);
      TERMINATE;
      END;
    END;
$PAGE
    PROCEDURE FS'ERROR (FILE'INDEX, FILE'NUMBER, FS'INTRINSIC);
        VALUE FILE'INDEX, FS'INTRINSIC;
      INTEGER FILE'INDEX, FILE'NUMBER, FS'INTRINSIC;

    BEGIN
    FCHECK (FILE'NUMBER, FERROR);
    MOVE TERMLINE:=("** Unexpected FS error "),2;
    NUM'CHAR:=ASCII (FERROR,10,BA1);
    MOVE *:=BA1,(NUM'CHAR),2;
    MOVE *:=(" has occurred on "),2;
    CASE FILE'INDEX OF
      BEGIN
  <<0>> ;
  <<1>> MOVE *:="TERMINAL",2;
  <<2>> MOVE *:="MSG",2;
  <<3>> MOVE *:="(TEMP) MSG",2;
      END;  <<** END OF CASE **>>
    MOVE *:=(" during "),2;
    CASE FS'INTRINSIC OF
      BEGIN
  <<0>> ;
```

```
    <<1>> MOVE *:="FOPEN",2;
    <<2>> MOVE *:="FCLOSE",2;
    <<3>> MOVE *:="FREAD",2;
    <<4>> MOVE *:="FWRITE",2;
    <<5>> MOVE *:="FCONTROL",2;
    <<6>> MOVE *:="FLOCK",2;
    <<7>> MOVE *:="FUNLOCK",2;
    <<8>> MOVE *:="FUPDATE",2;
    <<9>> MOVE *:="FREADDIR",2;
          END;  <<** END OF CASE **>>
        MOVE *:=(". **"),2;
        LENGTH:=TOS-LOGICAL(@TERMLINE);
        IF PROCESS'BEING'INITIATED THEN
          BEGIN
          PRINT (LTERMLINE,-LENGTH,0);
          FERRMSG (FERROR,LBA1,LENGTH1);
          PRINT (LBA1,0,0);
          PRINT (LBA1,-LENGTH1,0);
          PRINTFILEINFO (FILE'NUMBER);
          END;
        WRITE'MSG (-92,LENGTH,TERMLINE);
        TERMINATE;
        END;
$PAGE
      <<**********************************>>
      <<** Start of main program code. **>>
      <<**********************************>>

      BEGIN
      TERM'OPEN:=FALSE;
      TRUE'COND:=TRUE;
      CHECK'F'KEYS'ONLY:=FALSE;
      PROCESS'BEING'INITIATED:=TRUE;

      MOVE TERMLINE:=("*** MONITORR *** B.01.00 Copyr. 1987, ",
                      "Benge Bruno.  All rights reserved."),2;
      LENGTH:=TOS-LOGICAL(@TERMLINE);
      PRINT (LTERMLINE,-LENGTH,0);
      DATELINE (TERMLINE);
      PRINT (LTERMLINE,-27,0);
      PRINT (LTERMLINE,0,0);

      GET'INFO (RUN'PARM, INFO'LENGTH, LINFO'STRING);

      IF RUN'PARM < 0D
        THEN DEBUG'MODE:=TRUE
        ELSE DEBUG'MODE:=FALSE;

      IF INFO'STRING = "BENGE" THEN
        BENGE'MODE:=TRUE
```

```
      ELSE
        BEGIN
        BENGE'MODE:=FALSE;

        IF INFO'LENGTH < 3 THEN
          BEGIN
          MOVE TERMLINE:="** Invalid ;INFO= length found. **",2;
          LENGTH:=TOS-LOGICAL(@TERMLINE);
          PRINT (LTERMLINE,-LENGTH,0);
          TERMINATE;
          END;

        MONITORR'RIN:=BINARY (INFO'STRING,3);
        IF <> THEN
          BEGIN
          MOVE TERMLINE:=("** Invalid RIN number of '"),2;
          MOVE *:=INFO'STRING,(3),2;
          MOVE *:=("' passed in ;INFO= string. **"),2;
          LENGTH:=TOS-LOGICAL(@TERMLINE);
          PRINT (LTERMLINE,-LENGTH,0);
          TERMINATE;
          END;
        END;

      IF DEBUG'MODE THEN
        RUN'PARM:=-RUN'PARM;
      DEV:=INTEGER(RUN'PARM);
      MOVE DEVICE:="000 ";
      ASCII (DEV,-10,DEVICE(2));

      TERMINAL:=FOPEN (,%4,%0,-1024,DEVICE);
      IF <> THEN
        FS'ERROR (1,TERMINAL,1);

      MOVE TERMLINE:="MSG ";
      MSG'FILE:=FOPEN (TERMLINE,%30105,%2303);
      IF <> THEN
        FS'ERROR (2,MSG'FILE,1);

      MOVE TERMLINE:="MSG ";
      MSGTEMP'FILE:=FOPEN (TERMLINE,%32106,%2300);
      IF <> THEN
        FS'ERROR (3,MSGTEMP'FILE,1);
$PAGE
      <<*****************************************************>>
      <<** Enable reads on files with no writers and vice versa. **>>
      <<*****************************************************>>

      L:=TRUE;
      FCONTROL (MSG'FILE, 45, L);
```

```
      IF <> THEN
        FS'ERROR (2,MSG'FILE,5);

      FCONTROL (MSGTEMP'FILE, 45, L);
      IF <> THEN
        FS'ERROR (3,MSG'FILE,5);
$PAGE
      IF NOT BENGE'MODE THEN
        BEGIN
        <<*****************************************************>>
        <<** Lock RIN 'MONITORR' to ensure that only one **>>
        <<** copy of this program is active.             **>>
        <<*****************************************************>>

        LOCK'COND.(15:1):=0;
        MOVE BA1:="MONITORR   ";

        LOCKGLORIN (MONITORR'RIN,LOCK'COND,BA1);
        IF <> THEN
          BEGIN
          IF > THEN
            BEGIN
            MOVE TERMLINE:=("MONITORR: ** Program is currently ",
                            "executing. **"),2;
            LENGTH:=TOS-LOGICAL(@TERMLINE);
            PRINT (LTERMLINE,-LENGTH,0);
            END
          ELSE
            BEGIN
            MOVE TERMLINE:=("** LOCKGLORIN intrinsic error: RIN#="),2;
            NUM'CHAR:=ASCII (MONITORR'RIN,10,BA1);
            MOVE *:=BA1,(NUM'CHAR),2;
            MOVE *:=(".  RIN password = MONITORR. **"),2;
            LENGTH:=TOS-LOGICAL(@TERMLINE);
            PRINT (LTERMLINE,-LENGTH,0);
            END;
          MOVE TERMLINE:=("MONITORR: ** This process cannot ",
                          "continue. **"),2;
          LENGTH:=TOS-LOGICAL(@TERMLINE);
          PRINT (LTERMLINE,0,0);
          PRINT (LTERMLINE,-LENGTH,0);
          TERMINATE;
          END;
        END;
$PAGE
      <<*********************************************************>>
      <<** Tell the MONITORF that this process is initiated. **>>
      <<*********************************************************>>
```

```
      LTERMLINE:=MONITORR;
      MOVE TERMLINE:=BLANKS,(2);  <<** Future msg area **>>
      FWRITE (MSG'FILE,LTERMLINE,-4,0);
      IF <> THEN
        FS'ERROR (2,MSG'FILE,4);

      MOVE TERMLINE:="PASS";
      FWRITE (MSG'FILE,LTERMLINE,-4,0);
      IF <> THEN
        FS'ERROR (2,MSG'FILE,4);

      PROCESS'BEING'INITIATED:=FALSE;
$PAGE
GET'MSG'REQUEST:

      IF DEBUG'MODE THEN
        BEGIN
        MOVE BA1:="MONITORR: Posting MSGTEMP file read.",2;
        LEN:=TOS-LOGICAL(@BA1);
        PRINT (LBA1,-LEN,0);
        END;

      IN'LENGTH:=FREAD (MSGTEMP'FILE,LINPUT'RECORD,-80);
      IF <> THEN
        FS'ERROR (3,MSGTEMP'FILE,3);

      IF DEBUG'MODE THEN
        BEGIN
        MOVE BA1:="MONITORR: MSGTEMP file read complete.",2;
        LEN:=TOS-LOGICAL(@BA1);
        PRINT (LBA1,-LEN,0);
        END;

      IF LINPUT'RECORD = -1 THEN
        BEGIN
        <<**************************************************>>
        <<** Control record from MONITORF father process.   **>>
        <<**   -1 = if TERMINAL open -> close the terminal. **>>
        <<**************************************************>>

        IF TERM'OPEN THEN
          BEGIN
          FCLOSE (TERMINAL,0,0);
          IF <> THEN
            FS'ERROR (1,TERMINAL,2);
          TERM'OPEN:=FALSE;
          END;
        GO TO GET'MSG'REQUEST;
        END;
```

```
      IF NOT TERM'OPEN THEN
        BEGIN
        <<*********************************************************>>
        <<** The terminal may have been previously closed by a **>>
        <<** previous request of -1; therefore open it!        **>>
        <<*********************************************************>>
        TERMINAL:=FOPEN (,%404,%40,1024,DEVICE);
        IF <> THEN
          FS'ERROR (1,TERMINAL,1);
        TERM'OPEN:=TRUE;
        END;

      FIRST'FS'ERROR'27'28:=FALSE;
$PAGE
GET'TERMINAL'INPUT:

      IF LINPUT'RECORD(1) = 0 THEN
        BEGIN
        <<*****************************************************>>
        <<** Passed value of READ LENGTH is zero, meaning **>>
        <<** only search for function keys.               **>>
        <<*****************************************************>>
        CHECK'F'KEYS'ONLY:=TRUE;
        READ'LENGTH:=-2;
        END
      ELSE
        BEGIN
        READ'LENGTH:=LINPUT'RECORD(1);
        CHECK'F'KEYS'ONLY:=FALSE;
        END;

      IF LINPUT'RECORD = 0 THEN
        BEGIN
        <<*************************************>>
        <<** Infinite read on terminal file. **>>
        <<*************************************>>
        L:=0;

        IF DEBUG'MODE THEN
          BEGIN
          MOVE BA1:="MONITORR: Posting infinite terminal read.",2;
          LEN:=TOS-LOGICAL(@BA1);
          PRINT (LBA1,-LEN,0);
          END;

        FCONTROL (TERMINAL,4,L);
        IF <> THEN
          FS'ERROR (1,TERMINAL,5);
```

```
    IN'LENGTH2:=FREAD (TERMINAL,LINPUT'RECORD2,READ'LENGTH);
    IF <> THEN
      BEGIN
      FCHECK (TERMINAL,FERROR);
      IF FERROR = 27 OR FERROR = 28 THEN
        BEGIN
        IF NOT FIRST'FS'ERROR'27'28 THEN
          FIRST'FS'ERROR'27'28:=TRUE
        ELSE
          BEGIN
          MOVE TERMLINE:=("** Two consecutive FS ",
                          "error 27 or 28 have ",
                          "occurred. **"),2;
          LENGTH:=TOS-LOGICAL(@TERMLINE);
          WRITE'MSG (-92,LENGTH,TERMLINE);
          TERMINATE;
          END;
        MOVE TERMLINE:="** Unexpected FS error ",2;
        NUM'CHAR:=ASCII (FERROR,10,BA1);
        MOVE *:=BA1,(NUM'CHAR),2;
        MOVE *:=(" has occurred on TERMINAL ",
                 "during FREAD. **"),2;
        LENGTH:=TOS-LOGICAL(@TERMLINE);
        WRITE'MSG (-91,LENGTH,TERMLINE);
        GO TO GET'TERMINAL'INPUT;
        END
      ELSE
        FS'ERROR (1,TERMINAL,3);
      END;
    END
  ELSE
    BEGIN
    <<**************************************>>
    <<** Enable timed read on terminal. **>>
    <<**************************************>>

    IF DEBUG'MODE THEN
      BEGIN
      MOVE BA1:="MONITORR: Posting timed terminal read.",2;
      LEN:=TOS-LOGICAL(@BA1);
      PRINT (LBA1,-LEN,0);
      END;

    L:=LINPUT'RECORD;
    FCONTROL (TERMINAL,4,L);
    IF <> THEN
      FS'ERROR (1,TERMINAL,5);

    IN'LENGTH2:=FREAD (TERMINAL,LINPUT'RECORD2,READ'LENGTH);
    IF <> THEN
```

```
      BEGIN
      FCHECK (TERMINAL,FERROR);
      IF FERROR = 27 OR FERROR = 28 THEN
        BEGIN
        IF NOT FIRST'FS'ERROR'27'28 THEN
          FIRST'FS'ERROR'27'28:=TRUE
        ELSE
          BEGIN
          MOVE TERMLINE:=("** Two consecutive FS ",
                          "error 27 or 28 have ",
                          "occurred. **"),2;
          LENGTH:=TOS-LOGICAL(@TERMLINE);
          WRITE'MSG (-92,LENGTH,TERMLINE);
          TERMINATE;
          END;
        MOVE TERMLINE:="** Unexpected FS error ",2;
        NUM'CHAR:=ASCII (FERROR,10,BA1);
        MOVE *:=BA1,(NUM'CHAR),2;
        MOVE *:=(" has occurred on TERMINAL ",
                 "during FREAD. **"),2;
        LENGTH:=TOS-LOGICAL(@TERMLINE);
        WRITE'MSG (-91,LENGTH,TERMLINE);
        GO TO GET'TERMINAL'INPUT;
        END
      ELSE
        BEGIN
        IF FERROR <> 22 THEN
          FS'ERROR (1,TERMINAL,3);
        END;

      <<*****************************************>>
      <<** Inform MONITORF that timeout occurred.**>>
      <<*****************************************>>

      IF DEBUG'MODE THEN
        BEGIN
        MOVE BA1:="MONITORR: Terminal read has timed out.",2;
        LEN:=TOS-LOGICAL(@BA1);
        PRINT (LBA1,-LEN,0);
        END;

      WRITE'MSG (-3,0,BLANKS);
      GO TO GET'MSG'REQUEST;
      END;
    END;

  FIRST'FS'ERROR'27'28:=FALSE;

  IF DEBUG'MODE THEN
    BEGIN
```

```
        MOVE BA1:="MONITORR: Terminal read has completed.",2;
        LEN:=TOS-LOGICAL(@BA1);
        PRINT (LBA1,-LEN,0);
        END;

      IF IN'LENGTH2 = 0 THEN
        BEGIN
        <<*************************>>
        <<** Return key pressed. **>>
        <<*************************>>
        WRITE'MSG (-1,0,BLANKS);
        END
      ELSE
        BEGIN
        <<*************************************>>
        <<** Check for function keys or data. **>>
        <<*************************************>>
        IF INPUT'RECORD2 = ESC THEN
          BEGIN
          I:=INTEGER(INPUT'RECORD2(1));
          IF I > %157 AND I < %170 THEN
            BEGIN
            I:=I-%160+1;
            WRITE'MSG (-1,I,BLANKS);
            END
          ELSE
            BEGIN
            <<*********************************>>
            <<** Strange stuff; read again. **>>
            <<*********************************>>
            GO TO GET'TERMINAL'INPUT;
            END;
          END
        ELSE
          BEGIN
          <<*****************************************>>
          <<** Data sent in terminal; send length. **>>
          <<*****************************************>>
          IF NOT CHECK'F'KEYS'ONLY
            THEN WRITE'MSG (-2,IN'LENGTH2,INPUT'RECORD2)
            ELSE GO TO GET'TERMINAL'INPUT;
          END;
        END;

      GO TO GET'MSG'REQUEST;
$PAGE
END'OF'PROGRAM:

      IF TERMINAL <> 0 THEN
        BEGIN
```

```
      FCLOSE (TERMINAL,0,0);
      IF <> THEN
        FS'ERROR (1,TERMINAL,2);
      END;

    IF MSG'FILE <> 0 THEN
      BEGIN
      FCLOSE (MSG'FILE,0,0);
      IF <> THEN
        FS'ERROR (2,MSG'FILE,2);
      END;

    IF MSGTEMP'FILE <> 0 THEN
      BEGIN
      FCLOSE (MSGTEMP'FILE,0,0);
      IF <> THEN
        FS'ERROR (3,MSGTEMP'FILE,2);
      END;

    END;
$PAGE
END.
```

Summary

Local and global RINs may be used effectively for the monitor and control of processes within an HP application. If RINs are used in conjunction with the MPE message file facility, complete process control of initiation, termination, and automated recovery techniques can be developed. In fact, a very nice real time terminal emulator may even be developed!

Global RINs can now be displayed quite easily with the SHOWRIN program. It is my hope that this program and/or a similar capability be incorporated into the HP Online Performance Tool (OPT/3000).

Acknowledgements

I wish to advise the reader to read the paper entitled 'Secrets of System Tables Revealed' by Eugene Volokh of VESOFT which was presented at the International Users Group Meeting of 1985 in Washington, D.C. Eugene describes several nice techniques in order for one to read through MPE system tables with privileged mode. I found this to be very helpful as I plowed through the PCB, System RIN, user stack, etc. to develop the SHOWRIN program and procedure.

I want to thank My Phan of the Information Networks Division of Hewlett-Packard for helping me with the ATTACHIO procedure calls made in the FWRITE procedure.

I also want to thank Dave Martin of the Cupertino Sales Center of Hewlett-Packard who spent several sleepless months together as we wrote a whole bunch of code that used this terminal monitor and other nifty things. Sometimes there is just nothing that can compare to the freedom of developing a complete large application with a no holds barred concept!

Biography

Benedict G. Bruno has been working with the HP 3000 family of computer systems for seven years. His experience includes working for Hewlett-Packard Company as a Systems Engineer in the Los Angeles Airport Sales Office, a Network Consultant for Information Networks Division in Cupertino, and a Senior Applications Systems Engineer in the Rockville Sales Office. The integration of local and global RINs, message files, SPL, and HP data communications products for several application systems in retail, electronic mail, and others have required his need to develop the SHOWRIN program, terminal monitors, etc. He is currently president and cofounder of S.T.R. Software Company where he has developed POS/3000 to provide unattended data communications application software.

INTEGRATED OFFICE SOLUTIONS
HOW TO GET THE MOST FROM YOUR WORKSTATION

Brenda Buchwitz
Hewlett-Packard
3410 Central Express Way
Santa Clara, California

The purpose of this paper is to educate professionals and assistants on how they can increase their productivity by using integrated office solutions.

An emphasis will be placed on Vectra Office solutions in the office environment. This would include scenerios on how to integrate Graphics Gallery, Executive MemoMaker, AdvanceWrite, Lotus 1-2-3, AdvanceLink, AdvanceMail and Executive Card Manager. Scenerios will also illustrate how to integrate the workstation with the HP3000 as well as the advantages of migrating to the workstation.

The discussion will stress the increase in productivity, communication and quality materials that will be derived from the use of your workstation.

SLIDE 1 Emergence of Integrated Business Systems

The business systems marketplace is evolving. During this evolution we have become more of an information driven society. The nation is becoming more comprised of knowledge workers and information handlers than factory workers. The service industry and information processing business employs three times as many people as the manufacturing area. Even factory workers are seeing terminals on the factory floor.

Computer processing costs have decreased 30-40% over the last 20 years. In the 60's corporate processing was very centralized. In the 70's the industry began to see more distributed systems and now in the 80's stand-alone computers and personal computers are becoming widely used.

As a result of these trends, organizations have begun to utilize information to make strategic management decisions. Users have begun to demand access to more information and tools to effectively use information in their jobs. They are more interested in attaining more computing power to manipulate and analyze large volumes of data. Users also wish to use computers to communicate with each other and automate and streamline repetitive tasks.

These factors are driving the convergence of what were once four discrete technologies; data processing, personal computing, office

automation and communications/networking. This has caused the emergence of a new market, Integrated Business Systems.

SLIDE 2 HP Believes That...

HP believes that personal computers are the driving force in computing today. Increased productivity starts with the individual. The user acceptance of personal computers is growing. Between 1980 and 1985 the installed base of PCs grew from 400,000 to 8.9 million. It continues to grow daily as workers utilize them to increase their efficiency. PC's offer higher performance and flexibility and they reduce the overall system costs. PC users have the opportunity to use thousands of applications on powerful systems. They are able to communicate with other PC users via local area networks and department computers. They also have controlled access of the data available for use with these applications.

PC users benefit from high performance computers that don't require the maintenance of a larger system. They are also very flexible for the individual who wishes to manage the manipulation of their own data.

More users can be supported by a host system when personal computers are used instead of terminals. Therefore, additional computing power can be added less expensively by adding PCs vs purchasing another host computer.

SLIDE 3 Workgroups are the Focus

Workgroups are the natural focus for office productivity. Studies by the Stanford Research Institute support the belief that the management of information and communications within the immediate workgroup will increase productivity.

SLIDE 4 Why Emphasize Workgroups?

Two-thirds of the average worker's time is spent on interpersonal office communications. The majority of communication is with the immediate workgroup or in the same building. This includes face-to-face and written communications.

Written communication accounts for about 20% of their time and face-to-face represents about 32% of the time spent on the job.

SLIDE 5 Personal Productivity Center

Office Automation provides the user interface to data processing. Office Automation tools such as electronic mail, decision support, wordprocessing and graphics enable the office worker to effectively use company information. Productivity increases by tailoring these tools to meet the individual's needs.

Hewlett-Packard has leveraged its' strength in the four component technologies, to provide an integrated business system that incorporates the strengths of personal computing and distributed data processing into an office system.

Within Data Processing HP has made the commitment to provide compatible, upgradeable solutions across the HP3000 family. In the Networking area HP has provided extensive peer-to-peer communications across the entire HP3000 family. Office Automation offers an extension of our technology to a large number of users. Tools are available to the professional, managers and secretaries in the areas of decision support, graphics and document processing. Today we have over 125,000 users of our electronic mail system.

SLIDE 6 Most Workgroups Share Common Activities

Hewlett-Packard has a lot to offer in the workgroup environment. Most workgroups share common activities. These activities fall into the areas of collecting information, manipulating information, communicating results and managing shared resources. Everyone performs these functions on a daily basis. By improving efficiency in these areas productivity will increase.

SLIDE 7 The PPC Integrates Workgroup Activities into your Organizational Network

Integrating workgroup activities into your organization is done by using tools that are available on the HP3000 and the personal computer. By sharing data from your corporate or department computer it becomes easier to collect and communicate information. Downloading this information on the PC enables the users to manipulate the information to attain the desired results.

Managing shared resources, like printers and plotters, within your organization leverages the hardware investment. Entire workgroups can share a Laser printer, plotter or disc drives.

SLIDE 8 Collecting Information Solution

HP Access provides direct access to the information stored on the personal computer, departmental computer or other databases on the network. Information can be easily extracted from the HP3000 IMAGE database and downloaded onto the PC for manipulation in a local database or spreadsheet. HPAccess makes it very easy to move data between an IMAGE database, R:base, dBase II, Condor, Executive Card Manager, Lotus 1-2-3 and
Gallery.

SLIDE 9 Collecting Information: Information is Accessible in Usable Form

Users can create their own professional reports, Lotus spreadsheets, graphics, mailing lists and text documents. Time is saved by not having to re-key the data into other applications for reporting and analysis. MIS saves programming time. Repetitive queries can be easily automated.

SLIDE 10 Manipulating Information Solution

A comprehensive series of integrated personal applications are available from Hewlett-Packard. We offer wordprocessing, business graphics, data management, spreadsheets and communications packages.

Executive MemoMaker, EMM, and Advancewrite are the strategic wordprocessing packages. Executive MemoMaker is very simple to use. It was designed for managers and casual users that don't need highly structured documents. It offers an easy way to create, edit, format and print documents. EMM also comes with a Spell Checker.

Advancewrite was designed for professional secretaries. It provides advanced wordprocessing functions such as index and table of contents generation, automatic footnoting and pagination. Spreadsheet functions can also be performed within the document.

EMM and Advancewrite eliminate the need to retype documents to correct mistakes and replace updated text. All documents can be prepared for electronic distribution.

The Gallery Collection provides a very comprehensive solution to business graphics needs. It enables managers, professionals and secretaries to easily produce quality reports and visual aids. High quality charts and graphs can be produced. Over 300 pictures and symbols are available in Gallery Portfolios. They can be easily added to your graphs. All of the slides in this document were created with Gallery.

Scanning Gallery offers the capability of scanning images with a Scanjet and including them in a desk top publishing document like Pagemaker.

Gallery files can be copied into Executive MemoMaker and Pagemaker. HPGraphics Curator/3000 enables you to convert Gallery files so that they can be used by TDP, HPDraw and Word/3000.

Data Management solutions are provided by Executive Card Manager and R:Base.

ECM provides create, edit, format and printing capabilities. It is easy to use. Card files and lists can be maintained and accessed quickly. Phone numbers can be dialed automatically and form letters

and mailing labels can be generated easily.

R:Base is a full function data base system. It provides relational data base capabilities, built-in functions, and application generators. R:Base can import information from Lotus 1-2-3 and dBase.

Lotus 1-2-3 increases the productivity of anyone who works with numbers. People who analyze financial information can create, edit, format and print financial reports. Automatic graph generation can also be accomplished. Boiler plates can be developed for common financial forms or reports. This eliminates the manual process involved with recalculating financial reports. Sensitive data can easily be removed when reporting summary data. Lotus spreadsheets can be added to EMM or Advancewrite documents to create complete reports.

AdvanceLink provides communications capabilities to the HP3000, other PCs, VAX and IBM computers. A powerful command language is provided to automate connections and the passing of data between the host and the PC. AdvanceLink has an easy to use menu driven interface to transfer files between the host and PC.

Charts and slides can be mailed through AdvanceLink to HPDesk. They can also be read from HPDesk if you use HPGraphics Curator/3000.

AdvanceLink provides HP terminal emulation. It is the basis of integration between the PC and the HP3000.

AdvanceMail is a communication product for anyone who wants to have an HPDesk function on their PC. It provides an In Tray/Out Tray capability, message editor to prepare mail and a message filtering to prioritize communications. The user can create, edit and reply to messages while not connected to the HP3000. This saves connect charges and unloads this processing off the HP3000 and onto the PC.

SLIDE 11 Communicating Results: Challenge

Teamwork can be improved by communicating more efficiently. In many companies telephone tag hampers internal communications, express delivery charges for documents is expensive and spreadsheet and graphics data must be mailed on diskette or rekeyed.

SLIDE 12 Communicating Results: Solution

An HP Mail Network, HPDESK can act as an information distributor to manage important information. DeskManager connects to data processing, other systems and external mail systems. It also integrates with PCs via AdvanceLink and AdvanceMail. HPDesk provides a tool to communicate with anyone in your organization. At Hewlett-Packard we use HPDesk daily to communicate worldwide. Information is passed easily and efficiently. Re-entry of graphic, spreadsheet and text information is no longer a time consuming process. Such documents can be mailed easily to other workers.

SLIDE 13 Resource Sharing

Personal computers are proliferating throughout organizations. Users demand to access high-quality peripherals. Backup protection is needed by PC users. Resource Sharing offers a solution to provide PC users the ability to leverage off peripherals that are on the network. A few good printers will go a long way. Disc sharing allows you to centralize and share data that can be backed up by MIS.

Here are a few examples of how you can save time in your organization by using an integrated solution;

SLIDE 14 Integrated Text and Graphics

Let's say that you wanted to prepare a monthly financial report for your department. Some of the data is stored in an inventory database on the HP3000. You would like to take this information analyze it and do some future projections. Then you would like to write up your results.

How could you easily do this? First you could use HPAccess to extract the data you need off the HP3000. The data can be saved on your PC as a Lotus 1-2-3 spreadsheet.

In Lotus 1-2-3 you would simply load the spreadsheet. There you could develop formulas for summaries and other targeting calculations. The Lotus file could be saved and then read into Charting Gallery. Charts and graphs could be enhanced to produce high quality presentational materials without re-entering the data. Charting Gallery files can easily be loaded into Drawing Gallery where you can add symbols and pictures.

Last you would use Executive Memomaker to load in the Lotus spreadsheet and the graphs you produced. The text of the report could then be written to produce a complete report.

Let's look a little more closely at the type of files these applications read. This is the basis for integrating at the workstation level. Below are the types of file formats that each application reads and creates.

SLIDE 15 Gallery Integration

Gallery integrates easily with Lotus 1-2-3, HP Access, EMM and DTP packages like Pagemaker. Charting Gallery can load data in ASCII or DIF format and pre-made graphs from Lotus 1-2-3 or Symphony worksheets. Charting Gallery can also read HPAccess ASCII files.

GAL files are created by Charting Gallery. They are vector graphics files compatible with Drawing Gallery, Executive MemoMaker and HP3000 Graphics via Curator.

Drawing Gallery can load and save GAL files. It can save TIFF and PC Paintbrush formats, both raster graphic files, for Desk Top Publishing integration. Pagemaker reads TIFF files and PC Paintbrush files. Ventura reads PC Paintbrush files.

SLIDE 16 Executive MemoMaker Integration

Executive MemoMaker, EMM, supports "Wordstar document" and ASCII file types. EMM can take text from Wordstar or any ASCII file, and include Graphics Gallery ".GAL" pictures. It can also save text in ASCII or "Wordstar document" formats.

SLIDE 17 AdvanceWrite Integration

AdvanceWrite can share documents and data with most wordprocessors, spreadsheets and database applications via DCA, ASCII and DIF formats.

SLIDE 18 Executive CardManager Integration

Executive CardManager, ECM, supports DIF and ASCII file types. HPAccess can save applications in ECM card file format, thus allowing tight integration with many applications.

SLIDE 19 AdvanceMail Integration

AdvanceMail supports DCA and ASCII formats. It will pass text from AdvanceWrite or EMM to HPDesk. AdvanceMail will also take HPDesk text and pass it to EMM or AdvanceWrite.

Documents can therefore be created on the PC and mailed through the 3000 network for other users to read and modify.

SLIDE 20 PPC Integrates the Information Activities of your Workgroup

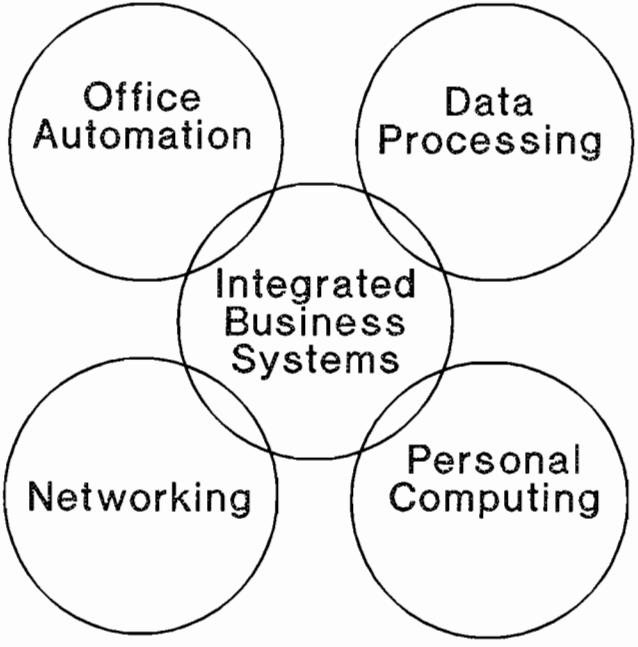
Better decisions are made through better access to information. Higher quality output is achieved through easier manipulation of information. Effective communication expands teamwork. Managing resources will provide a better return on your investment.

SLIDE 21 HP Has Integrated 55,000 Internal Users

Hewlett-Packard has integrated 55,000 internal users into a company-wide network. HPDesk is used over the network to pass information on a 24 hour basis. Marketing and development groups utilize this system on a daily basis to save time and money for Hewlett-Packard.

APPENDIX OF SLIDES

EMERGENCE OF INTEGRATED BUSINESS SYSTEMS



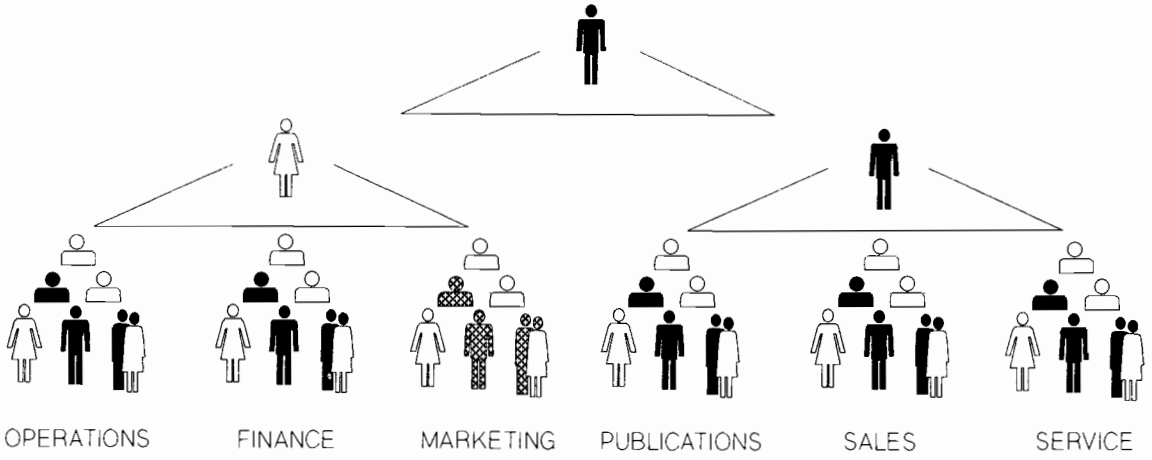
User needs are forcing the convergence of four major technologies

HP BELIEVES THAT . . .

- Personal computers are the driving force
- Workgroups are the focus for increased office productivity
- "Office Automation" provides the user interface to data processing
- Productivity increases by effectively implementing usable technology

WORKGROUPS ARE THE FOCUS

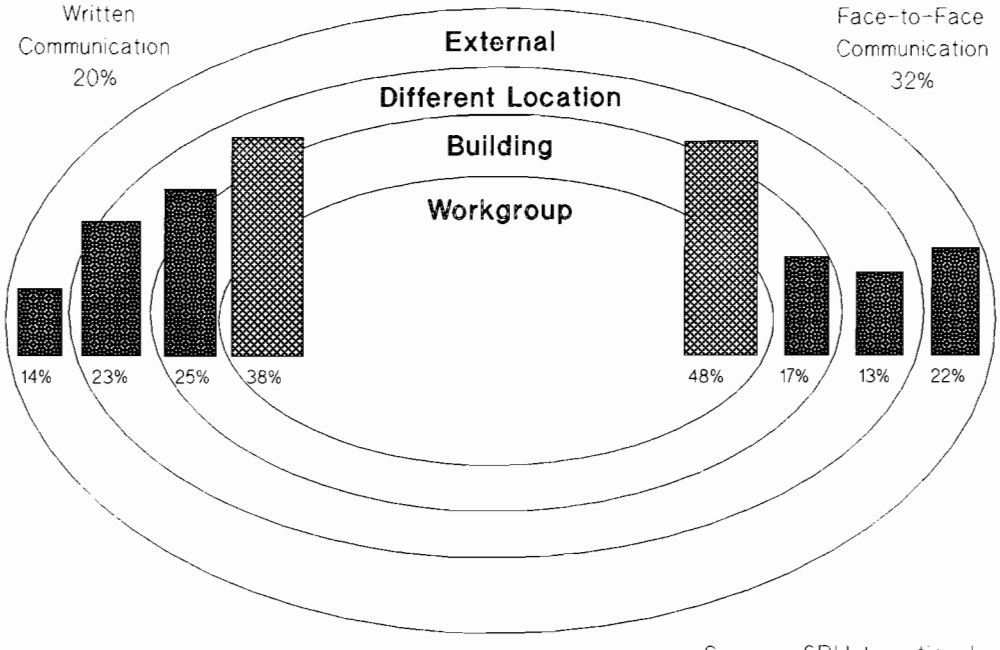
For Increased Organizational Effectiveness



Each Workgroup contributes to Organization Success

WHY EMPHASIZE WORKGROUPS?

Office communication centers on the Workgroup

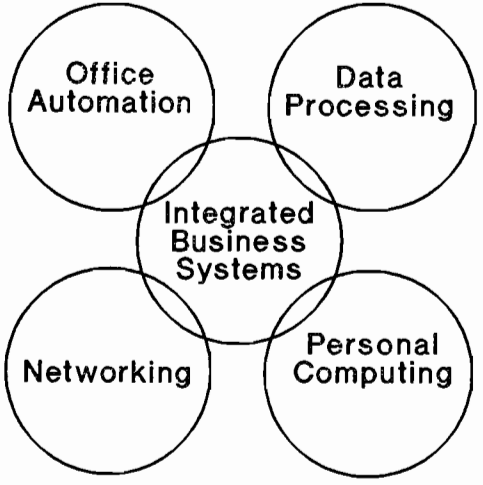


Source: SRI International

PERSONAL PRODUCTIVITY CENTER

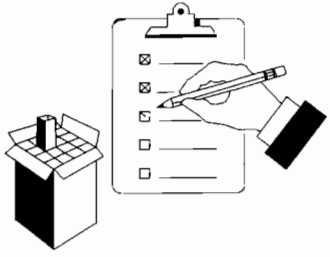
Hewlett Packard's Integrated Business System

- Sharing high-quality resources and information at the personal, workgroup, department and corporate levels
- Tools to make people and organizations more effective



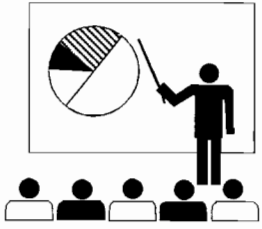
MOST WORKGROUPS SHARE COMMON ACTIVITIES

Collecting Information



Manipulating Information

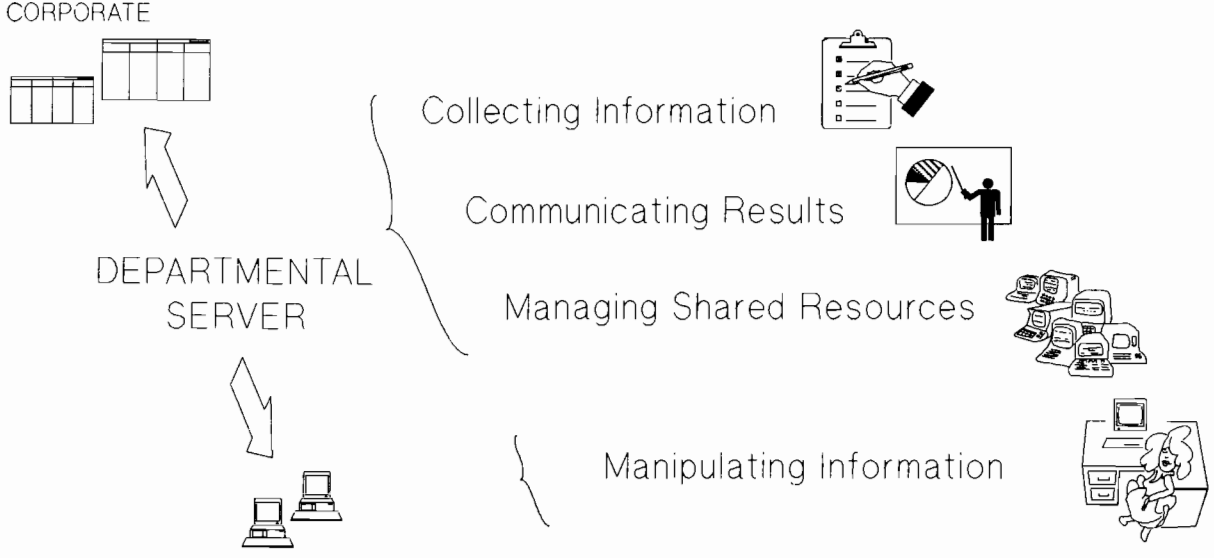
Communicating Results



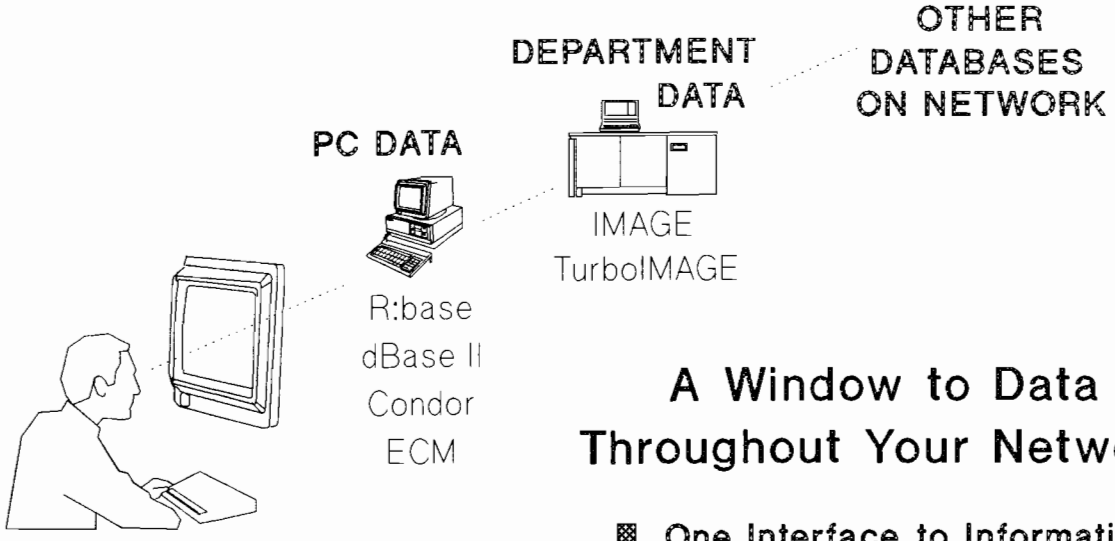
Managing Shared Resources

THE PERSONAL PRODUCTIVITY CENTER

Integrates Workgroup Activities into Your Organizational Network



SOLUTION: Provide your Users Direct Access to Information



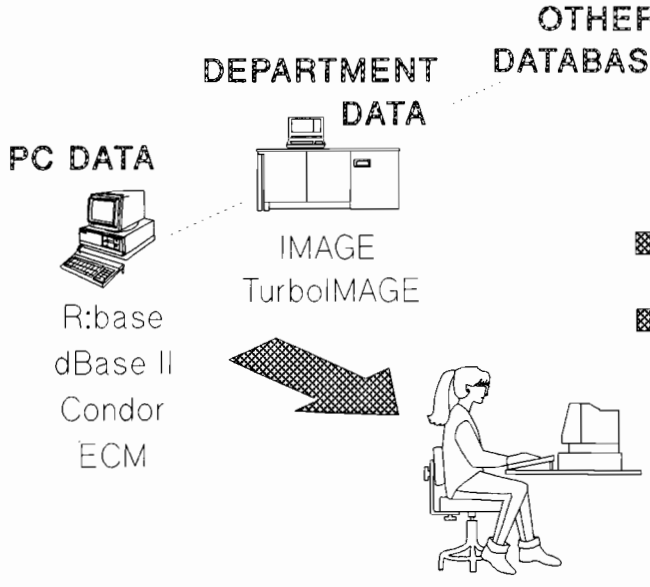
A Window to Data Throughout Your Network

- One Interface to Information
- Comprehensive Security

INFORMATION IS ACCESSABLE IN USABLE FORM

Users can Create their own:

- Professional Reports
- Lotus Spreadsheets
- Graphics
- Mailing Lists
- Text Documents

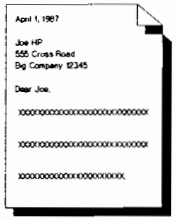


- Users save Re-Keying Data
- MIS saves Programmer Time

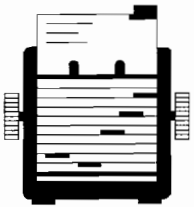
Repetitive Queries
can be automated.

SOLUTION: A Comprehensive Series of integrated Personal Applications

Word Processing: Executive MemoMaker
AdvanceWrite



Data Management: Executive Card Manager



Communications: AdvanceLink
AdvanceMail

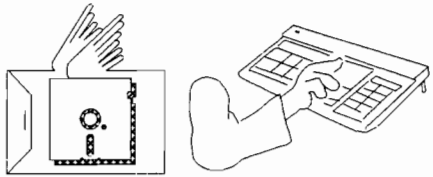
Business Graphics: Graphics Gallery

Spreadsheet: Lotus 1-2-3

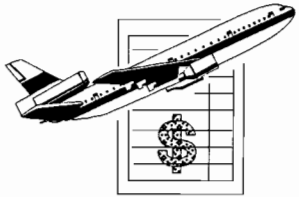
CHALLENGE: How can We Improve Teamwork by Communicating more Efficiently?

Telephone Tag hampers internal communications

Spreadsheets and graphics must be mailed on diskette or rekeyed

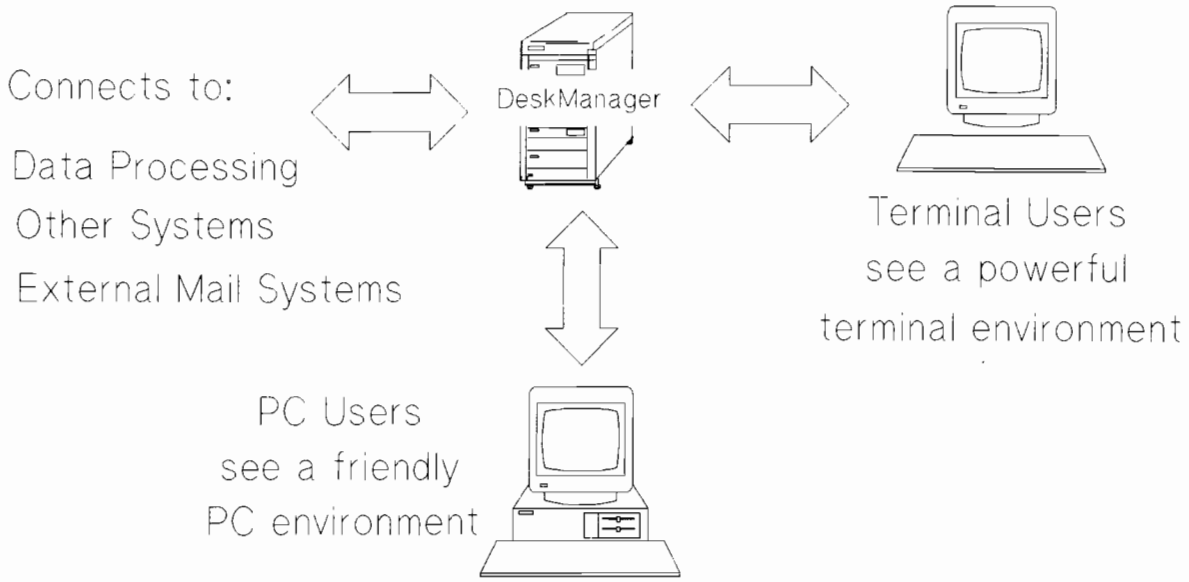


Express Delivery charges consuming larger portion of budget.



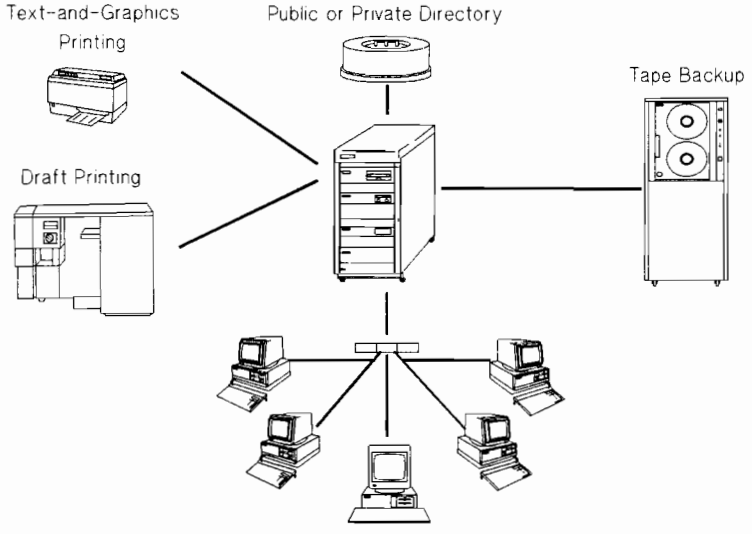
Daily data processing reports arrive late and consume paper & space.

SOLUTION: An Information Distributor to manage your Mail Network

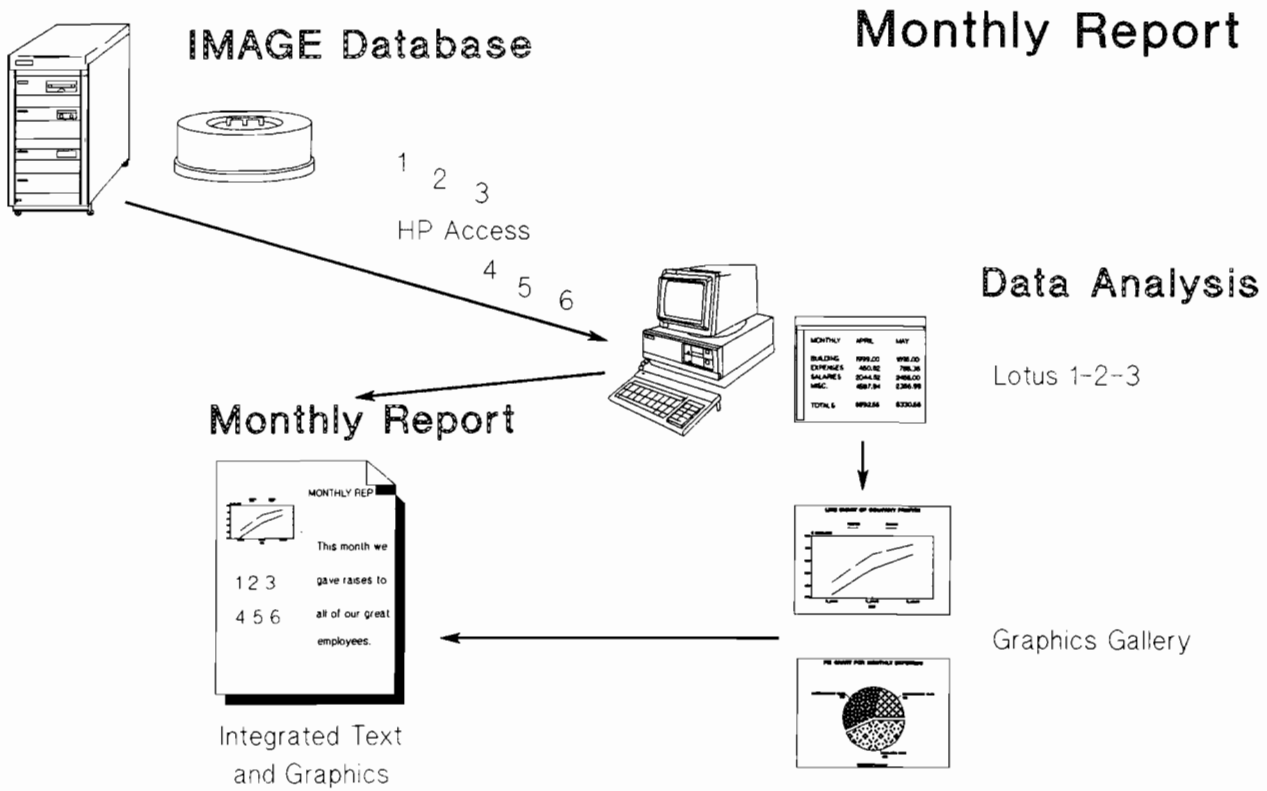


RESOURCE SHARING

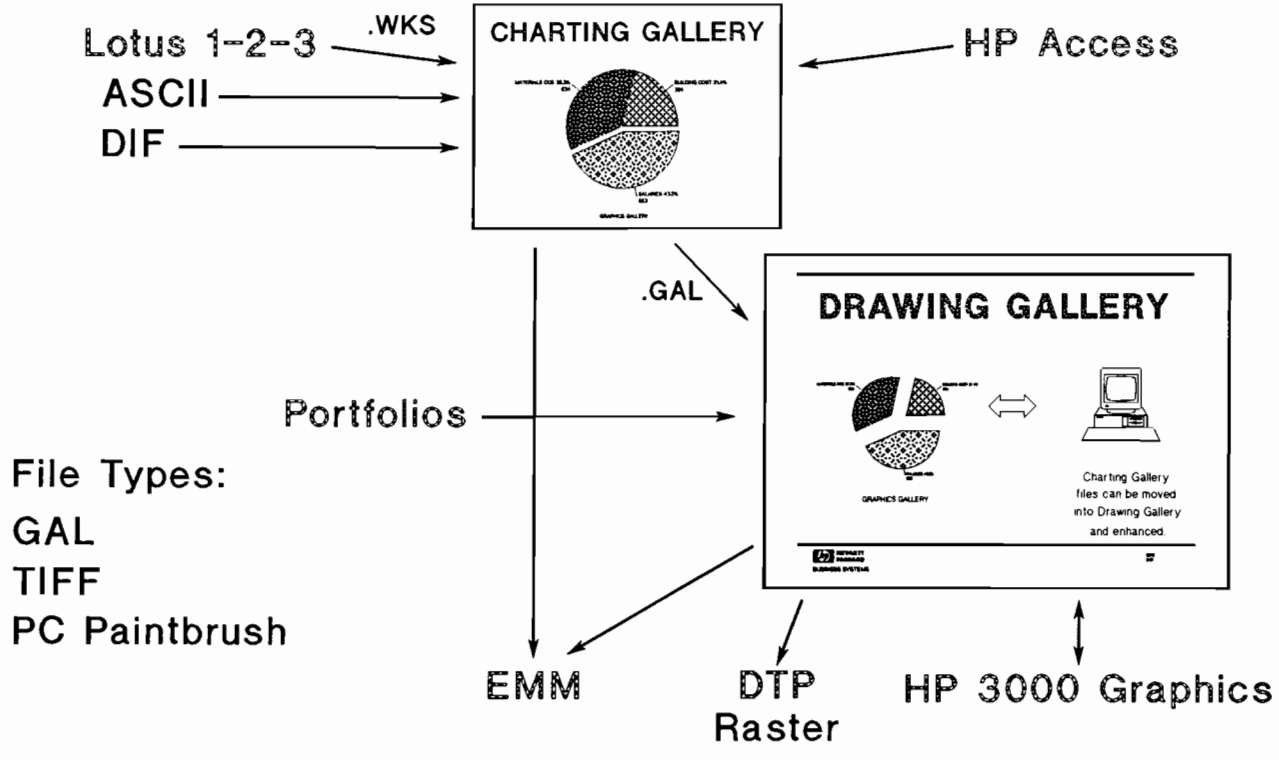
- Provides PC Users Access to Shared Resources
- Allows MIS to Better Manage Peripheral Investments



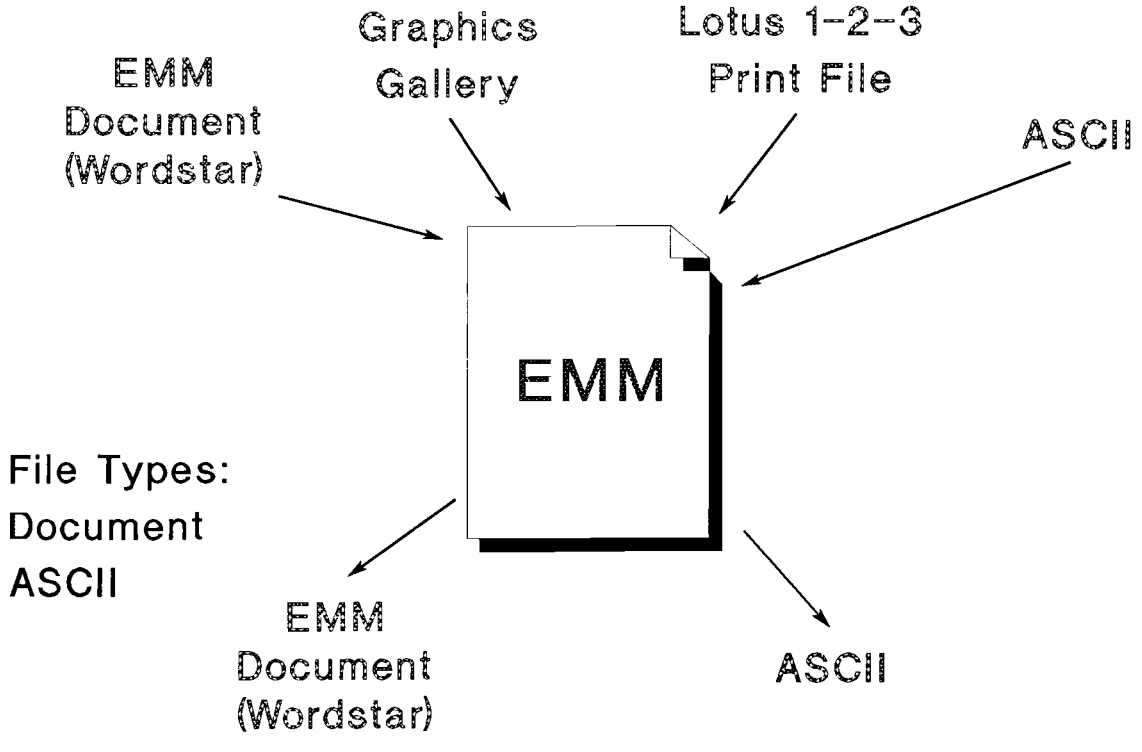
INTEGRATED TEXT AND GRAPHICS SOLUTION



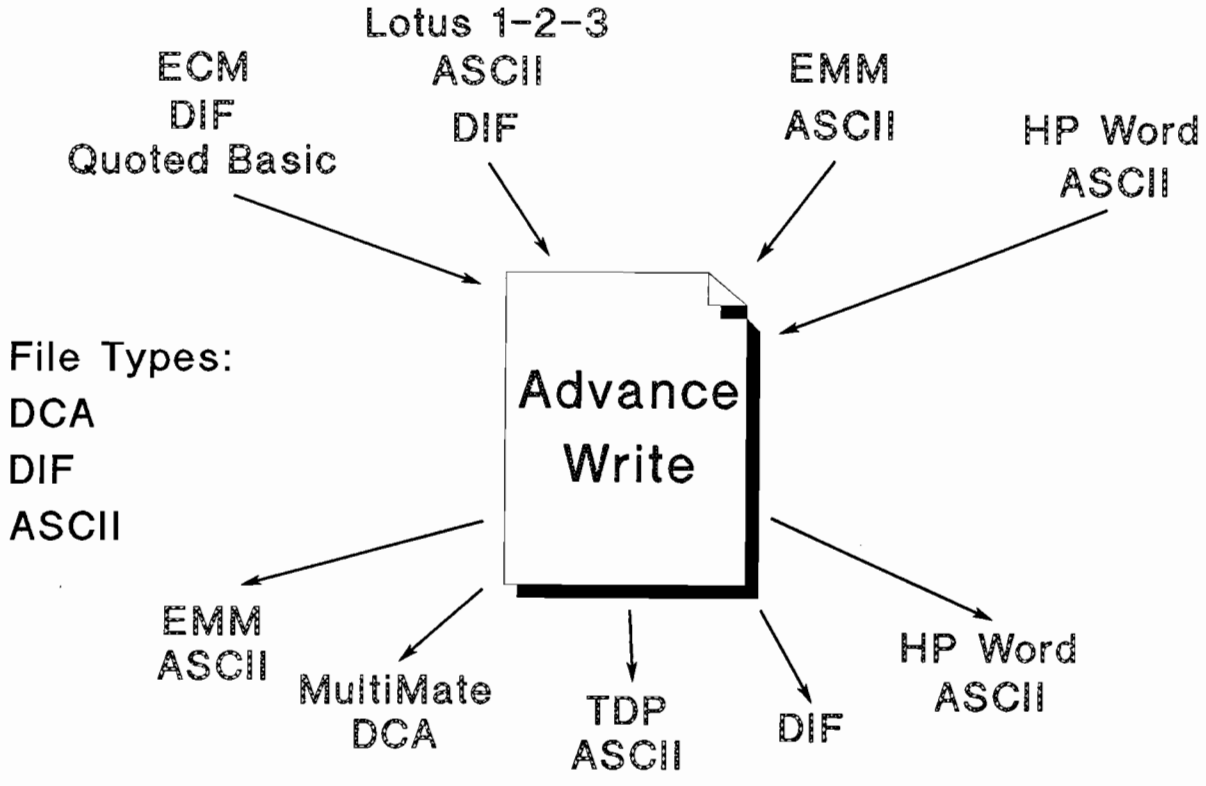
GALLERY INTEGRATION



EXECUTIVE MEMOMAKER INTEGRATION

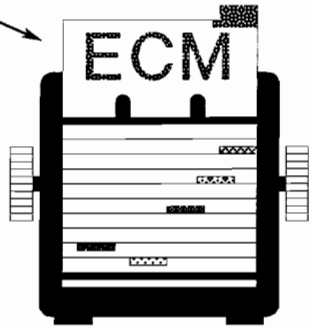


ADVANCEWRITE INTEGRATION



EXECUTIVE CARD MANAGER INTEGRATION

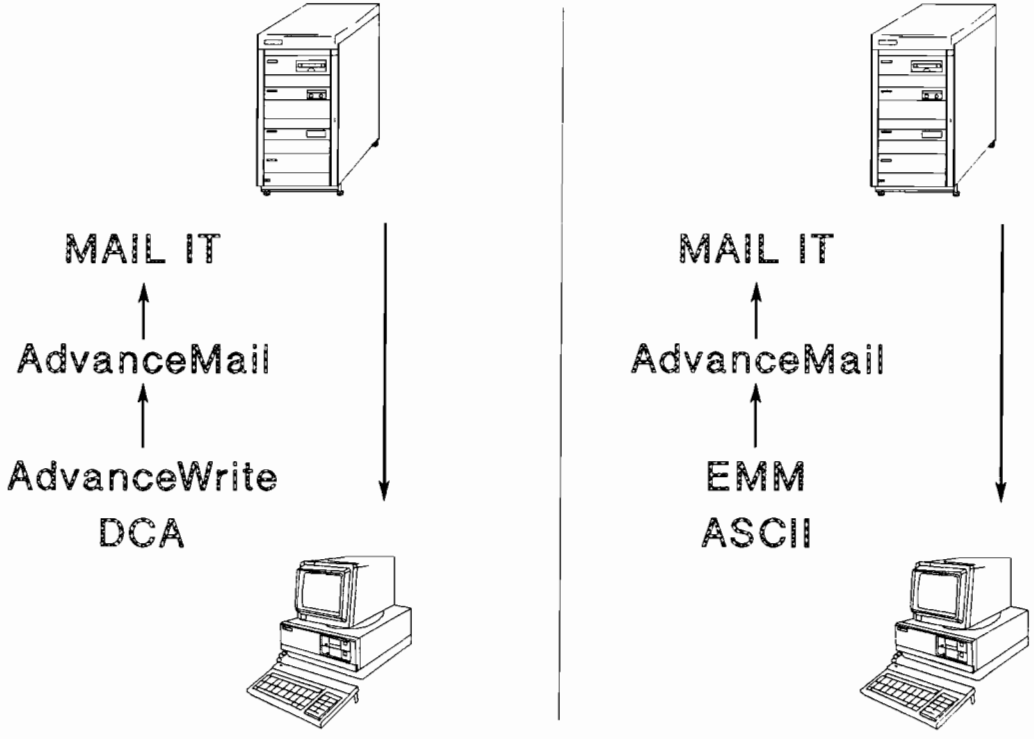
HP Access



File Types:
DIF
ASCII

AdvanceWrite
ASCII
DIF

ADVANCEMAIL INTEGRATION



THE PERSONAL PRODUCTIVITY CENTER

Integrates the Information Activities of your Workgroup

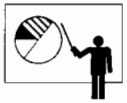
Collecting Information

Better Decisions through
better Access to Information.

Manipulating Information

Higher-Quality Output through
easier manipulation of information.

Communicating Results



Expanded Teamwork through
more Effective Communication.

Managing Shared Resources

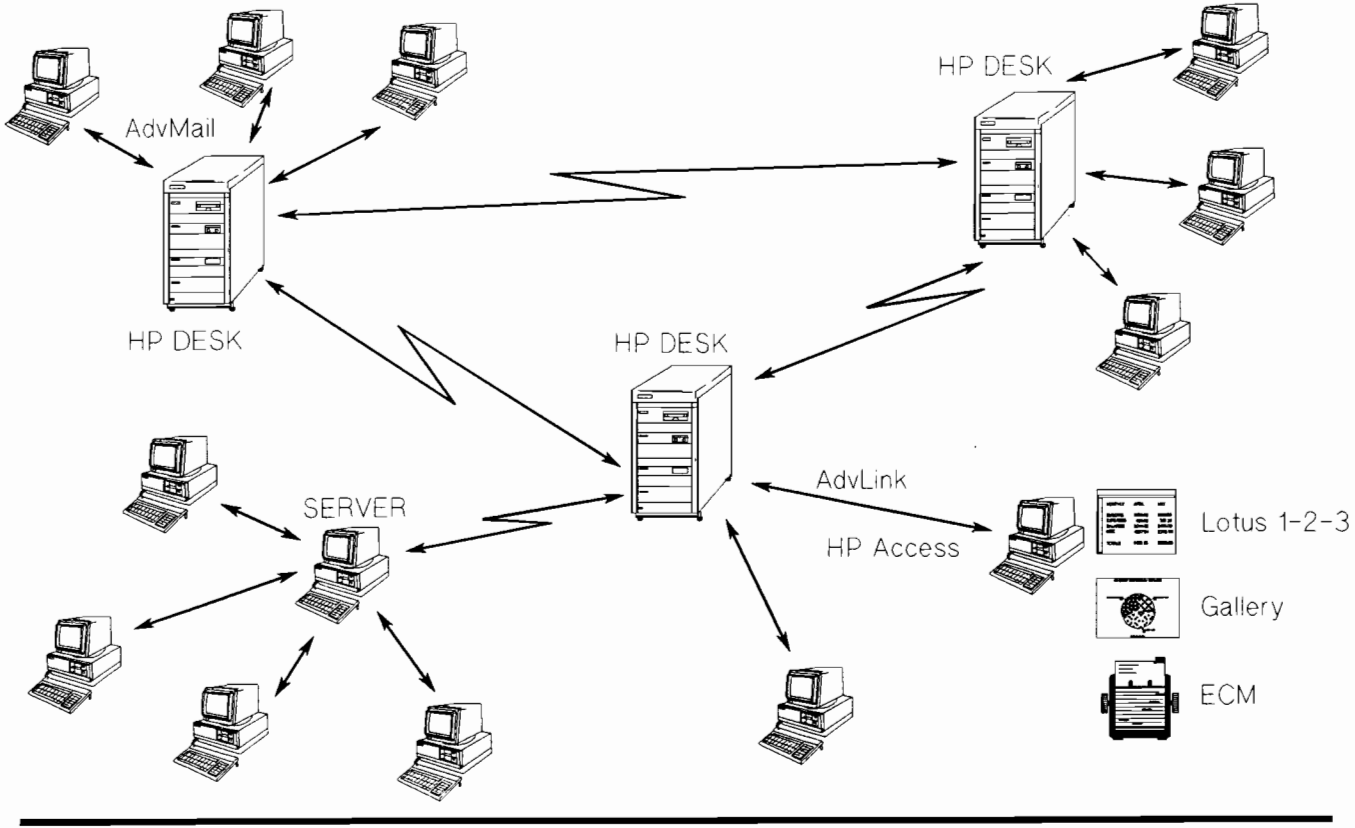
Increased Return on your
Resource Investment.

HP HAS INTEGRATED 55,000 INTERNAL USERS

into a Company-Wide Network

Marketing and Development Projects are Coordinated on a World-Wide, 24-hour Basis.

INTEGRATING THE WORKGROUP



"LOOK MA, NO WIRES"--THE USE OF RADIO CONTROLLED TERMINALS IN A HP3000 WAREHOUSING APPLICATION

W. Michael Burroughs
Jeffery J. McKay
Richard C. Kellough

Macro Systems, Inc.
8630 Fenton Street
Silver Spring, MD 20910

"LOOK MA, NO WIRES"--THE USE OF RADIO CONTROLLED TERMINALS IN A HP3000 WAREHOUSING APPLICATION

INTRODUCTION

This paper describes a development project which used radio frequency (R/F) technology to provide 2-way wireless digital communication terminals in an industrial environment. These devices, interfaced to an HP3000, are used to control and manage all operations in a very large distribution warehouse.

This paper will describe:

- The application driving the need
- The need for an R/F solution
- The R/F hardware
- "The Solution"
- Operational Impact
- Other Applications

This development project produced an integrated solution that became known as WCS/3000.

I. TYPICAL WAREHOUSE

To introduce our solution, we first need to establish the problem--a generic warehouse; from its basic functions and physical environment, to its need for information access.

At a bare minimum, a warehouse must be able to receive material, put it away in storage, and be able to retrieve it in a timely fashion. A manufacturer must be capable of receiving and storing both raw materials for the production line, and finished goods prior to shipping. A nonmanufacturer receives from suppliers and then stores materials in the role as a distributor between the producers and end users. Common to both are the needs to record the receipt, validate the receipt against an outstanding purchase order or production schedule, and update the inventory. There may also need to be an inspection process in which received material may be quarantined until approved for distribution.

Storage of material needs to be done to best optimize the use of all available space. This can take the form of fitting specific high-volume material into easily accessed places or matching the size of the material to be stored to the size of the location. In either case, that transaction needs to be recorded to provide for subsequent retrieval. It is also desirable to keep track of multiple locations of an item by lot number or age.

The last basic function, retrieval of specific stock, uses the information stored during the first two functions. A warehouse should be able to retrieve specific material quickly and easily. It should also properly rotate stock in a first in, first out (fifo) scheme to prevent old merchandise from accumulating in the warehouse. In the event a retrieval cannot be made, it should be easy for a worker to communicate the problem to the supervisor for resolution.

All of these functions need to be performed in a somewhat hostile environment. Generally, temperatures will swing with the seasons, hot and humid in the summer and cold and dry in the winter. Some facilities require refrigeration in which case it will be continuously cold and dry. Air quality may range from dusty to filthy dirty, and the cleaning process may require force sufficient enough to preclude any normal automation equipment from surviving. The equipment needs to be rugged enough to withstand unintentional abuse (such as dropping), but provide portability for ease of use by all. The equipment should also be easy to use, since the typical workforce is not familiar with office procedure or equipment.

The accumulated information must be stored and shared between all the workforce to provide for a smooth

operation. In a very small warehouse workers may often try to remember the location of stock. But human memories are inaccurate and it may be difficult to communicate stock availability to sales people and order clerks. To improve accuracy, this information can be stored on paper in the form of purchase orders, receiving forms, inventory storage records, and pick lists. Although these are easy to create and use, they can become illegible or lost. Even if used successfully, they require large areas for filing and elaborate systems for retrieval of a particular item. Due to this volume of data, the use of a computer was inevitable. Like most early data processing applications, the automation of the warehouse was driven by the need for financial accounting.

The typical processing cycle involves data entry of the work performed during the prior day, allowing for possible transcription errors to enter the system. After entering this information, a batch updating procedure has to be executed to put the new data into permanent files. This entire process could create a lag time of 3 to 5 days before an event can impact operations. Subsequent events have already occurred, and available information is never up-to-date.

Inventory counts are usually kept in total, by item, to allow for the valuing of the inventory. Counts by individual location are not maintained. Hence, picking instructions produced against the last batch update will always contain invalid or out-of-date information.

II. THE NEED FOR AN RF (RADIO FREQUENCY) SOLUTION

As we have seen, the key difficulty with traditional warehousing/inventory systems is the lack of immediate communications between the workforce and the information system. If some event were to occur that affected operations, there would be no way to quickly detect it and adjust the warehouse workflow. Consider, for example, the situation where a stock picker discovers an empty location that is supposed to contain several hundred items. In a batch-oriented inventory system, other pickers may be on their way to the same location. If this new information cannot be directly communicated to them, a considerable amount of effort, time, and money will be lost.

A mobile, radio-linked terminal would resolve this problem. Suppose that all the stock pickers are carrying lightweight devices that allow them to send and receive messages with the warehouse information system. Further, suppose that the computer system is transmitting picking instructions, pick-by-pick, to each worker. Since the link is two-way, the picker can inform the system that, as in the scenario above, there is insufficient stock at the location. With this information, the system can simply avoid sending any more pickers to that location until it is re-stocked.

In order for "real-time" warehouse control to work effectively, there are two important criteria that the communications device must meet. First, it must be mobile. The terminal must be able to go anywhere the worker (or his vehicle) can go. It would be highly inefficient to require personnel to return to a fixed location terminal in order to receive instructions or send messages. This eliminates any device connected by wire, or that depends on line-of-sight transmission (such as infrared terminals). Second, it must be a full duplex (two-way) transmission. The system must do more than just direct warehouse activity--it must also be able to receive status and confirmation reports from the warehouse floor.

It is not difficult to imagine other ways in which an RF system could be beneficial to inventory control systems. Below are just a few of the possibilities:

- Task Reassignment--If a worker cannot complete a task (as in the case discussed above), the system can immediately reassign the task to another worker, or, direct the worker to an alternate way of completing the task (e.g., retrieve from a different location).

- More Accurate Inventory--The system can be programmed so that stock pickers are directed to count stock at various locations on a rotating basis. For example, each tenth time a picker comes to a location, he is directed to count the stock and report the results to the host computer. Although this technique may slow picking production, it will avoid inventory shutdown or typical cycle counting.

- Control Floor Traffic--Several modern warehouses use a "narrow aisle" configuration, in which storage racks are installed so closely that only one forklift can fit down an aisle at a time. Imagine the chaos that could be caused by a system that did not keep track of the location of each forklift and prevent "aisle contention" from occurring. In a real-time RF system, the computer can avoid sending two pickers down the same aisle at the same time.

- Eliminate Deadheading--"Deadheading" refers to a forklift operator returning to a staging area with an empty forklift, having performed no useful work. Typically, the picker is returning in order to get the next assigned task. If the computer can transmit a new task immediately following completion of the previous task that takes advantage of his current position, an enormous amount of wasted time can be recovered.

- Keep Accurate Statistics--With a real-time system, worker performance can be easily measured. Statistics such as average time per task, time in transit, tasks per hour, etc., can be routinely maintained.

- Process Priority Requests--In a typical batch system, the introduction of a "hot" or priority stock request is a cumbersome affair. With RF technology, however, the normal workflow need not be interrupted to process the order. The order can be integrated so that the priority request immediately becomes a normal task that is assigned via the terminal.

The full range of benefits of a real-time, mobile terminal network depend on the exact nature of the warehouse operation. Clearly, the larger the warehouse, and the more

transactions processed, the greater are the possibilities of significant cost savings.

Current unautomated and batch automated systems usually create operating instructions (pick lists) the night before, the day before, and often the week before they are used. And even if the instructions are created now, they are probably based on data that is at least a week old! Good RF hardware and software allow immediate reaction to current conditions, and that spells $$$.

The "conditions" which should immediately impact operations include:

- New receipt of material
- Order cancellation
- Receipt of "hot" order
- Storage location problems including:
  - Out of stock
  - Wrong item
  - Inaccessibility
  - Physical damage
- Aisle contention
- Item "quarantine"
- Equipment malfunction
- Change in workforce status

III. RF HARDWARE

The RF hardware that has been discussed was first developed in the early 1970s for law enforcement applications. During the mid 1970s, electric utility companies began to implement the use of RF terminals. It was not until the late 1970s, however, that the RF terminal was considered for use in warehouse operations. The advent of bar code capability in the early 1980s made the RF terminal even more feasible for warehousing applications.

At the time our project started, early 1986, there were three substantial manufacturers of R/F hardware. The manufacturer we selected to use was LXE, a Division of Electromagnetic Sciences, Inc. Although we are not attempting to isolate one specific manufacturer, some of the stated characteristics may be unique to LXE products. LXE continues to maintain the greatest marketshare in this industry. Figure 1 shows an LXE vehicle terminal, hand held terminal, and two computer interface units.

Terminals

Warehouse environments are hostile to electronic equipment. Forklifts and trucks are not known for their smooth rides. Dust, humidity, and temperature extremes spell disaster for the standard office terminal. For a terminal to be practical in a warehouse, it must be portable and durable. It must also be able to withstand shock, the environment, and still be usable by warehouse personnel.

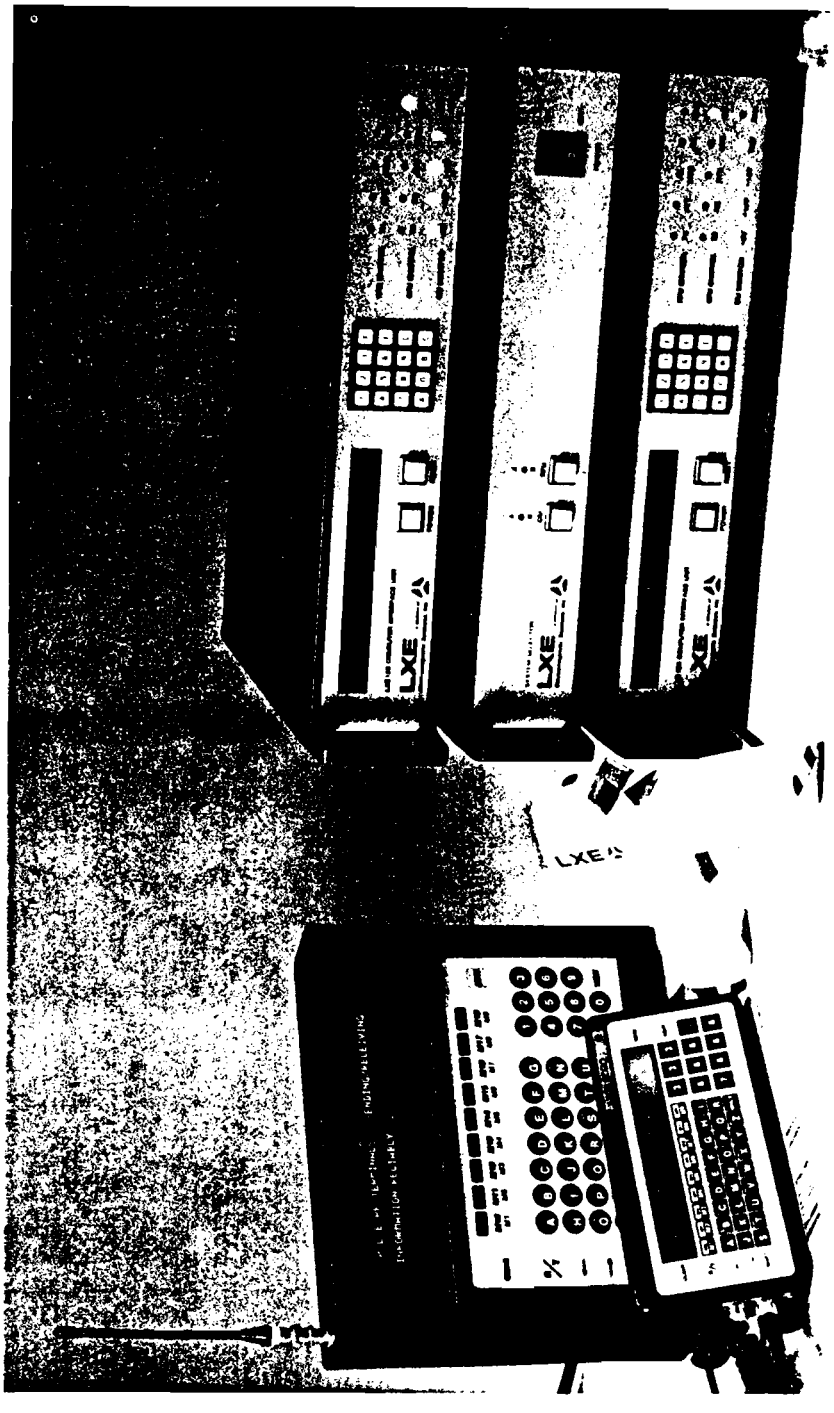
There are two general styles of RF terminals. A hand held terminal that connects to a belt mounted two-way radio and a 12V-36V model that is mounted directly onto a forklift/truck. Both models allow the use of a bar code reader and data entry through a keyboard. Data entry/formatting is facilitated through the use of screens on the terminals. Typical RF terminals have 2-6 lines of display in either LCD or LED, depending on strength of power source. Function keys can be utilized to facilitate one key data entry.

Base Stations

The base station acts as an interface between the computer and the RF terminals. Messages to/from the computer must be received/sent from the proper terminals. Communications error handling and I/O must be dealt with at the base station. The transformation of data to and from radio signals must be transparent to the computer, terminals, and users.

Communications between the base station and RF terminal is performed using VHF radio bands and frequency modulation.

Figure 1



This combination works well in overcoming interference from electrical devices and ensures communication throughout the entire warehouse. The number of terminals serviced by a base station can range from 1 to 100 with the optimum being about 40 terminals/base station.

Several things might affect the performance of the base station, including building construction and porous materials which tend to diminish the radio signals. Nonporous material may in fact enhance the signal. Many base stations have a separately housed radio frequency unit (RFU) that is physically mounted (with antennae(s)) near the signal-strength center of the warehouse.

In addition to terminal communications, the base station must communicate with the computer. Synchronous and asynchronous base station/computer interfaces are available and are usually ordered to fit a specific computer application.

IV. THE SOLUTION - WCS/3000

In this section, we detail the RF solution that was developed to solve the warehousing problem previously described. The total solution consists of a hardware network, system software, and application software.

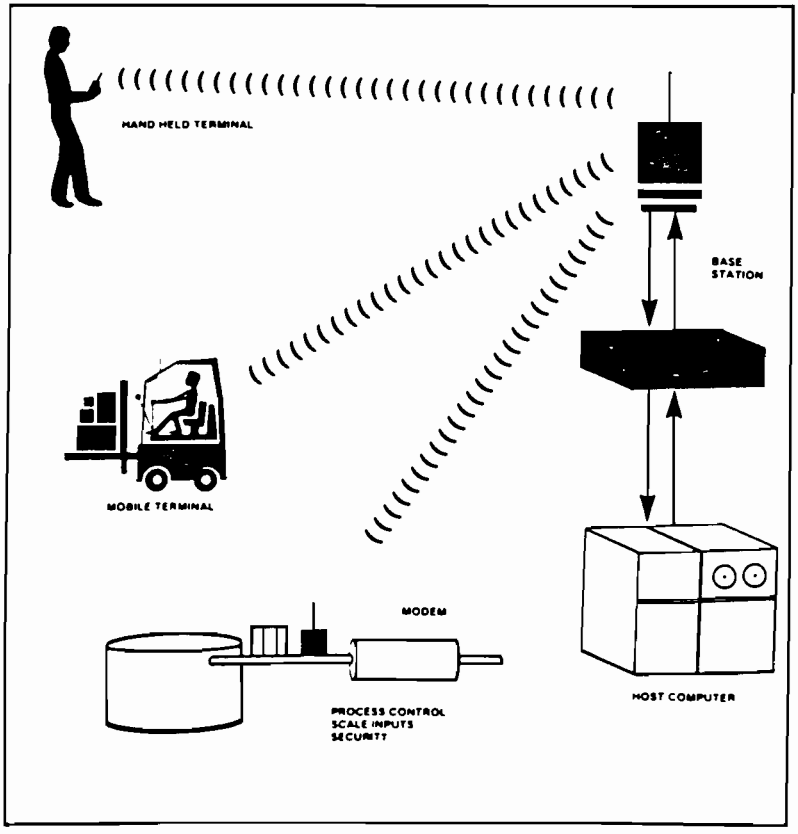
A. Hardware Configuration

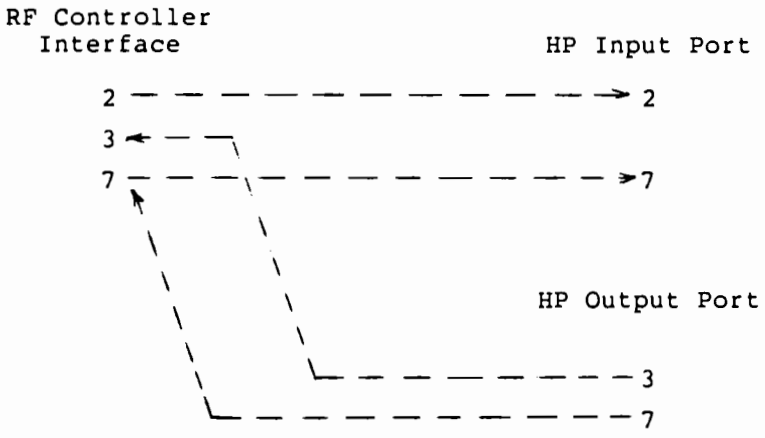
The application as developed can run on any of the HP3000 models, from the series III through the series 70. The choice of processor depends mainly on the number of mobile terminals installed, as well as any ancillary software in use (accounting, MM/3000, etc.). In general, a single mobile terminal does not require the same level of resources as a typical standalone HP terminal, since all mobile terminals are controlled by a single program. For a typical application (24 mobile terminals and 6 CRTs) a series 42 or Micro 3000 performs adequately.

Figure 2 depicts the physical arrangement of hardware required for a WCS/3000 installation. Mobile terminals communicate with the radio frequency unit (RFU) via one or more antennas located throughout the area of coverage. Depending on the layout of the warehouse, it may be necessary to install several receive antennas, since the power output of the mobile terminal is only two watts. Note, however, that in most installations there is only one transmit antenna, since the transmitter can be powered at various levels. The RFU is then connected to the computer interface unit (CIU) using a coaxial cable which can be up to 2,000 feet in length.

From the CIU, a special cable connects to two HP RS-232 ports. One port is used to transmit data from the HP to the CIU, the other is for receiving data. This arrangement is necessary because the HP does not support true full duplex communications; that is, a process cannot be simultaneously reading from and writing to a communications port. Since the CIU has a single standard DB-25 connector, the cable needed to perform the data split looks like this:

Figure 2


HAND HELD TERMINAL
BASE STATION
MOBILE TERMINAL
MODEM
PROCESS CONTROL
SCALE INPUTS
SECURITY
HOST COMPUTER



The HP ports are configured with TERMTYPE = 4; otherwise, they are standard terminal ports.

Communications Protocols

The logic built into the CIU performs extensive checks to ensure that data is not lost between the host computer and the mobile terminals. There are two distinct phases of communication: terminal to CIU and CIU to host (HP). The latter phase (CIU to host) depends on the proper software being installed on the host computer in order for communications to be handled correctly. This interface software is described in the next section.

For the CIU to terminal link, the problem is complicated by the fact that all terminals are sharing a common medium, that is, the radio frequency. In some installations, it is possible to use one frequency for terminal to CIU data, and a separate frequency for CIU to terminal data. This duplex configuration lessens the contention problem, and can also increase the range of the overall system by a factor of two. Regardless of whether the system is full or half duplex, the LXE system uses a sharing mechanism similar to that seen on bus-type local area networks. In essence, the technique works as follows:

1) All mobile terminals have a unique two character ID, which is used as a message prefix. When the CIU transmits information to a terminal, the message is also preceded by the ID so the terminal can tell the message is directed for its use.

2) When the SEND key is pressed, the terminal checks to see if the TX carrier (transmission frequency) is busy. If yes, the terminal waits a short time before trying again. If no, the message is sent.

3) If the CIU sees a message from a terminal, it performs a LRC check on the data and sends back a NAK or ACK. If two terminals have transmitted simulantously, a "collision" will have occurred and no ACK or NAK will be transmitted.

4) If the transmitting terminal does not get an ACK within a predetermined time, it tries again. After a specific number of retries, it tells the operator to send the message again.

On the CIU to host link, messages are transmitted to the HP handler in the sequence received. There is no LRC or CRC check done at this point, only parity checking. The host software is responsible for using the terminal ID to get the message to the proper application program controlling that terminal (son processor). The host is also responsible for transmitting an ACK back to the CIU. This ACK is passed back to the terminal so that it knows whether or not the host received the message.

Messages from the host application to the mobile terminals are processed in the opposite fashion. Each message, with the appropriate terminal ID, is passed to the CIU and on to the TX carrier. When the mobile unit recognizes its ID, it transmits an ACK back to the CIU which is passed back to the HP handler.

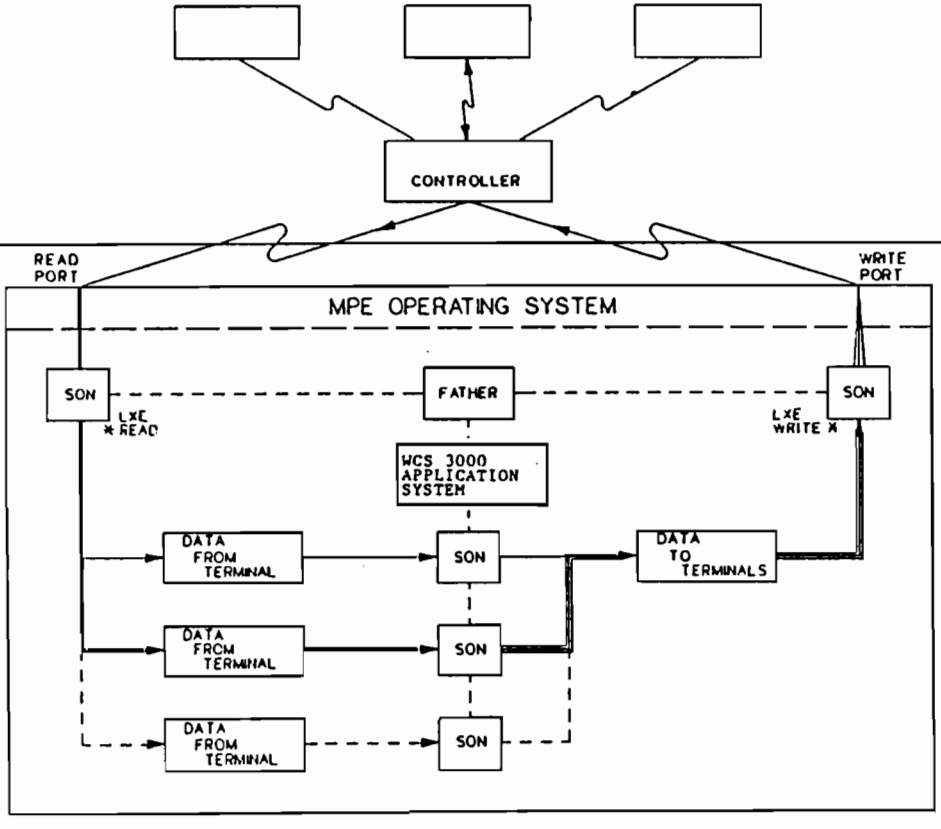
The exact nature of the RF protocols is actually a bit more complicated than this. Most of it, however, is completely transparent to the application program or terminal user. It is important when developing RF applications to understand the basic processes involved, otherwise debugging can become a nightmare.

B. System Software - HP3000 R/F Interface

Although it might be necessary to write your own RF interface software for certain specialized applications, in general it is much more efficient to use a pre-written driver. Since we are focused on LXE equipment in this paper, we will describe the basics of how the LXE HP3000 interface works, and how it communicates with application software.

The system software interface between the remote terminals and the HP3000 makes extensive use of process handling capability of MPE combined with message files to achieve interprocess communication (IPC). Figure 3 shows the way in which data moves between the R/F controller (CIU) and the various software processes involved. Essentially, the driver creates two processes--a read process and a write process--that run constantly in job mode. The write process

Figure 3


CONTROLLER
READ PORT
WRITE PORT
MPE OPERATING SYSTEM
SON
FATHER
SON
LXE * READ
LXE WRITE *
WCS 3000 APPLICATION SYSTEM
DATA FROM TERMINAL
SON
DATA TO TERMINALS
DATA FROM TERMINAL
SON
DATA FROM TERMINAL
SON

(LXEWRT) looks for a file that contains the number of the computer output port, and opens it for write access. Similarly, LXEREAD opens the defined input port for read access. LXEWRT creates a message file, which it then scans looking for write requests. When it finds one, it sends the data to the CIU, handling low level protocol acknowledgements (those required between the host and CIU only).

When the system is brought up, the initialization routine reads a file called TMFILE, which contains the terminal ID of every mobile terminal, along with the name of the application associated with that device. Using process handling, the driver creates a "son" program for each terminal, which is then in charge of making the terminal do whatever it is supposed to do. In addition to starting the application programs, the driver creates a message file that each program uses to communicate with LXEREAD. When LXEREAD receives a message from the CIU, it passes it through to the correct message file based on the terminal ID. If the terminal ID was not defined in TMFILE, the terminal will not get an acknowledgement.

LXEREAD uses five read buffers and no-wait I/O so that a read is outstanding to the input line at all times. Note that LXEREAD sends protocol messages directly to LXEWRT, relieving the application program from that responsibility.

Through the use of one main application program and extensive subprograms, the same code can be shared for all terminals, thereby reducing the impact of a separate process for each terminal. This code uses several intrinsics and occupies approximately 40,000 bytes in object form.

C. Application Software - WCS/3000

We do not intend to go into great detail describing WCS/3000. It is the result of our research and development efforts in R/F technology and our prior knowledge in warehousing systems. We use it as an example of what can be and is being done today using R/F terminals on an HP3000. Figure 4 shows the main system functions from a user perspective. The remainder of this section highlights each subsystem within WCS/3000.

Receiving

As material arrives in the receiving area, WCS/3000 can match it to purchase orders. Discrepant receipts can be suspended for further research. Items requiring inspection are so indicated and can be retained in the receiving area for inspection, or moved into the warehouse for future inspection.

WAREHOUSE CONTROL SYSTEM/3000

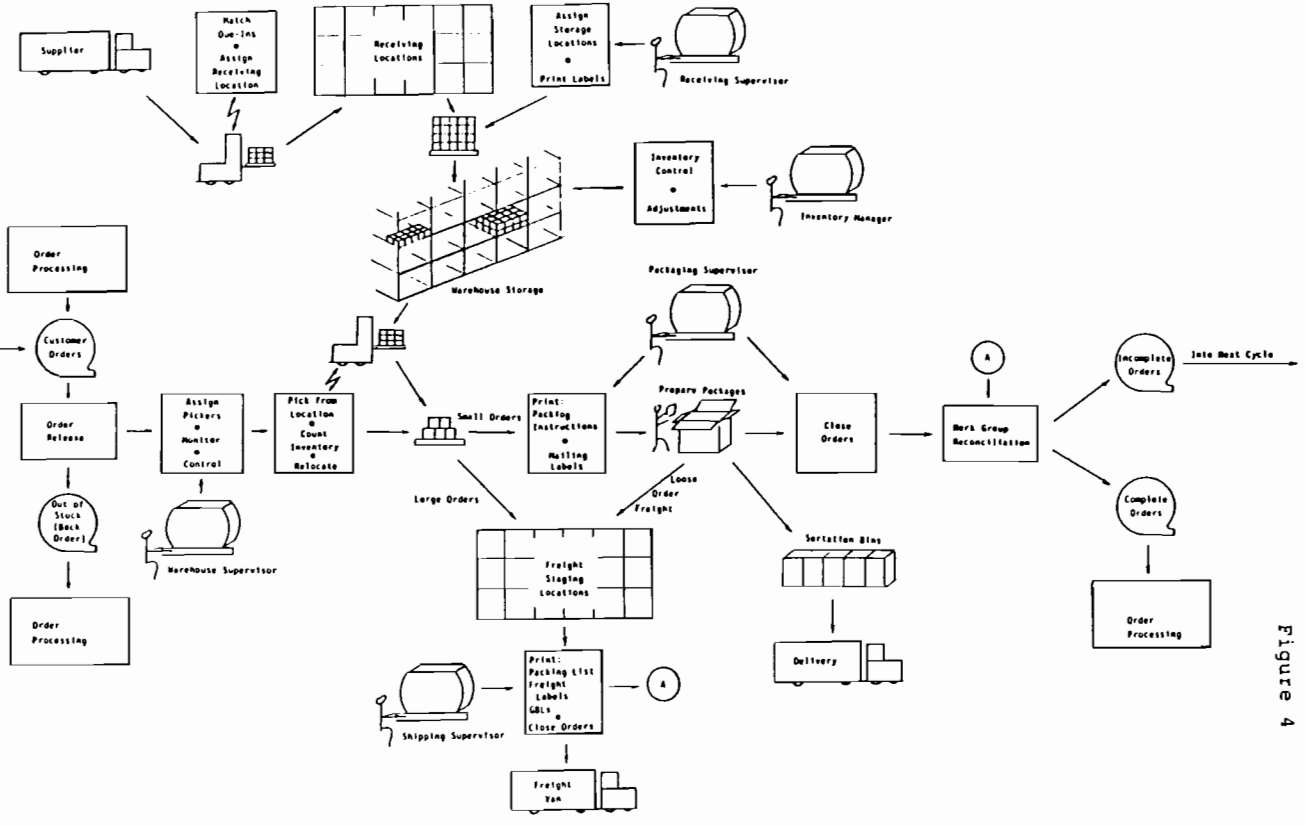


Figure 4

Physical characteristics of the material are recorded--pallet size, carton quantity, etc. WCS/3000 recognizes that different receipts of the same material can be in different carton quantities.

A receiving record is created and the material is placed in a temporary receiving location awaiting permanent storage.

Back orders are checked and automatic "Pick Tasks" are created to fill back orders from the receiving area prior to put-away.

Storage Assignment

Storage assignment is made automatically using WCS/3000 in a random, space-available method to optimize storage utilization. The operator always has the capability to override this feature to make direct storage assignments.

As each pallet is moved from the receiving area, it is either:

- Moved directly into final storage
- Moved to a temporary storage area, creating another move task
- Moved to an automatic guided vehicle (AGV) station, with necessary instructions for AGV transport, and creating an AGV put-away task

Order Interface

Customer/manufacturing orders are entered outside of WCS/3000. Computer-to-computer communications will allow the financial computer to pass orders directly to WCS/3000 without additional data entry. Orders can be prioritized to affect pick sequencing.

The order release process allows orders to be systematically selected for single or consolidated picking. Single customer picking is selected to process large accounts and/or to maximize customer service. Consolidated picking (location picking) is selected to maximize vehicle and operator efficiency and minimize aisle contention.

When orders are received for out-of-stock items, a record is created to inform order entry, and the order is placed on the WCS/3000 back order file.

WCS/3000 has a very sophisticated consolidated picking algorithm which includes:

- Quantity available per location
- FIFO/lot number rotation
- Carton quantities per location
- Pallet/carton/loose picking
- Order round-up, round-down
- Minimum picking threshold per item
- Minimizing number of stops

Managing Task Selection

The warehouse manager has an up-to-date picture of all task requirements (picks, moves, etc.) throughout the warehouse by section, aisle, etc. He or she assigns a territory for each vehicle and the types of tasks each can receive. WCS/3000 warns of territory overlaps. As work progresses, the manager can make any necessary changes to manage the production force.

Automatic Task Assignment

Based on the parameters above, WCS/3000 automatically assigns each material handling equipment (MHE) its next task. Current location is of primary consideration in order to prevent unnecessary "deadheading." Tasks are mixed, only when prescribed by the warehouse manager. When picking orders, the operator indicates when the pallet is full and proceeds to the packing area, directly or by an AGV transfer. Tasks other than normal order picking include:

- Salvage operations
- Excess stock
- Picking empty pallets
- Restocking
- Inventory counting

Packing

After an order picker indicates the pallet is full, it immediately is visible in the packing supervisor's queue. He can select pallets (work groups) for label printing. Labels are printed in customer order, along with packing lists, reflecting any change manifested during the picking process. That is, if an order was picked short, a supplemental order is created for subsequent picking, while the original order label reflects the shortage. In a similar fashion, if a shortage is not detected until packing, the label is manually corrected, and the supplemental order is created. Partially picked orders can be remarried in shipping.

Alternative configurations would allow the shorted order to be repicked from another location prior to packing.

Inventory Control

Inventory is maintained by location and in total. Each stock keeping unit (SKU) can be stored at an unlimited number of locations through our random storage assignment. Each transaction immediately updates total, allocated, and available amounts by location, with appropriate audit trails.

Each SKU is marked with a desired number of days between recounts. As each location reaches its recount point, the next normal visit to that location (pick, move, etc.) will trigger a recount task. Separate recount tasks can be created for locations not visited and counted within acceptable limits.

Managers have the capability to control their inventory with immediate access to up-to-date data provided via WCS/3000. Locations can be created, modified, reserved, etc., as needed. As problems are encountered during normal operations--wrong stock, damaged location, leaks, etc.--locations are instantly marked for inspection. The inventory control team is subsequently assigned reconciliation tasks in location sequence.

Freight Staging

WCS/3000 assigns each major customer a freight staging location(s). Multiple accounts can be assigned per customer. As orders are picked for each customer, the picker is told to direct his pallet to a particular staging area. The freight manager can see all pending orders for this customer, including:

- Unpicked, in the warehouse
- Picking work in progress (WIP)
- Packing WIP
- Completed, awaiting shipment

The freight manager determines when to "close" the staging area for shipment. A packing list is printed to verify contents of shipment. Discrepancies are entered into the system. At that time, additional data are entered (weight, carrier, date, etc.) and a bill of lading (GBL) is printed.

V. IMPACT

A good R/F-based application such as WCS/3000 will have a dramatic effect on the cost and efficiency of warehouse operations. These effects have been seen in the areas of storage space utilization, productivity, cost of inventory, order turnaround, and employee morale. Some Fortune 500 companies, such as Ford and AT&T have already realized these benefits. "The ideal system...an on-the-spot-computer versus bringing the job to the computer. Every transaction is verified for immediate feedback and immediate corrective action if necessary"--William Shirk, senior material management engineer, AT&T.

Storage Space Utilization

R/F technology allows dynamic assignment of storage locations thus permitting more material to be stored in a smaller warehouse. Permanent storage locations are eliminated, and multiple locations per item are simple to handle. Reductions up to 60 percent have been realized.

Cost Of Inventory

Effective inventory management stems from up-to-date knowledge of exact inventory levels. With R/F technology, picking and counting tasks can be intermixed; locations can be automatically counted at specified time intervals; out of stock conditions are immediately communicated to re-order software. These features allow for real "just-in-time" inventory levels and lower safety stock levels. Lower stock levels mean less waste/spoilage and lower inventory investment.

Productivity

Productivity is measured in terms of the amount of work that can be performed in a given period of time. Workers are made more efficient in at least two ways:

- The order of assigned tasks is not predetermined. A worker can be given his next job to do based on his capability and position at that instant. "Deadheading" (carrying an empty load) is minimized.

- Once a storage location is found to have a problem (wrong item, damage, etc.), no other workers are assigned to go there until the problem is corrected.

Order Completion

An order that cannot be filled as directed (shortage error) can be immediately assigned to be picked from a different location and/or worker. If the item is available at all, there need be no measurable interruption in the order through-put time. This means faster, consistent order processing time. This can improve order lead time and customer relations. If a "hot order" is received, it could theoretically be the next order picked in a routine picking procedure.

Employee Morale

Employees using good R/F applications usually have greater confidence in the "system" because they are not repeatedly told to do things that they know are inaccurate. "With radio-linked terminals, warehouse accuracies reach 98 to 99 percent"--Ford Motor Company. With these levels of accuracies, employees perceive the system to be an operational tool for them to use, rather than an obstacle. As each transaction is completed by the workforce, the system will accurately record the assigned task, worker, and duration. Each worker's productivity can be fairly and accurately measured against predetermined "standards." This allows greater flexibility and control in management's relationship with the workforce.

VI. OTHER APPLICATIONS

We have described how this technology has been implemented in a warehousing environment. R/F terminals have already evolved in another direction, in support of police activity. Motorola's product was developed for this purpose, and is currently used as an on-board terminal installed in police cruisers.

R/F in the computer industry is still in its infancy. There will certainly be many more applications as software specialists and system integrators become more aware of its potential, and competition drives down the cost of R/F hardware. Although the current industrial R/F hardware is very ruggedized, it certainly need not be for all applications. Less expensive, less rugged alternatives are certain to emerge, although the mobility factor creates a need for more protection than a standard "wired to the desk" terminal.

In looking for other applications, let's examine the two main values of an R/F terminal network:

- Mobility--The key. Any time you have people on the move that need interactive access, take a serious look at R/F.

- Wiring--Installing a large R/F network may require significant wiring, but individual terminals require no wiring. Locations that find conventional wiring difficult or impossible should also look to R/F as an answer.

Now let's examine current restraints in a potential R/F solution:

- Federal Regulation--These devices are regulated to operate in the 450-470 MHz band for private users. FCC controls the licensing of these frequencies--some areas are already experiencing "crowding" and waiting lists to be licensed in this band. The restricted, Government band (406-420 MHz) also has similar problems.

- Coverage Limitations--The R/F network will operate as physically far as signal coverage will allow. Indoor operations are impacted by building structure and contents (paper being the worst). Outdoor signals can travel much farther. Typical entry-level base stations

and terminals transmit at 2 watts. That provides about 1,000,000 square feet coverage in most indoor structures. Coverage can be increased by:

- Increasing base station power
- Using multiple "receive" antennae
- Using repeaters/modems

The largest North American warehouse (3.2 million square feet) chose this technology. Known outdoor installations exceed 9 square miles.

. Cost--This technology is still very new and R&D costs are still reflected in equipment prices. Typical prices are:

- Base Station - $3,000 - $10,000
- Terminals - $3,000 - $5,000 each
- Installation/Engineering - $2,000 - $10,000

. Lead Time--You should currently plan on 6 months lead time for new installations to cover the time required to receive and install R/F equipment, and receive FCC licensing.

HP SITEWIRE IN THE BUSINESS OFFICE

DUNCAN CAMPBELL
HEWLETT-PACKARD
ROSEVILLE, CALIFORNIA

INTRODUCTION

In today's marketplace, companies are looking for ways to become more competitive while reducing costs. Choosing the right network to improve communications in the business office is the key to cost-effective productivity improvements.

The office network must be able to meet a wide range of information needs, and flexible so it can be adapted to changing needs. The network also has to provide connections with many vendors' systems so that users' investments are protected.

To achieve this, the network has to be based on a comprehensive wiring foundation. Thus site wiring has become an important strategic issue for vendors and users. Customers need a well-designed wiring system that won't have to be overhauled every time new applications or configurations are developed. Ideally, the wiring infrastructure serves as a permanent foundation for the different kinds of networks currently being used or planned for future use.

AT&T's Premises Distribution System (PDS) is Hewlett-Packard's preferred wiring solution for office applications. PDS is a uniform distribution architecture that ties together voice, data and office products from multiple vendors. It is also a full product line that includes the consulting, installation and maintenance services needed for a complete solution.

Hewlett-Packard endorses PDS primarily because of AT&T's commitment to maintaining PDS as an open-wiring architecture with demonstrated multivendor compatibility. HP thus believes that PDS offers the lowest price, highest flexibility, and greatest functionality when compared to competitive offerings.

Hewlett-Packard also recommends unshielded twisted pair cable as the primary wiring medium for office networks. Unshielded twisted pair is an economical alternative to coaxial cable. HP also offers a wide assortment of links that are supported over unshielded twisted pair. These include low-speed asynchronous connections (ATP); integrated voice-data digital PBX connections (DMI); and the new 1 Mbps StarLAN connections. HP will also be supporting the emerging ISDN standards now under development.

HP StarLAN is Hewlett-Packard's LAN solution for HP Vectra and IBM PCs. It provides access to HP Personal Productivity Center (PPC) office automation applications and distributed data processing. HP StarLAN also provides local peripheral sharing with Vectra PCs or HP 3000s acting as servers. HP's StarLAN solution was designed to run over unshielded twisted pair cable to provide a more flexible, low-cost networking solution.

The HP Serial Network for remote PCs offers asynchronous connections to HP's PPC office automation software. The HP Serial Network is also supported over unshielded twisted pair.

BACKBONE ALTERNATIVES

Office subnetworks are linked together with a site backbone. HP offers several backbone technologies.

Baseband

A baseband backbone using ThickLAN coaxial cable is HP's recommended backbone for office wiring solutions. A baseband backbone allows connection of StarLAN or ThinLAN subnetworks. StarLAN networks are connected to the baseband backbone by a StarLAN to 10 Mbps Bridge; ThinLAN networks are connected to the backbone with a ThinLAN Hub.

HP's Repeater Kit makes it possible to extend baseband backbones beyond the 500-meter limit within a building or between buildings close together. Baseband backbones can be

extended between buildings in a campus setting by the IEEE 802.3-compatible fiber optic inter-repeater links.

A baseband backbone offers several important advantages. It is a simple technology that makes installation and maintenance easy. It is based on the IEEE 802.3 industry standard for local area networks, which enables users to choose their solutions regardless of vendor. And the availability -- today -- of a full set of HP accessories enables connections to several subnetworking solutions.

Broadband can provide a suitable backbone technology for office applications, most commonly in manufacturing situation. Broadband is more expensive than baseband, but it provides a flexible topology, multiple channels, and many connection types.

Broadband is the recommended backbone for manufacturing applications. StarLAN or ThinLAN subnetworks can be "bridged" onto a broadband backbone; HP recommends Ungermann-Bass' Buffered Repeaters to provide this connectivity.

SUBNETWORK ALTERNATIVES

Unshielded twisted pair cable is the logical choice for office wiring, largely because it is required for the telecommunication system and thus is often in place already. HP StarLAN is HP's twisted pair LAN solution.

HP ThinLAN is HP's LAN solution for Touchscreen PCs that gives users access to HP's PPC applications and distributed data processing. HP ThinLAN runs at 10 Mbps and provides an alternative to HP StarLAN in technical environments that require office automation solutions as well as technical applications. HP ThinLAN also offers flexibility that makes it easy to retrofit in small areas if a user is implementing it in stages.

SUPPORT FOR HP SITEWIRE

HP SiteWire is implemented by HP's Network Consultants, who work with a group of third-party specialists. HP SiteWire is thus implemented in a series of steps as follows:

1) Network Consulting Proposal -- HP Network Consultants estimate the scope and complexity of a user's networking needs, and present a custom proposal for planning and designing a complete network solution.

2) Network Requirements and Design Analysis -- HP Network Consultants interview users to determine overall network objectives and specific requirements for features like reliability, performance, security, capacity, flexibility and cost. Then a Network Requirements and Design Analysis Report makes recommendations for the wiring system, network technology and functionality of major components.

3) Network Design Report -- Next the consultant creates a detailed proposal for network design based on the requirements analysis. This report includes a design overview, detailed network diagram, description of components, financial estimates, and implementation considerations.

When the planning and design phase is completed, HP assigns a project manager to help ensure that installation goes smoothly.

Hewlett-Packard was recently ranked #1 in customer support for the sixth straight year in the highly respected Datapro survey. This means that users have learned that they can count on HP personnel to be there with help on any phase of the network project.

HP SITEWIRE FOR INTEGRATED FACILITIES

DUNCAN CAMPBELL
HEWLETT-PACKARD
ROSEVILLE, CALIFORNIA

INTRODUCTION

To meet the challenges of today's marketplace, manufacturers are turning to computer integrated manufacturing (CIM) to achieve the quality, productivity and flexibility improvements needed to compete.

For CIM to succeed, organizations require networks that can move information throughout the entire manufacturing organization. In addition to production areas, the typical manufacturing operation includes support and engineering in office environments. The sum of these activity environments is called the Integrated Facility.

A proper wiring foundation is crucial to the success of a network in such a facility. This paper examines the role this foundation plays as a facility-wide information backbone that meets the data, video and security needs of the organization.

HP SITEWIRE: AN OVERVIEW

It's helpful to look at Hewlett-Packard's overall premises networking strategy before focusing on the Integrated Facility. Premises networks are defined as the interconnections of individual workstations and systems within a building or buildings on a single site.

Understanding premises networks requires stepping back to look at the activities that take place at the site as well as the environments in which the activities are performed. The combination of activity and environmental requirements is a major determinant of premise networking needs and strategy.

Four of the five kinds of activities that typically occur at a manufacturing site have certain requirements for information handling. These are: administration, planning and control;

marketing and sales; manufacturing operations; and engineering. The environments in which these activities take place can be divided into three categories: distributed office workstations, the factory floor and the computer center.

HP SiteWire is the name Hewlett-Packard has chosen for its communications wiring infrastructure. The manufacturing network is built around a common communications channel, called a backbone network, that connects different workgroups throughout the facility.

Hewlett-Packard's strategy for HP SiteWire has several elements. It has been designed as a foundation for the complete range of HP AdvanceNet solutions, and it has been designed with specific kinds of customer activities and environments in mind.

HP SiteWire thus includes fully defined backbone and subnetwork architectures, and it offers products and support services from HP and other companies, most notably Ungermann-Bass.

In addition, HP SiteWire is based on industry standards, including AT&T's Premises Distribution System (PDS), IEEE 802.7 broadband cabling, IEEE 802.3 LAN cabling, and EIA's TR41.8 commercial wiring system.

These standards are central to HP SiteWire's two backbone options. The primary and most versatile option is based on the IEEE 802.7 broadband standard. This option makes possible multiple information channels of video, voice and data. It lets users mix terminals, point-to-point links, LANs and more on a single backbone for the most demanding backbone situations.

For subnetwork needs and less complex environments, HP SiteWire offers low-cost twisted pair solutions for terminals and LANs. When users need more rugged subnetworks, IEEE 802.3 ThinLAN is supported on all our factory floor-oriented systems.

THE BROADBAND BACKBONE

A backbone network serves as an information pipeline throughout the plant. To be effective, a backbone must be capable of carrying various kinds of information - more than just datacomm. Voice and video are examples of non-datacomm information that a backbone cable might carry.

The ability to carry several kinds of information requires a backbone technology that can support several information channels at the same time. Tolerance for the harsh electrical environment, as well as the ability to cover long distances, are also key capabilities.

The technology that manufacturers have most widely accepted for backbone networks is broadband. It offers several key features:

--Flexible Topology -- Their branching tree topology allows broadband backbones to spread out in as many directions as needed from a central hub called the head end. Total radial distance from the head end can reach 10 kilometers.

--Multiple Channels -- A typical broadband cable plant has a bandwidth of about 300-450 MHz, which is usually divided into 6 MHz channels. All channels can be made available anywhere the backbone is physically located, and any device on a given frequency can communicate with any other device on the same frequency, which adds up to high connectivity.

--Multimedia -- Broadband backbones can be used for tasks beyond datacomm, including for security systems such as card readers, video cameras and training systems with on-line interactive video.

--High Reliability -- The technology for broadband has been used for years in wiring American cities for cable television. Its reliability has been proven in many settings.

--Many Connection Types -- Broadband can support almost any kind of datacomm connection method, from RS-232-C, to T1, to LANs. It's a nearly universal communications facility.

Thus broadband offers a great many advantages as a backbone technology. But it is important to recognize that the backbone concept itself has value, apart from the network or link type. In fact, an HP 802.3 baseband network could be used as a backbone in a very limited operations networking environment.

HPS 802.3 BASEBAND BACKBONE

By starting with a smaller scope, a user could set up an 802.3 baseband network as a data-only backbone, with plans to move into broadband as a greater physical area is covered and more information is carried.

In choosing to start with an 802.3 baseband network, users should be cognizant of these limitations:

--Maximum length that can be covered (using repeaters) is 1,800 meters;

--The network is more prone to electrical interference than a broadband network, but less than RS-232;

--802.3 is a non-deterministic access method.

HP'S FOUR-STEP MIGRATION STRATEGY

A big advantage of a broadband backbone is the elimination of costs associated with multiple dedicated cables running throughout a facility. Even though converting an existing facility may seem like an insurmountable task, users shouldn't be discouraged. Hewlett-Packard has developed a four-step strategy that enables the manufacturer to integrate a facility onto a broadband backbone.

These steps are:

1) Design and implementation of the cabling system

2) Installation of terminal servers

3) Connection of subnets to the backbone

4) MAP and multivendor networking

Hewlett-Packard provides solutions for steps 1 and 2 through an agreement with a quality Value Added Channel (see below).

The next sections examine these steps in more detail.

DESIGN AND IMPLEMENTATION OF THE CABLING SYSTEM

CIM works best when it's designed from the top down and implemented from the bottom up. Thus the first step toward an integrated production site is to set up the media over which all functional areas will communicate. Design and implementation of the backbone should be done without interrupting current operations.

In addition, the user should work closely with HP to map out both current and projected use of the backbone. One of the biggest pluses of broadband is how easy it is to expand the backbone, primarily because of the branching tree topology. It is easy to add new buildings, floors or areas to a broadband system; it is not so easy to add new taps to a previously wired area.

Despite this ease of expandability, the cable plant should be intentionally over-built to allow for expected growth. This is because the cost of building in excess capacity during construction is a fraction of the cost of expansion. Such expansion costs 30-40% more for companies that don't plan for it in the initial design.

The entire process of installing a cable plant includes several phases. These are:

1) Site survey and design

2) Construction

3) Network testing

4) User training

INSTALLATION OF TERMINAL SERVERS

Once the broadband cable plant has been installed, the next step is to begin utilizing the backbone. In most cases, providing terminal connectivity to any system on the network will be the first service provided.

Terminal servers offer two distinct advantages over point-to-point terminal connections in manufacturing. First, they enable the user to eliminate dedicated wiring between terminals and computer systems. Second, terminal servers enable a terminal to communicate with any computer on the network that is also connected to a terminal server.

In addition, terminal servers allow users to move terminals without re-laying cable; the terminal is simply reconnected to the nearest server.

Broadband terminal servers, sometimes called Network Interface Units (NIUs), packetize the data from the terminal or computer, convert that data to an analog signal, and transmit it onto the broadband network. All the terminal servers on the network share the same broadband channel, and therefore need access to that channel. Installing terminal servers on a broadband LAN results in a simpler, more integrated terminal connect network, and familiarizes users with the broadband backbone.

CONNECTING SUBNETWORKS TO THE BACKBONE

To lower costs and improve the quality of manufacturing processes, it is critical to be able to move information around the functional areas of a facility efficiently. The problem is that HP computers in the Computer Center, Production and Manufacturing Engineering areas can communicate among themselves over 802.3 baseband Local Area Networks, but cannot communicate with each other.

This step of the broadband migration strategy solves the subnet connection problem. Since the broadband backbone easily extends into all the functional areas, the subnetworks can be "bridged" onto the broadband cable by a device similar to a terminal server, with one basic difference: the signals on the baseband side of the bridge use the 802.3 protocol, not the RS-232-C protocol.

MAP AND MULTIVENDOR NETWORKING

Steps 1 through 3 of HP's migration strategy provide a strong foundation for integrating factory communications. What's still missing is an important element of CIM -- multivendor communications.

MAP (Manufacturing Automation Protocol) is a standard that will provide the multivendor communications solution. HP is a major supporter of MAP and a leader in key MAP pilots and demonstrations. In addition, HP is committed to the development of MAP solutions for the HP 1000, the HP 3000, and for the HP 9000 Series 800 computers.

HP AND UNGERMANN-BASS

To provide a solution for Steps 1 and 2 of this migration strategy, Hewlett-Packard's Information Networks Group (ING) has developed an agreement with Ungermann-Bass (UB). Under this agreement HP sales personnel can call in UB to design, quote, sell and install a broadband cable plant. HP then supports certain configurations for connection of

computers and terminals to the network. HP and UB field personnel implement their support services in a coordinated way that avoids "finger pointing" in the event of a problem.

ABSTRACT

HP: A Leader in MAP (Manufacturer's Automation Protocol) Implementation

Duncan Campbell, Hewlett-Packard Co.

HP recognizes the challenges facing major manufacturing companies as they strive to gain an advantage in an increasingly competitive world market. Many of our customers are turning to their manufacturing organizations for improved quality, productivity and flexibility to gain that competitive edge. Our customers, like HP, are turning to CIM to achieve their manufacturing goals. HP offers a CIM networking solution to meet our customer needs in this area. HP recognizes the requirement for connectivity between computers and equipment from different vendors already established in the manufacturing process. By conforming to emerging industry standards HP can provide a multivendor network that supports the customers capital equipment investment while allowing for future growth. For CIM networking this means Manufacturer's Automation Protocol, better known as MAP. For this reason, HP has been and will continue to be a leading supporter and contributor to MAP. This session will cover MAP basics, HP's history with MAP and future directions in multivendor networking solutions for CIM.

MPE Disc Caching

Bryan Carroll
Computer Systems Division
Hewlett Packard
Building 44-MX
19111 Pruneridge Avenue
Cupertino, Ca. 95014

MPE Disc Caching

Introduction

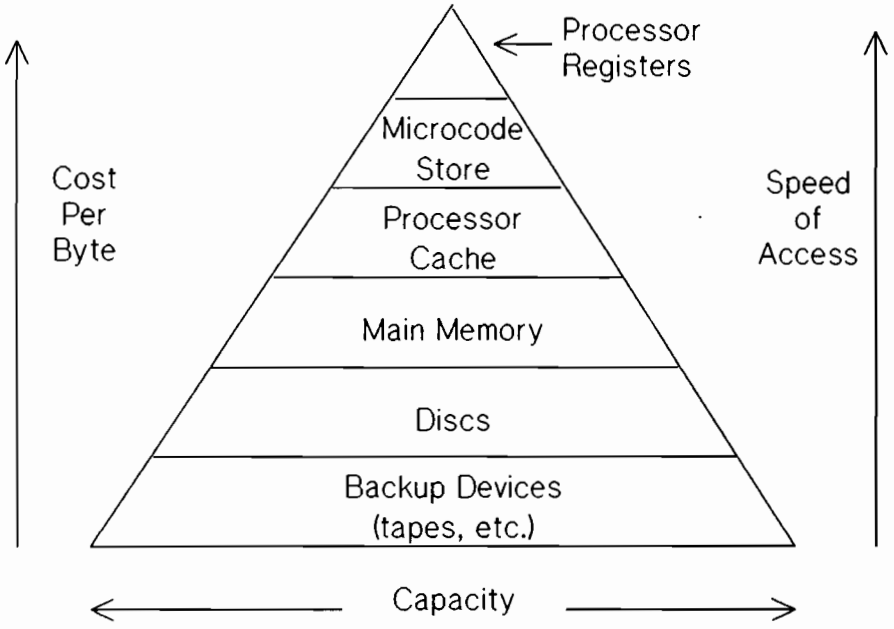
Now that MPE Disc Caching has finally arrived, it has become the job of many users to determine what is happening inside this box we call Disc Caching and learn how to take full advantage of its capabilities. Disc Caching is being widely used by HP3000 users. There is little user documentation available for internal flow and organization to help understand how Disc Caching works. Frequently asked questions by users attempting to understand Disc Caching include: What is a Random/Sequential Fetch Quantum?, What is a Serial Write Queue? or What happens when you issue a STARTCACHE/STOPCACHE command?.

These are excellent questions that can help a System Manager tune his/her system, a programmer increase the integrity of an application system, and an operator understand what it means to start or stop Caching on one or more discs. Based on my experience with the MPE Operating System, I would like to share what I have learned and am continuing to learn about the Disc Caching Subsystem. This paper will focus primarily on the implementation details of Disc Caching including some guidelines for its successful use.

History

During the development of the MPE IV Kernel, algorithms were developed that effectively managed the resources and optimized performance for the existing HP3000 family. Since that time, the Series 64 was introduced with a processor twice as fast and main memory size four times that of the previous top of the line HP3000. The Series 64 performance was found to be very sensitive to disc access times and relatively insensitive to main memory size. The performance bottleneck of the Series 64 was then found to be the disc subsystem. Efforts were then focused toward increasing the disc throughput. Figure 1 illustrates a typical storage hierarchy with the cost per byte and access speed increasing as you move up the scale and capacity increasing as you move down the scale. There have been four different traditional approaches for addressing the bottlenecks between storage levels.

These four approaches are: increase the management algorithms of the levels, increase the capacity of the faster level, speed up the access time of the next lower level, or introduce a new level into the system. The HP3000 research and development lab went to work with these four ideas to determine which, if any, of these we could implement on the HP3000 family.



Standard Computer System Storage Hierarchy
Figure 1

The next lower level of the hierarchy was the disc subsystem and one of the tasks was to determine if we could speed up the access times of the disc subsystem. Processor speeds have been increasing and the cost per byte of semiconductor memory has been dropping by orders of magnitude but the disc technology has not been keeping up with the increases in these other areas. The gap between a semiconductor memory reference and a disc reference is currently about five orders of magnitude and growing. Although discs have been getting more and more dense, the access times have remained relatively constant with the current moving head technology. When a new technology is developed or when major advances can be made in moving head technology the gap between main memory and discs may be narrowed, but until that time, we will have to look elsewhere for improved disc throughput.

Another traditional solution to removing a bottleneck is to introduce a new level into the hierarchy. This level would have

to be introduced into the gap between the disc and main memory to help the Series 64 bottleneck. Research was then done to investigate the possibility of a disc resident cache independent of any higher levels in the hierarchy. Introducing a new level at this point using either bubble or semiconductor memory technology has shown little cost effectiveness. Bubble memory has not been able to keep pace with the density improvements of semiconductor memory which has kept the cost per byte relatively high. Semiconductor RAM memory has become more and more dense and the cost per byte has been dropping but the cost per megabyte is still two orders of magnitude greater than that of discs. Until major technology changes occur, we can expect magnetic discs and semiconductor main memories to be dominant in computer storage hierarchy.

The remaining two approaches have been combined to give us disc caching. The main memory capacity on the HP3000 family has been increased to 16 megabytes on the Series 70 and the Series 58 capacity has been increased to eight megabytes and it is likely that future HP3000s will have even larger main memories. The MPE IV memory manager has been enhanced with the addition of the Disc Caching module to allow what we know as disc caching.

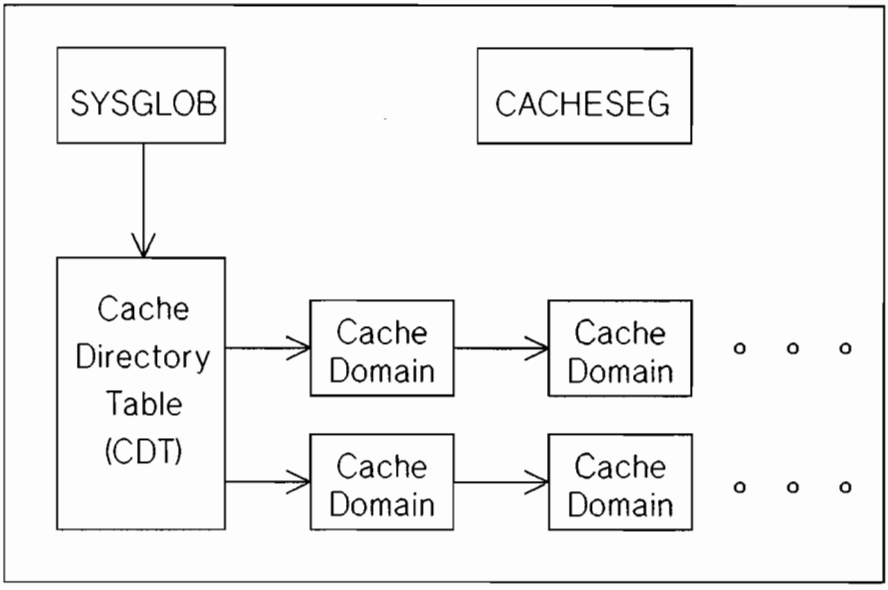
Overview

Disc caching on the HP3000 family is an optional feature that takes advantage of excess CPU power and excess main memory to keep frequently referenced portions of disc buffers in main memory. The Disc Caching Subsystem monitors memory usage and uses as much free memory as it can to increase the chance that the next requested disc record will already be in main memory. There is no permanently allocated portion of memory set aside for disc caching. The algorithm monitors the current amount of memory required for user and system data and code segments and adjusts the amount of memory it uses for Disc Caching accordingly.

Disc Caching takes advantage of its knowledge of the file system to maximize the chance that the next requested record will already be in memory. The goal of the Disc Caching Subsystem is to maximize the number of times a request is made to read a record which is in memory (read hit) and to minimize the number of times that a request is made to write a record to disc which already exists in main memory and is currently being written out to disc (write hit). In both of the above cases, a disc access will be required to satisfy the request. In the HP3000 family, a main memory move instruction, which will be executed when a disc record is found to already be present in memory when a process requests it, takes from 3 to 5 milliseconds. A disc access, which will be required when a requested record is not already in

main memory, takes from 30 to 40 milliseconds, which by comparison is a very long time.

Disc Caching can be enabled or disabled on a disc by disc basis. When Disc Caching is enabled for the first disc, all caching related resources are created and likewise when Disc Caching is disabled on the last disc, all related resources are deleted from the system. This concept will ensure that Disc Caching will not introduce any overhead when it is not enabled.



Disc Caching Data Structures
Figure 2

When caching is enabled on the first disc, a Cache Directory Table (CDT) is created to keep track of all caching related data. The CDT maintains an entry for each disc that has been enabled for caching. The CDT also maintains pointers to the portions of disc that have been saved in main memory (cached) in hopes of satisfying a disc request without having to access the disc. The portions of disc that are memory resident are called cache domains and are variable in size depending on the file structure and the available memory. The cache domains are also connected by a linked list for each disc. This list is linked in increasing disc address order.

As we just mentioned, the Cache Directory Table (CDT) keeps track of all caching related information. There is one CDT for each system with caching enabled. The CDT is pointed to by three fixed low memory locations. These locations are in the area of memory known as System Global (SYSGLOB) and includes a one word Data Segment Table (DST) number, a one word bank address, and a one word bank offset address for the CDT. These words will be zero for a system without caching or when caching is not enabled.

The CDT contains three different types of entries: the Header, Device, and Mapped entries. There is one Header entry in each CDT that contains some global CDT statistics and pointers. The information found in a Header entry includes the number of entries in the table, the first free entry, the number of logical devices currently cached, etc.

There is one Device entry for each logical device currently being cached. The device entries contain information such as the logical device of the disc being cached, pointers to the cache domains, caching statistics, etc. A Device entry is created for every successful STARTCACHE command and deleted for every successful STOPCACHE command. The SHOWCACHE command uses the information found in the Device entries to report caching statistics.

Each Mapped entry corresponds to a cache domain with a logical or physical I/O in progress. The mapped entries are searched by the read and write logic to determine if there is an I/O already in process for the desired disc location. The mapped entries are for every physical any logical I/O performed by the Disc Caching Subsystem. The mapped entries hold information such as the sector start and stop disc address of the I/O, the target memory address of the cache domain, and a number of flags. Each mapped entry pertains to a device entry and all mapped entries belonging to the same device entry are linked together.

A cache domain is a main memory resident temporary storage area for data brought in from the disc. A cache domain may reside in any available area in memory and is very similar to an extra data segment but does not require a data segment table (DST) number.

We have briefly covered all the Disc Caching specific data structures. We will now walk through the Disc Caching commands and see how the resources are used by the Disc Caching subsystem.

The STARTCACHE command is responsible for all operations necessary to enable Disc Caching on the specified disc. The STARTCACHE command executor can be broken down into three different sections. First, some checks are made of the state of caching; second the resources are obtained; and third, the data structures are completed with valid data and caching is allowed to begin.

There are several checks that must be made before any resources can be allocated. The first check is made to see if caching exists on this system and if the specified device will support caching. The device specified in the STARTCACHE command must be a disc. At this writing, all random access discs will support Disc Caching. Once we have determined that the system and logical device will support caching, we determine if caching is currently being used on the system or if we are the first command to enable caching. If caching is not currently active, we must build the CDT and lock it into memory. The CDT cannot be swapped out to virtual memory since this would defeat its primary purpose of reducing disc transfers. In addition to building a CDT, the caching code segment (CACHESEG) must also be loaded into memory and locked for the same reason. The fixed low memory locations are then updated to point to the CDT memory location for later use by the Disc Caching subsystem. If we are successful at all of these checks, we then proceed to collect the needed resources.

Some of the more general resources have been established in the checking phase. We must now gather specific data entries to allow caching to be enabled. The primary entry we are interested in is a device entry in the CDT table. There is no upper limit on the number of device entries that can exist in the CDT. This number is only limited by the maximum number of supported disc drives which is currently 24 per system. If for some reason, we are unable to obtain a device entry in the CDT, a system failure 1009 will result. A device entry is obtained and initialized to zeros, and we now begin the business of completing the entry with valid and useful data.

The device entry is completed and linked in with any other device entries that may exist at this time. All counters are initialized to zeros to ensure integrity of the counters, and the pointers to the mapped entries and cache domains are initialized to zeros since these entries do not exist for this device yet. Some control words and pointers in the header entry are updated to reflect the addition of a new device entry. The final value set which makes caching available on the specified device is a bit in the Device Information Table (DIT) for this device. The bit is set on to indicate caching is enabled and the next disc request to this disc will use caching code to resolve it.

The next disc request that comes down from the file system will be intercepted by the caching routines. If this is a user I/O, the operation will be performed on the users stack and will require approximately 255 (%377) words of stack space to execute. The first caching routine to execute is called CDT'ATTACHIO and will perform checks of the many parameters that are passed to it as well as some more checks specific to disc caching. If the tests pass, we will check to see if caching is in the process of terminating (STOPCACHE command just executed). If this is the case, we will not perform the I/O but instead will wait until the STOPCACHE command has finished executing and will return to perform a physical I/O. If all is well, we will continue by building an entry into a table called the Logical Disc Request (LDR) table. The LDR is a specially formatted Disc Request Queue (DRQ) entry and is not part of a separately configurable table. The LDR changes the DRQ entry format for the specific needs of the Disc Caching subsystem. Once this entry is built, a separate routine called REQUEST'CACHE is called to determine if the requested data is already in memory.

REQUEST'CACHE will attempt to satisfy the request from the cache domains in memory. If it is successful, we have saved a physical access and will return control to the current process without requiring the process to block. If REQUEST'CACHE is not successful, a physical disc access must be initiated to satisfy the request and the process must block while this is done. REQUEST'CACHE will begin looking for the requested data with the device entry in the CDT. The device entry contains a pointer to the first and last mapped entry for this device. Each mapped entry will be searched sequentially for a match with the requested address and length. If a match is not found in the mapped entries, the cache domains are then searched sequentially. HP has investigated implementing other methods of scanning the mapped entries for a specific address in an attempt to increase the throughput of the system, but all the methods investigated so far proved not to provide substantial throughput increases.

As a result of the search, one of four things could happen; the requested data could be found in a cache domain (hit), the data could be found in a mapped entry, the data could not be found (miss) or part of the data could be found (partial hit). If all of the requested data is found, it is moved to the user's data area and control is returned back to the current process who continues executing without being rescheduled. If the data was found in a mapped entry, it means that an I/O is already in progress for this data. If necessary, the disc I/O priority is raised to the priority of the requesting process and the process blocks awaiting the I/O completion. If the data could not be found, or if only part of the data could be found, the request cannot be satisfied without a physical disc access and the process must block. In this case, a new block of memory is

obtained, or if memory pressure exists, the least frequently accessed memory region is flushed to disc, if necessary, and the region is allocated. Once the new cache domain is allocated, the physical disc I/O is performed to read the data from the disc into the new cache domain. The physical disc I/O will usually move a data block larger than the requested data. Data that is close to the requested data is also read into the cache domain to satisfy any requests for that data that might be made.

After the data has been transferred to the user, some counters are updated in the device entry to track how successful the disc caching subsystem has been. These counters are used to compute the numbers displayed by the SHOWCACHE command to be discussed later.

Cached Write

A cached write is very similar to a cached read in its operation but the goal of the writing strategy is the opposite. The goal of the read logic is to maximize the number of times that the requested data is already in memory. The goal of the write logic is to maximize the number of times that a record being sent to the disc is not found in memory. The goal of both systems is to minimize the number of physical disc accesses which require a process to block. When the caching subsystem receives a write request, the mapped entries in the CDT and the cache domains are searched just as they are for a read request. This search also yields the same four possible results; the requested data was found in a cache domain (hit), the data was found in a mapped entry, part of the data was found in a cache domain (partial hit), and the data was not found in any cache domains (miss). If the data was not found in any cache domains, a new cache domain is acquired as with the read miss situation and the data to be written to the disc is put in this new cache domain. The new cache domain is linked into the linked list of cache domains for this logical device and the process is allowed to continue without blocking. A physical write must still be scheduled for this cache domain but this write can take place at a background priority. A background priority means that the physical I/O will not compete for disc accesses with processes that are blocked waiting for a physical I/O to complete. This has the effect of increasing overall throughput because the process issuing the write does not have to wait for this data to be posted to the disc to continue. The data will remain in the cache domain until a time that no higher priority physical disc request is pending for this disc. It is to our advantage then, to minimize our write hits as well as to maximize our read hits.

In the case of a partial hit, a total hit or a hit in a mapped entry when a write function has been requested, more work must be

done. In all cases, the current process must block for a physical write to complete before continuing. The cache domain that caused the hit during the search will be flushed to the disc and then the same logic used for a write miss described above will be used to add the data being written to a cache domain. When flushing the cache domain to the disc, a Disc Request Queue (DRQ) entry must be obtained and completed and passed to the disc driver. If this cache domain has been previously written to, a DRQ entry may already exist at a background priority so a search must be made of the DRQ. If an entry is found for the cache domain we need to flush to the disc, its priority is raised to the priority of the process forcing the disc I/O and the process is blocked pending the I/O completion. When the I/O completes, the cache domain that was just flushed is filled with the data that the blocked process is writing and a background disc write is initiated for the new cache domain. Once the background write is initiated, the process that initially requested the write is unblocked and allowed to continue processing. The blocked process does not wait for the data it is sending to be posted to the disc.

CACHECONTROL

We have seen how Disc Caching gets started and how a typical read and write operation take place, however, you probably have many questions in your mind about how you can control Disc Caching. The CACHECONTROL (appropriately named) command and the FSETMODE intrinsic will give you some control over how things happen in the Disc Caching subsystem. The CACHECONTROL command has three options, SEQUENTIAL, RANDOM and BLOCKONWRITE.

The SEQUENTIAL and RANDOM options allow you to set a target number of sectors that you want Disc Caching to bring into a cache domain in memory every time it encounters a read miss. These are both called "fetch quantums". The size of the cache domain built, following a write miss, is always the size of the requested transfer so write requests are not affected by any fetch quantums. In the case of a read miss, Disc Caching will take the requested size and round it up. The value selected will be the largest even multiple of the requested size that is less than or equal to the fetch quantum, but never less than the requested size. This means that the requested size will always be brought into memory and usually several times more than the requested size will be brought in. The values of the fetch quantums can range from a low of only 1 sector to a maximum of 96 sectors. The default values are 16 sectors for the RANDOM option and 96 for the SEQUENTIAL option. Disc caching uses the file system information concerning the type of access the file is being used for to know which option to use on any given read. If the read request is a FREAD, Disc Caching will choose the SEQUENTIAL

option value. If the read request is any other type of read, including FREADBACKWARD, Disc Caching uses the RANDOM fetch quantum.

The setting of these two values can vary your performance and total disc throughput, and there are many factors to consider in setting these values for your shop. We will address a few of the factors here briefly. In most cases, the default settings for both options will give the best results. These values were chosen after extensive testing using varying system loads and configurations.

The SEQUENTIAL option should usually be set as high as it can. If a file is being accessed sequentially, records will be read in sequence so the more records you can put into memory, the more disc accesses can be saved. One important note about sequential file access is that after the last record in the cache domain of a sequential file is transferred to the users stack, the cache domain is marked as a Recoverable Overlay Candidate (ROC) which means that the memory space used by that cache domain can now be used by some other operation if it is needed. This is the method the memory manager uses to determine who has been least recently accessed.

The RANDOM option gets a little more complex. There are several reasons why a default value of 16 was chosen, the most important of which deals with the internal disc resident hardware buffer found in the CS80 discs. There is a 4K byte hardware buffer built into the CS80 discs to reduce the time encountered because of latency (disc rotational delay) for small transfers. Any transfer requests greater than about 16 sectors will not use this hardware buffer and will cause the transfer to be slower. In some cases, this slightly slower transfer may be advantageous because of a possible higher hit rate, but you should consider the options and experiment with your system. The number of process stops displayed by the SHOWCACHE command is another indicator of system performance and in particular the Disc Caching performance. This number is often affected by the RANDOM fetch quantum value. If the process stops, as a percent of cache requests (process stops/cache requests), is around 13% or greater, this is an indicator of excessive process stops that may be due to an improperly set RANDOM fetch quantum. If the RANDOM fetch quantum is set too high, then you may be encountering many write hits which causes a disc access and a process stop. You may want to experiment with a lower RANDOM fetch quantum in this case to reduce the chance of a write hit.

The size of both fetch quantums also impacts the amount of memory that will be used for caching. If you are short on memory, it would probably be best to experiment with smaller fetch quantums to relieve any memory pressure that may be present. Since disc caching may be enabled and disabled on a disc by disc basis, you should look at the performance of each disc individually. In some cases it may be best to stop caching on one or two discs

with a poor read percentage, read hit or high write hit rate, and release the memory used by those discs to increase the performance of the good performing discs even more. The bottom line concerning the fetch quantums is to experiment in your own shop with your own application mix running to see what settings give you the best performance.

The BLOCKONWRITE option of the CACHECONTROL command is an option that will allow you to sacrifice performance for data integrity. The BLOCKONWRITE option, if on, will tell the disc caching subsystem to always cause a write operation to block the requesting process until the data is recorded on the disc. This option will prevent the situation where data could be written by a program and held in a cache domain when a system failure occurs. This would result in the data being lost since memory contents are not recovered after a system failure. This option will always decrease performance (up to 30%) since unlike the case of a write miss, every write operation will cause a physical disc access to occur while the requesting process waits. This can be a sensitive issue which must be addressed by each individual shop.

The BLOCKONWRITE option on the CACHECONTROL command will establish the BLOCKONWRITE option for the entire system. You may also establish the BLOCKONWRITE option on a file by file basis with the FSETMODE intrinsic. After a file is FOPENed, a call to FSETMODE with the appropriate parameters will allow you to establish the BLOCKONWRITE feature even if the rest of the system is not using BLOCKONWRITE. Image with logging and KSAM both use the FSETMODE intrinsic to enable BLOCKONWRITE so you do not have to worry about those two subsystems.

Another option available with the FSETMODE intrinsic and disc caching is the Serial Write Queue (SWQ). Since a write operation does not always transfer the data directly to the disc, several write operations could all be waiting in cache domains at the same time. Since all of these requests will be queued waiting to be transferred to the disc at the same priority, they may not arrive at the disc in the same order that they were written in. In most cases this will not be important but in a few other cases it can be critical. It is for this reason that the Serial Write Queue was developed and made available. When the Serial Write Queue is specified for a file with the FSETMODE intrinsic, you are guaranteed that all writes performed against that file will be written to the disc in the order they were issued by the program. The SWQ is a First In First Out (FIFO) linked list of entries in the Disc Request Queue. There is only one SWQ for the entire system so as an extreme example if the entire system used the SWQ, only one disc write would be serviced at a time even though there may be multiple discs, controllers, GICs, and even multiple Inter Module Buses (IMB's). This would cause extreme system degradation, so caution should be used in determining if the SWQ is necessary for your application.

Here are a few closing notes about BLOCKONWRITE and the Serial Write Queue.

1. Setting BLOCKONWRITE on globally with the CACHECONTROL command will disable the Serial Write Queue since all writes are occurring as they are requested in the order that they are requested.

2. Both Image and KSAM use BLOCKONWRITE and the Serial Write Queue for "important" transaction such as Intrinsic Level Recovery (ILR) for Image.

3. With BLOCKONWRITE off, the disc writes will occur after the requesting process has issued the write and continued its processing. If a write error occurs when the cache domain is finally flushed to the disc (i.e. a bad track), a system failure 650 or system failure 651 will result. If the same disc error occurred with BLOCKONWRITE on, or without caching enabled, a write error would be returned to the file system.

SHOWCACHE

The SHOWCACHE command is the only way to get Disc Caching statistics from the Disc Caching subsystem without purchasing any performance monitoring programs or performance consulting. The output from the SHOWCACHE command is, however, a very simple and accurate picture of the current status of the Disc Caching system. Figure 3 shows a sample SHOWCACHE output. The output shows one line per disc with a total line showing a summary of the combined Disc Caching subsystem.

DISC LDEV	CACHE REQUESTS	READ HIT%	WRITE HIT%	READ%	PROCESS STOPS	K-BYTES	% OF MEMORY	CACHE DOMAINS
1	264738	86	79	80	34625	575	7	349
2	113893	67	66	68	27575	337	4	132
11	165391	83	70	79	23714	715	8	308
12	715884	86	84	86	115000	1970	24	673
Total	1259906	84	78	82	200914	3597	44	1462

69% of user I/Os eliminated.
Data overhead is 374K bytes.
Sequential fetch quantum is 96 sectors.
Random fetch quantum is 16 sectors.
Block on Write = NO.

SHOWCACHE Output
Figure 3

DISC LDEV is the logical device of the cached disc for which the following data applies.

CACHE REQUESTS are the number of times a process has requested a read or write operation to this disc, since Disc Caching was started on this disc.

READ HIT% is the number of times a read request was satisfied without a physical disc access divided by the total number of read requests made to this disc since caching was started on this device. You should try to maximize this number (i.e. greater than 60%).

WRITE HIT% is the number of times a write request required a process to block in order to satisfy the request divided by the total number of write requests made to this disc since caching was started on this device. You should try to minimize this number (i.e. less than 50%).

READ% is the total number of read requests divided by the total number of cache requests since caching was enabled on this device. The difference between this number and 100% is the write percentage. Caching will show the greatest performance gain when the read percentage is highest.

PROCESS STOPS is the total number of times that a process had to be blocked because a physical disc access had to be performed. If the process stops, as a percentage of cache requests, reaches about 13% or greater, adjusting the fetch quantums may increase throughput.

K-BYTES is the number of bytes in thousands used by caching this disc. This is for your information only.

% OF MEMORY is the number of K-BYTES used by caching this disc as a percentage of the total memory available on the system.

CACHE DOMAINS are the total number of memory regions set up and being used to cache this disc. The average size of a cache domain for this disc can be calculated by dividing the number of K-BYTES by the number of cache domains.

The information at the bottom of the display indicates the status of the fetch quantums and the BLOCKONWRITE flag. In addition, the data overhead and percent of user I/O eliminated is calculated and presented here. The data overhead number includes the memory used by the CDT, the header and the trailer information for each cache domain. The percent of user I/O's eliminated is a calculation of the overall success of disc caching on this system. It is calculated by multiplying the overall read hit percentage by the read percentage and does not take into consideration any write operations transferred at a background priority.

When you are trying to determine what combination of Disc Caching settings work best in your shop, it is useful to try a setting for an hour or more and check your success with the SHOWCACHE command. Since the totals displayed in the SHOWCACHE command are not reset, you must stop and then restart caching before each test to initialize the counters. This can be a high overhead operation. A contributed program has been written and contributed to the INTEREX Contributed Library to help this situation. This program will lock the Caching SIR and zero all totals for all currently cached discs. This program should be used with care as you would any other contributed library program. The program is called ZEROTOT.

STOPCACHE

The STOPCACHE command is just like the STARTCACHE command only in reverse. The executor of the STOPCACHE command can be broken down into the same three sections that the STARTCACHE command was, except in the STOPCACHE command, they are executed in a different order. The first section performs some checks of the state of caching, the second section flushed all cache domains related to this device to the disc, and the final section releases all resources.

The first check is made to determine that caching exists on the system and that caching is enabled. Additionally, several checks are made against the system, such as verifying that the logical

device number passed to the routine is a disc device and is currently cached.

When it seems that we can perform the requested function of stopping caching on the specified logical device, we post all "dirty" cache domains to the disc. A "dirty" cache domain is one in which the data in the cache domain has been changed after it was brought into memory so that the copy in memory is different than the copy on the disc. Next, the bit in the device information table (DIT) which indicates caching is enabled, is turned off for the specified device. This will prevent any more cache domains from being built until we are done with the STOPCACHE command.

When all the data has been posted to the disc and we have prevented any more data from being put into cache domains, we release the resources used by caching this device. This includes the device and all associated mapped entries in the CDT and all the cache domains. If this was the only disc currently being cached, we will also unlock and release the CDT, and unlock the caching code segment so that it can be unloaded as soon as we complete the STOPCACHE command.

Disc Controller Cache

Disc Controller Cache is another recent attempt to reduce the I/O bottleneck found on many HP3000's. Like MPE Disc Caching, Disc Controller Cache has introduced a new layer into the storage hierarchy between the disc drive and main memory. With another new product to increase disc throughput, we must answer the question of which one do we choose. For more information about Disc Controller Cache, consult reference "Aldinger".

Disc Controller Cache has introduced a new set of hardware and some software (firmware actually) between the HP3000 and the disc. The hardware includes a redesigned controller and additional memory to use as the cache. The new firmware will add less than one millisecond to an average disc access to maintain and search the cache for a hit. Controller cache will have an advantage over MPE Disc Cache when the main system CPU is very busy (more than 90% excluding batch jobs) because it has its own CPU to use in searching the cache domains.

The additional memory added to the disc drive as part of Controller Cache can also provide an advantage over MPE Disc Caching if memory is a constraint on the system since it has its own memory. Controller cache can only provide this relief while maintaining the same performance if the amount of main memory used by MPE Disc Caching to cache that drive is equal or less than the amount of memory in the drive. If, for example, MPE Disc Caching is using 1.2 megabytes of main memory to cache a drive, then upgrading the drive to a 7933/XP which only has one megabyte of memory and turning MPE Disc Caching off, could cause

a degradation in performance. MPE Disc Caching required 1.2 megabytes of memory to attain the performance level. Reducing the available memory to 1.0 megabytes will probably reduce the ability of the caching algorithms to maintain the same performance level.

Disc Controller Cache also has some channel overhead which does not exist with MPE Disc Caching. Even in the best case of a read hit with Controller Cache, the disc must still compete for access to the channel which may cause a small degradation on busy systems.

The new 7936/37 XP disc drives have implemented caching differently than the 7933/35 XP disc drives. I will not get into the implementation details but it is important to point out some of the differences. The 7936/37 XP Controller Cache has implemented a write cache complete with a battery backup. The write cache, which did not exist on the 7933/35 XP Controller Cache, is a single 4Kbyte cache. This means that the first write request that is sent to the drive will be put in the cache. If a second write request is received prior to flushing the write cache, the second request will queue waiting for the write cache to be flushed.

The read cache size on the 7936/37 XP has been increased from one megabyte on the 7933/35 XP to two megabytes. The internal hardware buffer for use with RPS has also been increased from 4Kbytes or 16 sectors on the 7933/35 XP to 32Kbytes or 128 sectors on the 7936/37 XP. This changes our original thinking about setting the random fetch quantum to 16 sectors and allows us more freedom in experimenting with the random fetch quantum.

Disc Controller Cache Guidelines

The following general guidelines can be used when determining whether Disc Controller Cache could replace MPE Disc Cache and maintain or increase system throughput. Keep in mind that these are general guidelines. Consult your local HP office for information concerning your particular site if any of these guidelines fall into a gray area.

A predominately online system with greater than 90% CPU busy excluding any background batch jobs is a good candidate for Disc Controller Cache. In this case, MPE Disc Caching may be using some of the CPU that could be given to users if Controller Cache can maintain the same disc I/O throughput.

A read to write ratio of 3:1 or better is very important. A good read to write ratio is always good for caching but writes can be more of a problem for Controller Cache. The 7933/35 XP provides no performance increase for writes and the 7936/37 XP has only a

single write cache buffer while MPE Disc Caching can have any number of write domains.

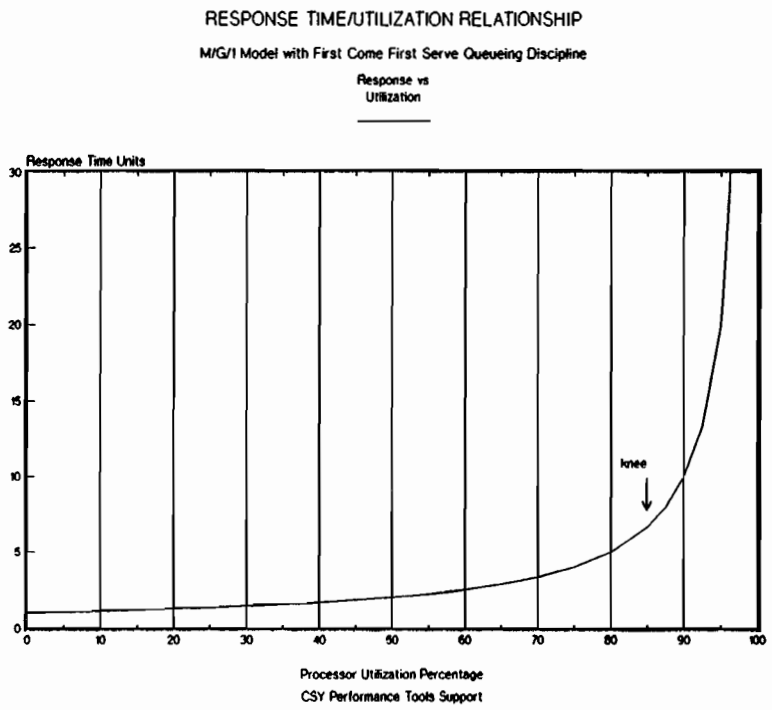
Care should be taken when adding Controller Cache and turning MPE Disc Caching off to consider the required memory. MPE Disc Caching may have excellent statistics for a given drive but may require large amounts of memory to maintain these statistics. This will usually not be a problem with the two megabytes of memory available with the 7936/37 XP drives but should be checked on all drives. The MPE SHOWCACHE command should be consulted to determine the amount of memory used by caching each individual drive.

The read hit percentage should be 75% or better. Since disc reads are the only accesses that can be eliminated, a high hit rate is important to Controller Caches success. MPE Disc Cache was born with knowledge of the MPE file system, the directory, and other areas of MPE which perform I/O and can in all cases make better decisions pertaining to caching like quantums, prefetch, etc.

These are a few of the most important considerations about purchasing Disc Controller Cache over MPE Disc Caching. You should consult your local HP office for specifics concerning your site.

Should I Purchase Caching? - Which One?

When we defined MPE Disc Caching for the HP3000 family of computers, we said it takes advantage of excess CPU power and excess main memory. It is very important that you take a careful look at your system before attempting to use Disc Caching. In most cases MPE Disc Caching will improve your system performance but in some cases, it could cause a performance degradation.



Knee of the Curve
Figure 4

Disc caching will add a small amount of system overhead to your system in terms of CPU. Although the actual overhead is relatively small, it can cause dramatic response time increases if the increase puts your total CPU utilization over a threshold that I will call the "Knee of the Curve". Figure 4 shows a graph of the knee of the curve concept. There is a point all systems can reach where a small increase in CPU utilization will cause a very large increase in response time. The actual CPU utilization figure will vary with the individual processor and the application mix running on the system. If a system was running very near the knee of the curve prior to installing disc caching, the Disc Caching overhead could cause enough additional CPU utilization to push the system over the knee of the curve and into dramatically longer response times. A system could also be pushed over this knee of the curve if any additional system load was added either along with Disc Caching or independently.

In the cases where MPE Disc Caching or other system loads put the CPU past the knee of the curve, we now have Disc Controller Cache. Disc Controller Cache can help by offloading the CPU required to do disc caching to the CPU built into each disc controller. Care should be taken to ensure that Disc Controller

Cache has enough resources (like memory) to maintain the MPE Disc Caching performance level.

Disc Controller Cache has incentives other than performance for its purchase including lower cost maintenance contracts because of the lower wear and tear on the mechanical parts of the drive. With these other incentives available, it is often advantageous to purchase both MPE Disc Caching and Disc Controller Cache. Both forms of caching could be used until the CPU or main Memory becomes a bottleneck when MPE Disc Caching could be turned off. It is also beneficial to use both forms of caching when the BLOCKONWRITE flag is turned on for MPE Disc Caching.

Summary

Both implementations of disc caching have already proven very successful for the HP3000 when their tradeoffs are considered. Both offer excellent price/performance and raise the overall performance of the HP3000 to new heights.

References

Rick Aldinger, "Disc Performance - What Is It?", Business Proceedings - HP3000, INTEREX Detroit Conference, Detroit, MI. Volume I, Paper 3201, September, 28, 1986.

J. R. Busch and A. J Kondoff, "Disc Caching in the System Processing Units of the HP3000 Family of Computers," HP Journal, February 1985.

J. R. Busch, "The MPE IV Kernel: History, Structure, and Strategies," Proceedings of the HP3000 Internals Users Group Conference, Orlando, April 27, 1982.

J. R. Busch and A. J. Kondoff, "MPE Disc Cache: In Perspective," Proceedings of the HP3000 International Users Group Conference, Edinburgh, October 1983.

"Series 64 with Disc Caching Beats IBM 3033 in Batch MRP Run," Hewlett-Packard Computer News, October 1, 1983.

Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik, Quantitative System Performance, Perntice-Hall Englewook Cliffs, New Jersey, 1984.

MPE V Tables Manual, Hewlett Packard Company, 1984.

ABSTRACT

Common Man's Performance Guide

Bryan Carroll, Hewlett Packard Company

After attending many performance round tables and other performance discussions, I have seen a need to collect all available guidelines and rules of thumb together in one place. This paper will provide a list of tested guidelines and performance numbers that I have collected over the years. These guidelines have been collected from respectable sources both inside HP and from knowledgable individuals in the HP 3000 community.

Many of the ideas presented here are not new. What is presented is a collection of the best performance information available pertaining to the HP 3000 written so that the non-performance trained engineer, the MIS Manager, the System Manager, the "Common Man" can understand the concepts and ideas.

A few of the areas to be covered include the CPU, Disc I/O including Disc Caching and Controller Caching, Main Memory, TurboIMAGE, UDC's, and Performance Tools.

HPTrend

Bryan Carroll
Computer Systems Division
Hewlett Packard
Building 44-MX
19111 Pruneridge Avenue
Cupertino, Ca. 95014

HPTREND

HPTREND is a product from Hewlett Packard that is intended to be used to measure your usage of several critical system resources over time and report that usage back to you. Hopefully you should be able to use these reports to better understand, and possibly even better utilize, the HP3000 in YOUR environment. These reports may also be useful to forecast the impact of changing requirements on your system and allow you to better plan the steps needed to maintain the system performance you would like.

If you have been using your HP3000 very long you might have noticed the many things you can ask it to do for you. You might be running applications you or your company have written, HP provided applications, or applications written by a "third party". These applications might range from Office Automation, through Data Base maintenance, and on to Satellite Image Processing, all on the same system and at the same time. Such is the nature of the HP3000. It's little wonder then, that it is difficult to find two HP3000's that have the same load imposed on them.

HP has done much work on characterizing the HP3000 and how it responds to various system loads. The missing part in this process is knowing what load you are placing on YOUR system. This is where trend analysis comes in. It attempts to measure certain critical system resources and log how much of each one you demanded through out the day.

HPTrend, as its name implies, provides long term trends in resource usage. HPTrend is designed to report trends from a period as short as one month to two years. When analyzing HPTrend data, longer collection periods will enable the analyst to make stronger conclusions. A collection period of at least three months is best.

HP also provides a product, HP SNAPSHOT, that will take a deeper look at your system and attempt to identify any system bottlenecks in detail. This product will capture large volumes of data which makes collection for more than about one day usually prohibitive. HP SNAPSHOT can tell you exactly what occurred on your system on the day of the test but does not have the ability to track long term trends on your system. Both HPTrend and HP SNAPSHOT have their place in the HP3000 world serving different needs.

Trend analysis captures data in less detail, but over a greater time interval, than the HP SNAPSHOT product. If you are looking

for system usage trends, you should be concerned with questions like "how did my CPU usage change when I added that new application last month?" rather than "The system was slow from 11:15 - 11:20 today, why?".

Critical System Performance Indicators:

The HP3000 can be characterized from two points of view. First you probably think of system performance in application terms. (Sales orders per day, response time or time needed to complete the nightly batch work). This usually works for a single application but rapidly gets to be a problem on a multi-user system like the HP3000. (For instance, How many sales orders equals one overhead slide produced? How many electronic mail messages must you send to equal one Satellite Image Processed?)

In order to produce a report with meaningful data on all HP3000 systems, HPTREND must take the other view of system performance and loading. It measures things from the system's point of view and relate everything in terms of critical system resources. Over time we have found the following items to most often be related to the user's view of performance:

CPU speed and usage; This is the system's ability to perform calculations, comparisons, and other program logic.

Main Memory; The amount of data that can be stored in the processors main (high speed semiconductor) memory vs the demands placed on it by the applications.

Disc I/O; The rate at which data can be stored and retrieved from the system disc drives and the need for such activity. (NOTE: Disc I/O rates are not the same as disc capacity. Capacity says how much data can you hold (ie. 404 megabytes) while I/O rate is how fast can you get at it (i.e. 30 I/O's per second)).

Other peripherals such as printers, magnetic tapes, and terminals may also limit the usefulness of your system, depending on the applications you are running.

HPTREND attempts to gather information as to how much of each of these resources were used on your system, and where possible, who was responsible for using them. In order to capture this data, HPTREND utilizes two features of the MPE Operating System.

MPE LOG FILES

The MPE Logging facility has been present on MPE since its inception. It logs "events" as they occur to special files in PUB.SYS. Your system manager can enable or disable up to eighteen or more different events that they would like to see logged. Once enabled, MPE takes care of writing information about each event as it occurs to the disc file. Special procedures have been

created to switch to a new log file as an old one is filled or each time the system is re-booted. You can identify these log files by their names. They all start with "LOG" then have four numbers and are in the PUB.SYS group. The numbers are incremented each time a new log file is created in order to obtain the new log file name.

HPTREND is particularity interested in analyzing the following log file records. Make sure that at least these events are turned ON for logging. (See the System Operations manual for details on how to configure the system in this manner).

```
type  1 Logging enabled (if not on then NO logging occurs)
type  2 Job Initiation  (A Job or Session Logs on)
type  3 Job Termination (A Job or Session Logs OFF)
type  5 File Close      (A file (disc or other) is used)
type  6 System Shutdown (=SHUTDOWN from your console)
type  8 Spooling        (Print to a spooled printer)
```

For diagnostic purposes, HP recommends that you enable types 7 and 11 as well (Power failures and I/O errors) although HPTREND does not use these records. You MUST enable at least the itemized logging records listed above or the HPTREND report can not be accurately generated.

MPE Logging was designed to be of minimal overhead to a running system. As such it doesn't log every event as it occurs but rather logs only summaries of activity at some logical end point. For instance, it doesn't log each time you do an I/O to a disc file but rather just how many I/O's you did, once you are finished with that disc file and close it. It doesn't try to keep a running tally of your varying CPU requirements as your job ran, but rather just logs the total CPU time you used at the time you log off the HP3000. In this way MPE Logging doesn't add significant impact itself to your system resource usage.

Normally HPTREND analyzes your MPE Log files each night at midnight. This usually only takes a minute or two and should not interfere with your other activities. The program is smart enough to determine which log files have been added to the disc since it last analyzed them and so can 'catch up' if HPTREND was inadvertently shut down for a time.

You may decide to purge the log files from PUB.SYS occasionally in order to save disc space. You might take care not to purge them before HPTREND can analyze them otherwise valuable data can be lost. As we shall see later, having the MPE Log file data available over a long period of time can be invaluable for many reasons. If the disc space they occupy becomes to large to justify then you should archive them to magnetic tape or some other media for long term storage.

The MEASUREMENT INTERFACE

The second feature of MPE that HPTREND uses to capture performance data is called the "Measurement Interface". This interface is actually embedded pieces of code in MPE that can be "turned on" to collect statistics on system utilization. This interface was added in order to support HP Performance tools like OPT/3000 but also allows other tools to use it. HPTREND activates this feature then captures data from it for trend analysis.

The advantage to using the Measurement Interface is that it can capture data on a more timely basis, and that it can actually break out MPE resource usage from that used by applications. The disadvantage to this feature is that it requires a special data acquisition program to be running on the system in order to capture the data. There is currently no automatic logging ability as is found in the MPE Log Files. HPTREND recognizes the need to obtain more timely data (in order to examine prime shift vs off shift usages etc.) and utilizes the Measurement Interface for this purpose.

The data acquisition program is designed to use very few system resources in the collection of its data. In fact, you should notice less that 1 percent additional CPU being used with both the measurement interface AND MPE Log file data being taken. Special steps have been taken to minimize the impact on ALL your system resources (disc I/O, disc space, main memory etc.).

You don't need to make any configuration changes in order to utilize the measurement interface like you did for MPE Logging. All you will have to do is to arrange a way for the HPTREND program to remain running (and collecting data) on your system through out the day.

THE COLLECTION PROGRAM

The HPTrend program that is kept running on your system all the time is the data collection program. This program will enable the Measurement Interface and every 5 minutes, it will sample the data. These samples are summarized and once every hour the summarized data will be written to an HPTrend data file. Once per day, the collection program will spawn a son process that will collect data from the MPE Log files and collect disc free space information. This program will read through all log files generated since it was last run and update several HPTrend data files. The program will also collect disc free space information by running the FREE#.PUB.SYS program. This process will usually only take a few minutes and will run at midnight by default so the impact on other system activity will be minimal.

INTERPRETING THE REPORT

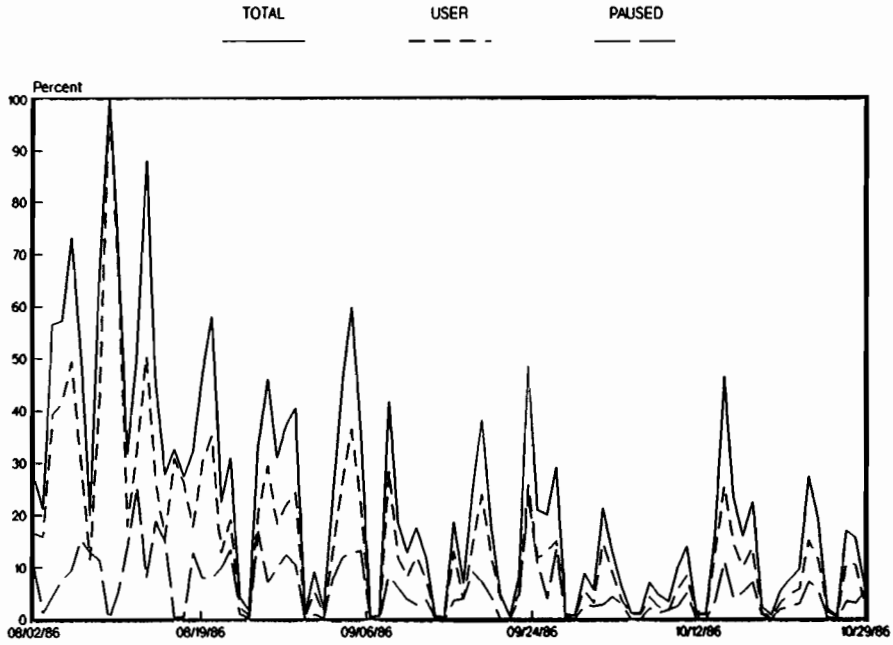
Happy day, my first HPTREND REPORT just arrived and I'm eagerly scanning through it to see what it can tell me. First, a word of caution, no matter WHAT you see, don't do anything about it

today. Let the report sink in a bit and try to relate what it says to what was going on while the data was begin taken.

What we will do next is to give a little bit of very general guidelines, much like old "country store advice" on what different things might mean. I'll say right up front that this "arm chair quarterbacking" for performance measurements can be right tricky stuff. If you disagree with any advice or think you might have something else going on then please get with your local performance specialist and check it out. We must necessarily stick to generalities and can not consider all the specifics of YOUR system. Never the less, we might be able to suggest a way to start looking at the graphs in order to get you going...

A general observation on performance analysis is probably in order at this point. "If you make a decision based on a single observation then you will undoubtedly be wrong"! There are many things that can cause a particular effect (such as high CPU usage) and to make an assumption as to cause without corroborating evidence will rarely yield correct results. You should examine all the effects on the system before you attempt to make conclusions.

DAILY CPU UTILIZATION



This graph is taken from the Measurement Interface data and shows day by day usage of the CPU in three different categories.

USER CPU is the percentage of time that the CPU was actually involved in running an application program. It includes the time spent in MPE code on the application's behalf (like the time the file system spent opening a file) but it does not include general housekeeping activities like memory management, spooling, and disc caching.

TOTAL CPU is actually obtained by taking the amount of time the CPU was IDLE (not performing any function) and subtracting it from 100 percent. This value includes the USER CPU plus the system housekeeping functions.

IDLE CPU is the time the CPU was no doing any work.

By comparing the difference between the TOTAL and USER graphs you can see how much time is assigned to system functions. These system functions are not resources used exclusively for maintaining the operating system environment. Many of these

functions, especially MPE Disc Caching, are functions performed at a users requests to improve the efficiency of the system.

INTERPRETATION GUIDELINES:

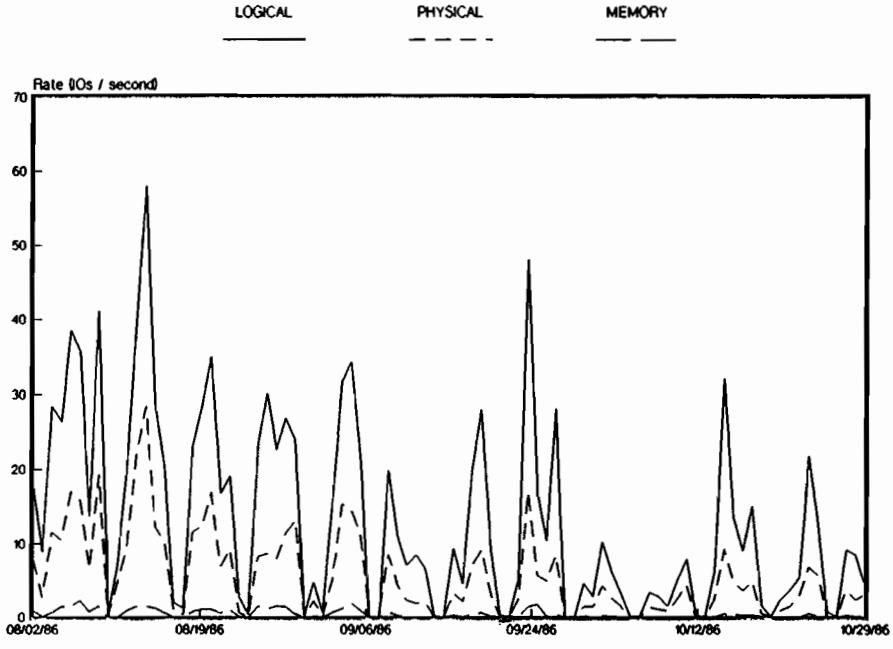
First off, be aware that the percentages shown are AVERAGES over a twenty four hour day. If you ran just a single eight hour shift and kept the CPU busy 100% of the time then this graph would show only 33% CPU used (8 hours out of 24). You can use later graphs to determine which hours of the day have most of the system usage.

Use this graph to determine the overall trend of CPU usage. Is it going UP? Down? Staying constant? Did something happen to change the graph on a certain date? What was it? You might see the effect on the CPU when you added a new application to the system or altered your system configuration.

If you attempt to do any forecasting of when you might expect to "run out" of CPU, then make sure you factor in the time of the day that the CPU is being used. (You may be 100% busy at noon but 0% busy at midnight).

System overhead (the difference between the TOTAL and USER curves) can be affected by many things. Notably memory management and disc caching can cause system overhead to grow. A good general guideline is that it is OK for system overhead to be from 5-15% on a system with sufficient memory and no disc caching. Disc caching can raise system overhead by 10-20% (for a total of 15-35%) without cause for concern. (The idea of disc caching is to trade off additional use of CPU and main memory in order to obtain higher disc I/O rates).

DAILY DISC I/O UTILIZATION



Like the previous graph, this one is taken from the measurement interface and shows trends as a function of date. It shows the activity on the system's disc drives. Please note that these graphs show disc I/O RATES and not CAPACITIES. They in no way reflect how much data is on your discs, rather they show how often the system had to transfer data to or from them.

LOGICAL RATE is the number of disc transfers (I/O's) per second that the I/O system has requested. If MPE Disc Caching is in use on your system, then some of these requests will be satisfied by Caching and will not require a physical access to the disc. If MPE Disc Caching is not in use on your machine, the Logical and Physical lines on this graph will the identical. This line represents an AVERAGE rate, much like CPU percent was. It averages all the disc I/O requests over all disc drives over the 24 hour day. To show 10 I/O's per second the system would have to have done (10 *60 sec/minute * 60 min/hr *24 hr/day) or 864,000 I/O's throughout the day. It could have done this evenly or, more often than not, unevenly. That means the discs might have been going 50 I/O's per second for part of the day then not doing any I/O's at all for other parts.

What good is a number like the AVERAGE I/O RATE then? It can be used to examine the TREND of demands on your disc drives. Again, ask yourself, "Am I using my disc drives more or less lately?" "Is my usage stable or are their peak times (like end of month)?"

As far as the absolute numbers on this graph. You can perform the exercise to see what potential you have left for disc I/O's but be careful that this data is insensitive to short time periods. You can be totally exhausted on disc I/O's for half the day, not use them at all the rest of the day and this report will say you have 50% of your disc I/O's left over.

PHYSICAL RATE is the number of disc transfers (I/O's) per second that were actually performed. If you don't have disc caching on your system, then this number should be the same as the LOGICAL RATE. With disc caching, it is possible for the I/O system to request an I/O (a LOGICAL I/O) then have disc caching satisfy it directly from memory cache and not have to do the real (PHYSICAL) I/O at all. For this reason, systems with disc caching should show a physical I/O rate that is lower than their Logical I/O rate. The difference between these two curves is the number of disc I/O's per second that disc caching has bought you.

MPE Disc Caching has recently been complemented with the introduction of Disc Controller Cache in the 7933/35 XP and 7937 XP disc drives. At this time, HPTrend does not collect any data concerning Disc Controller Cache and therefore does not report any information concerning Disc Controller Cache performance.

MEMORY RATE is the rate at which MPE's memory manager has had to swap a program's code or data segments to and from disc. It will have to do this for one of two reasons.

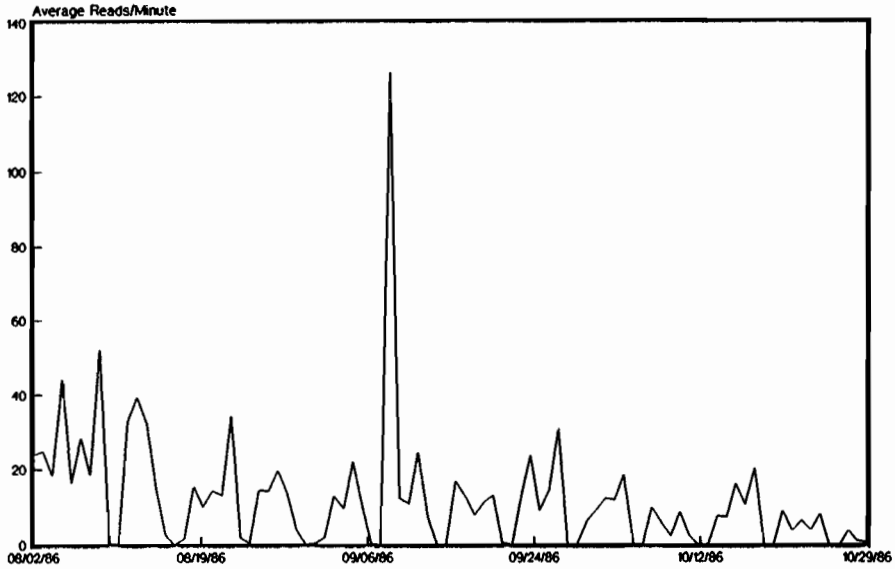
1) A program is initially run. In this case, it's segments must be brought from disc into main memory in order for it to execute.

2) A program has been running but something else requires main memory. If there is insufficient main memory to satisfy their request then the first program's segments may be "swapped out" of main memory to make room. Later, in order for the first program to execute again, these segments must be "swapped back in" to main memory. If there are more demands on main memory than it can efficiently handle, then this "swapping" can consume a significant portion of the disc I/O rate, not to mention a lot of CPU also.

Some swapping is inevitable and not necessarily bad. Excessive amounts can indicate that additional main memory (or reduced requirements on it) could increase system performance. What is excessive? A general guideline is that if the Memory manager I/O rate is greater than 5 I/O's per second and it is also greater than 10% of the total disc I/O rate, then it might be excessive.

What can you do to reduce the Memory Manager I/O rate? First, you should try to minimize the requirements for main memory. Programs with overly large stacks or code segments or files with very large buffers can consume large amounts of memory. Disc caching adds an additional burden on main memory usage. Have you allowed enough memory size to support it? If you can't reduce the demands placed on main memory, perhaps you should consider increasing memory size.

DAILY TERMINAL ACTIVITY



This graph shows the daily average number of terminal reads. Notice that the vertical axis is an average number of terminal reads per minute. In order for HPTrend to report an average of 50 terminal reads per minute for a particular day, the system would have to have done 72,000 terminal reads during the day. Like the disc I/O's, most of this activity would have likely been done during the hours of 8:00 am to 5:00 pm when the rate would have likely been higher than 50 terminals reads per minute.

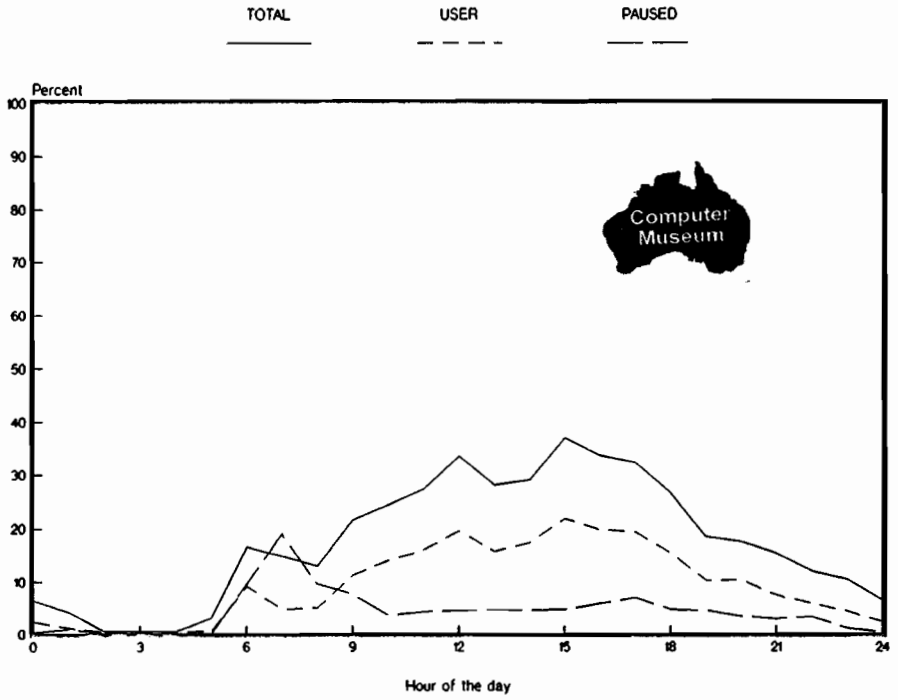
As with the previous graphs, this graph should be used to identify long term trends. Am I doing more or less terminal reads lately? Application changes such as moving from block mode to character mode can drastically affect this graph.

TRANSACTIONS

A transaction is a unit of work defined by each particular program to execute a task. A transaction is usually defined to be the time between the completion of a terminal read and the next prompt for the next terminal read so that a terminal read is the same as a transaction. This is true from the perspective of

the user but often times there are terminal requests such as status requests which are satisfied without user input. These requests are also counted as terminal reads by the computer measurement hardware/software. HPTrend does not attempt to filter these reads from the total terminal read count so the terminal read values reported here will be close to but usually greater than the transaction rate.

HOURLY CPU UTILIZATION



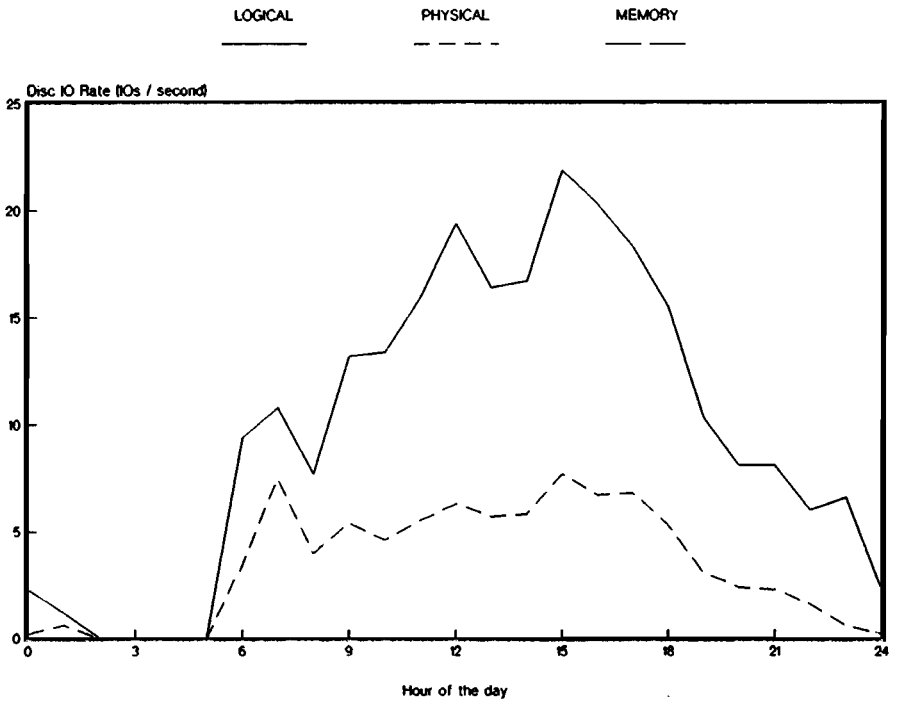
This graph shows CPU usage again. The curves have the same meaning as the first graph. The horizontal scale, however, is different. While the first graph showed CPU usage day by day, this one shows it as a function of the hour of the day.

The X-Axis shows the hour of the day in 24 hour time. The curves show the average CPU usage for the hour for all WEEKDAYS (Monday through Friday but not Saturday or Sunday) that occurred during the measurement interval. Week ends were excluded since they usually have a different type of usage than week days on most systems.

You may use this graph to see what hours of the day correspond to the peak and slack usages of the CPU. Consider whether you can move some activity from your peak times to your slack times and smooth out your use of the system. (Moving batch jobs to the evening shift can often help day shift performance considerably). You can also judge from this graph how to adjust the scales on the previous day by day graphs. If you see that your system usage starts at 8:00 AM each day, remains constant until 12:00 noon,

drops until 1:00 PM then continues until 5:00 PM (a typical day shift) then you can assume that you are only using the system for one third of the day (8 hours out of 24). If the day by day graphs show CPU at 33% or more then you can assume you were at 100% during the day shift. This is a bit simplified, but I think you get the idea.

HOURLY DISC I/O RATES



This is the hour of the day graph for disc I/O rates. It too excludes weekends from its calculations. this graph combined with the previous one can be used to adjust your system to different types of loads.

If you run highly interactive activities during the day shift and change to batch work at night, then you may be able to "tune" your system differently to adapt it to the different work loads. Here is an example:

Suppose you see that during the day the system CPU overhead is about 25% but that less than 100% of the CPU is normally used. (The CPU is an "all or nothing" sort of resource, having "just enough" is as good as having "lots of extra". Once you're out, though, you're out!) You also notice that during this time the difference between logical and physical disc I/O's is significant and that memory management disc I/O's are low. You can assume that disc caching is working well for you at this time. You are using additional CPU but you still have enough left over. Cachings' use of main memory is appropriate since it is not

forcing the memory manager to swap program segments at too high a rate. And lastly, caching is buying you something in that it reduces the number of physical I/O's necessary to meet the need.

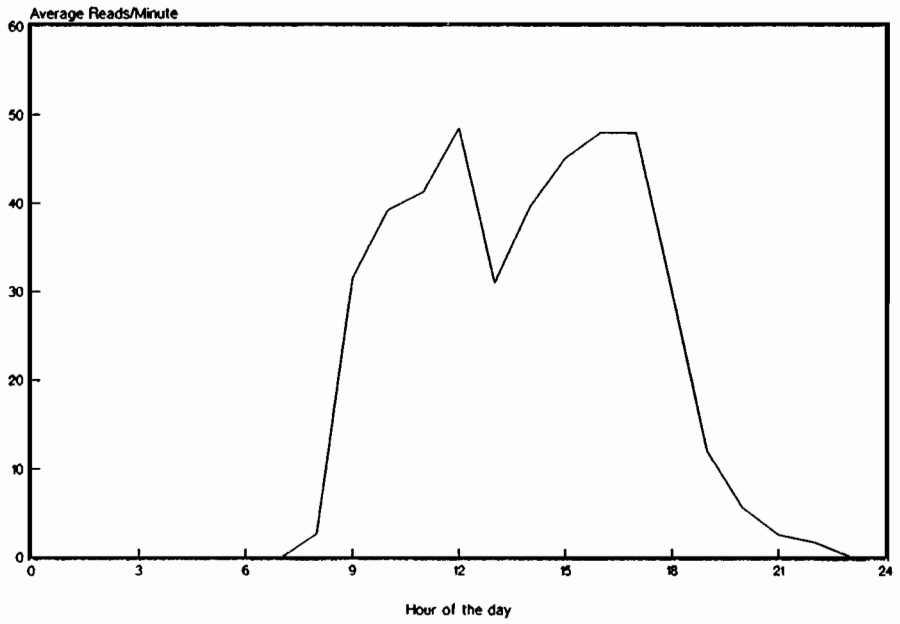
Next suppose you notice a change once you start your nightly batch work. Maybe the CPU usage goes to 100% with system overhead taking 30-40% and users getting the rest. Also suppose you notice memory manager I/O's going up and the difference between logical and physical disc I/O's shrinking. This would be an indication that the nightly batch work is not being helped as much by caching as the daily work was. In fact, it might be that you could gain performance during the night work by turning caching OFF!!

You could test this theory by turning caching off at night for a while and watching the results. (First see if the nightly work gets done earlier or later. This will be their REAL performance measurement). Don't settle for one nights work but rather compare data over several nights with caching on then off to see the real results. If this works, then you should notice some changes in your next HPTREND report. Day shift should be the same but night shift should change as follows; The logical and physical I/O rates should be the same (no caching). Hopefully the memory manager I/O rate should be lower. (without caching using memory, more program segments can remain in memory). The system overhead CPU should drop so that users are getting more than the 60-70% they used to get.

If you don't get the desired results then you made a mistake somewhere along the line. (Don't get upset, this still is more trial and error than science and overall system performance is a complex beast). Try to figure out where you went wrong and see if there is something else you could do to help performance such as tuning the disc caching quantum.

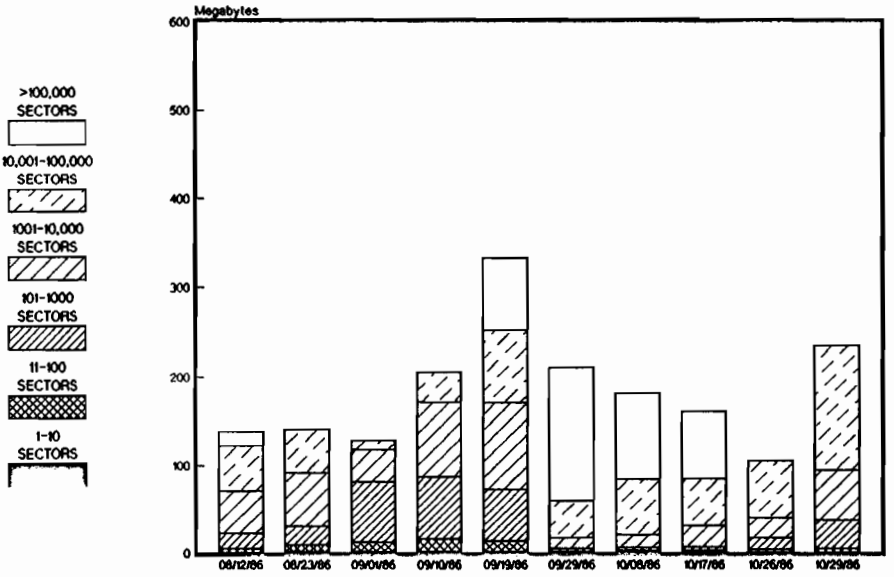
This technique can be applied to all the other data in this report. The example above is fictitious but based on sound theories. The important thing to remember is to try to understand the data you have, formulate a theory that should achieve the desired results, test the theory by implementing it under carefully monitored conditions, evaluate the results, then go back to step one and do it again.

HOURLY TERMINAL ACTIVITY



This graph again shows terminal activity with the vertical axis having the same scale. The horizontal scale is different showing the hour of the day rather than an average for each day. As with the previous two graphs, weekends (Saturday and Sunday) are excluded since they usually have different trends in terminal usage.

DISC FREE SPACE

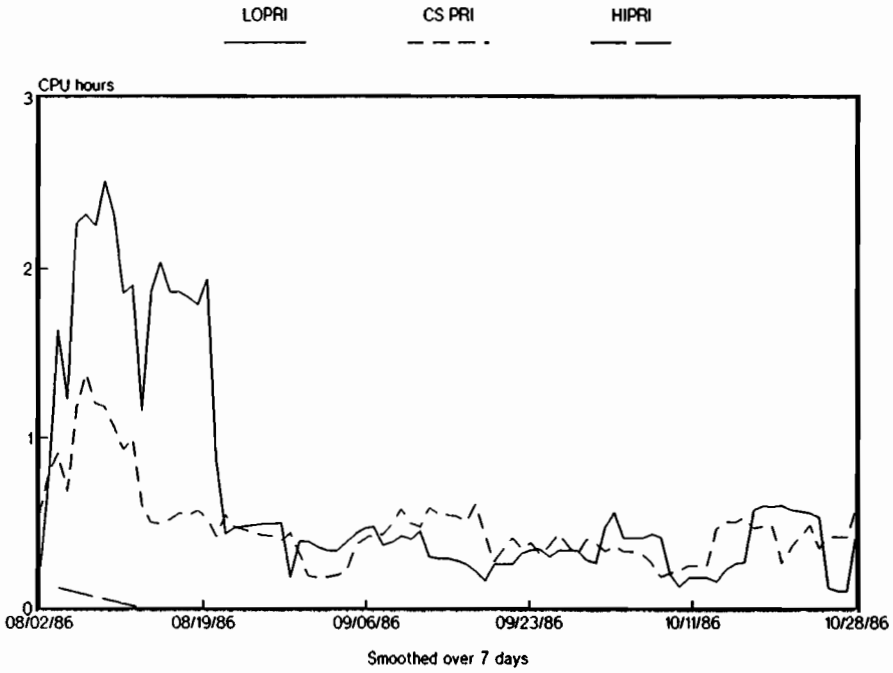


This graph shows an average of available disc free space over time. The data comes from a FREE#.PUB.SYS output that is done daily on your system. The graph breaks down the available free space by the size of contiguous free space. Use this graph to identify trends in disc space usage. The size of contiguous areas of free space is very important on this graph. Files on the HP3000 are partitioned into a variable number of chunks of contiguous disc space called extents. Each file can have from 1 to 32 extents but only the extents in use must be allocated space on the disc. When you ask for a file to be built or when to add enough records to a file with more than one extent and another extent is necessary to add the new records, an area of contiguous disc space must be found and allocated to the file before the records can be written. It is possible to have enough total disc space available to build the extent but not have enough contiguous disc space to allocate to this extent. If you have 50,000 sectors of free space split into 10 areas of 5,000 sectors each and get a request to allocate an extent that is only 6,000 sectors, the request would fail and the error returned would be out of disc space.

There are several things you can do to increase the size of contiguous disc free space on your system. The VINIT subsystem

which comes with each HP3000 has a CONDence command which will attempt to concatenate disc space on a disc by disc basis. A second option for bringing free space together is the Recover Lost Disc Space option of a Coldload. This option will take more time than the VINIT CONDence option but will also be a much better job of concatenating free space on all system domain discs. Another option is to schedule regular system reloads.

USER CPU BY EXECUTION PRIORITY



Now we get into those graphs produced by the MPE Log File daily information. As opposed to the ACCOUNT information, this data is system wide and organized day by day. It is primarily user data and does not usually contain MPE overhead. Remember the discussion on when MPE Log Files record their data? They log all the information about an event at the END of that event. In this case, the CPU used by a job or session is logged when that job logs off the system. It is possible for jobs or sessions that ran greater than one day in length to accumulate more than 24 hours of CPU time. All this time would be logged at the time they terminate and would appear as an impossible situation in these next graphs. (How can you have more than 24 hours of CPU used in a 24 hour day??) If you have such jobs or sessions running on your system then you will have to make allowances for them in these graphs.

The first graph breaks the CPU used into categories based on their logon job or session priority. LOPRI is any job or session that logged on with ;PRI=DS or ;PRI=ES. CS PRI is any job or session that logged on with ;PRI=CS. HIPRI (if present) is any job or session that logged on with ;PRI=BS. Note, no allowances

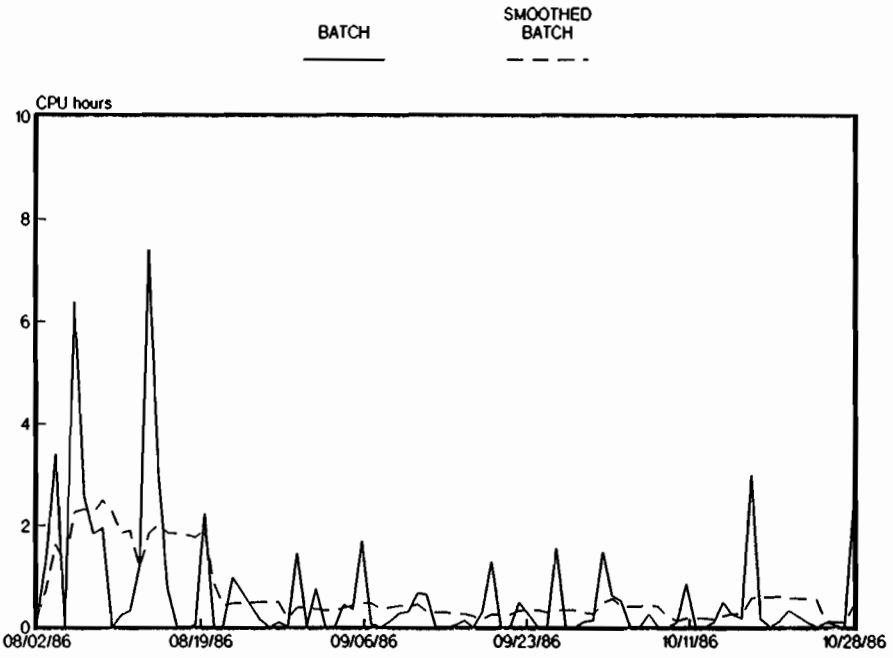
are made for processes that change their priority once they are running. All CPU is accumulated based on the priority the job or session initially logged on with. HIPRI data will not be graphed unless it is greater than zero.

A NOTE ON SMOOTHING:

This graph is the first to have a smoothing function applied to it. This is a "running average" calculation applied to the data before it is graphed. This smoothing tends to make the graph less "jumpy" and can make it easier to spot longer time trends. The problem with smoothing is that it tends to flatten the peaks and valleys from a graph. This means that it hides the true maximum and minimum values that the original data covered. Smoothing also tends to "fatten" a short duration "spike" in the data. For instance, if the CPU usage was normally 6 hours a day but then jumped to 24 hours for one day, then a seven day smoothed graph would look like a seven day wide "bump" of 8.5 hours CPU used.

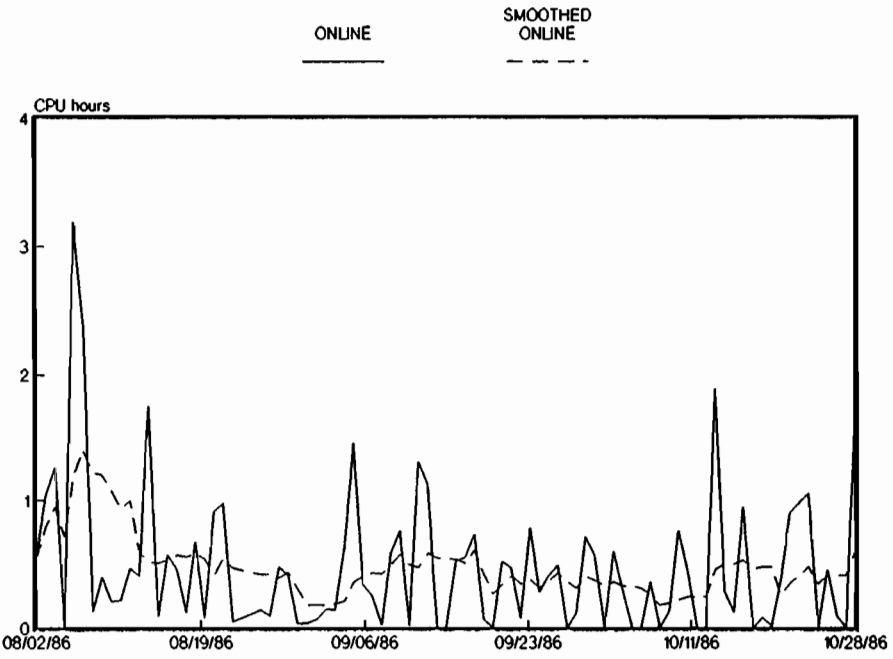
It is especially important on smoothed graphs to use them ONLY for general long term trend forecasts. Go back to an unsmoothed graph to see actual minimum and maximum usages for a resource. Smoothing becomes very nice when you want to see the usage trends over long periods. Without it the graphs become hopelessly jumbled and difficult to read.

CPU USED BY BATCH JOBS



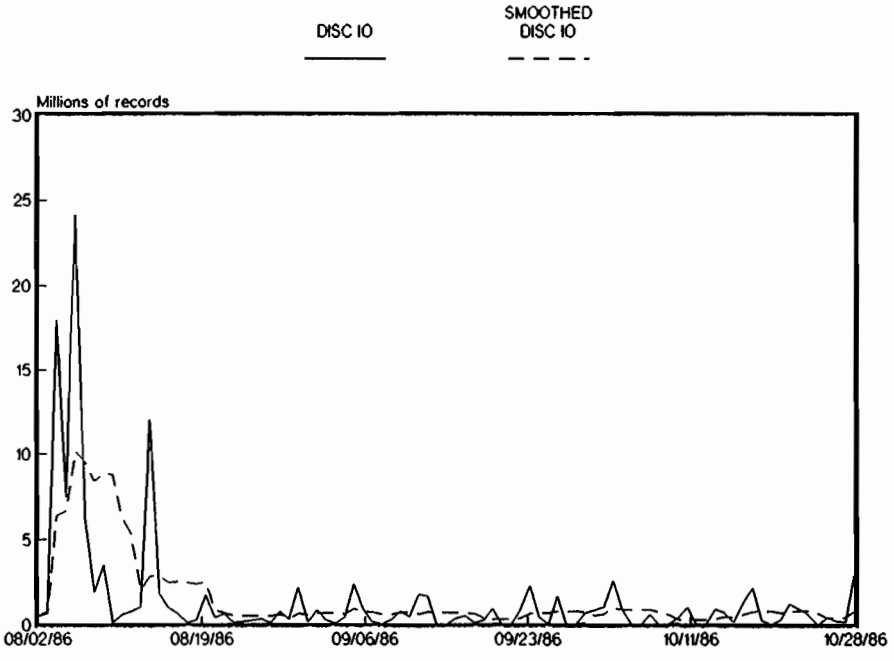
This graph shows both the smoothed and unsmoothed curves for Batch Job CPU usage. It is a good one to see the effect of smoothing. Use the unsmoothed graph for minimum and maximum data and the smoothed graph for trend information. The CPU included in this graph is for any batch job, regardless of its execution priority.

CPU USED BY ONLINE TERMINALS



This graph shows both the smoothed and unsmoothed curves for Session CPU usage regardless of execution priority. Note long term trends for your online users.....

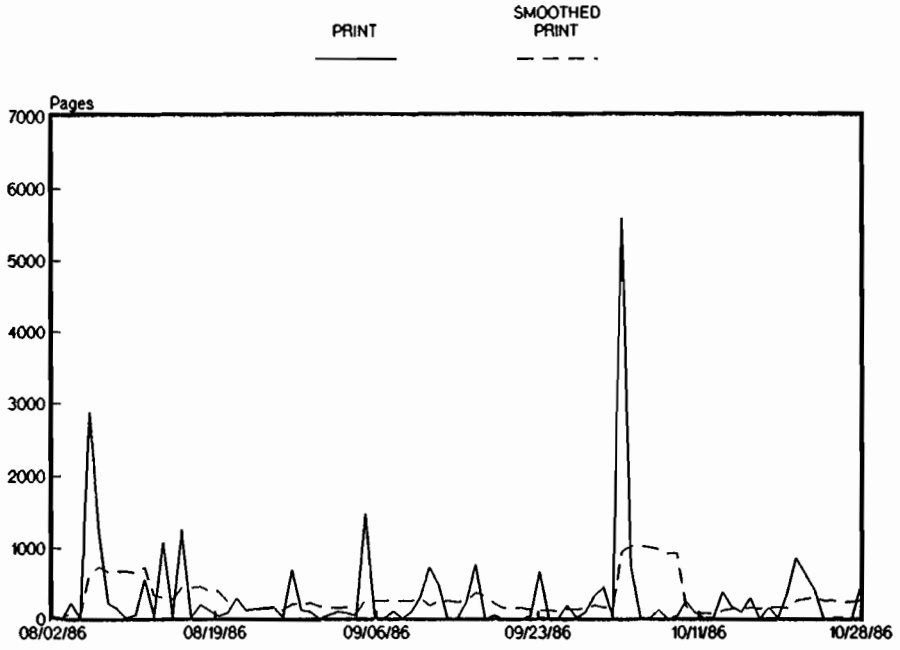
USER DISC I/O's



This graph shows day by day disc blocks transferred by user processes. Note that this is BLOCKS transferred and usually relates to logical disc I/O's. Also note that MPE disc I/O's are not normally included in this figure (other than the LOADER and Terminal Type file I/O's).

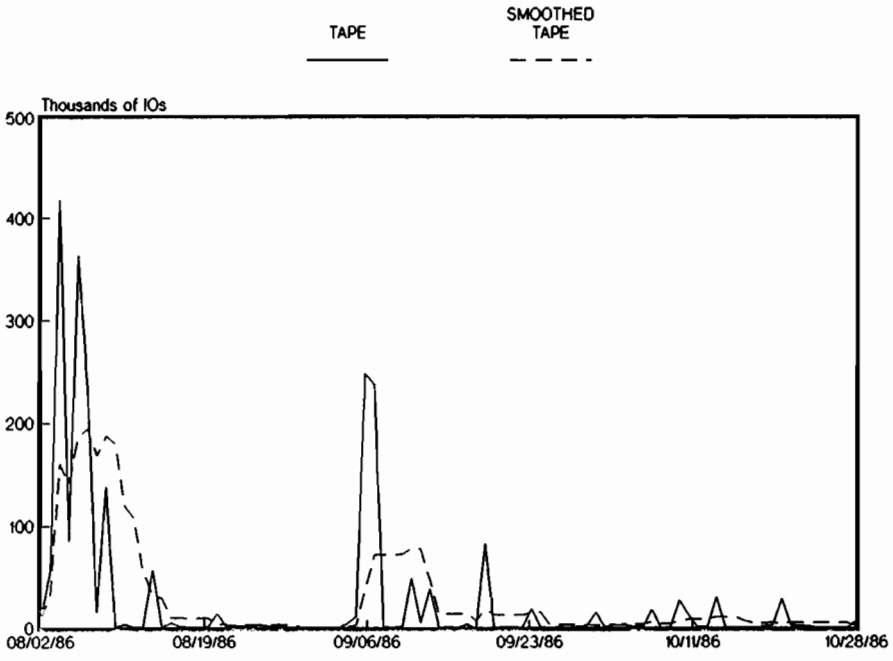
Also notice that the Y-AXIS scale on this graph is in millions of records transferred whereas the Measurement Interface graph was in terms of I/O's per second. As long as you are looking for trends this will not matter, but any attempt to correlate the two graphs must include a conversion based on the average system usage interval.

PRINTER OUTPUT



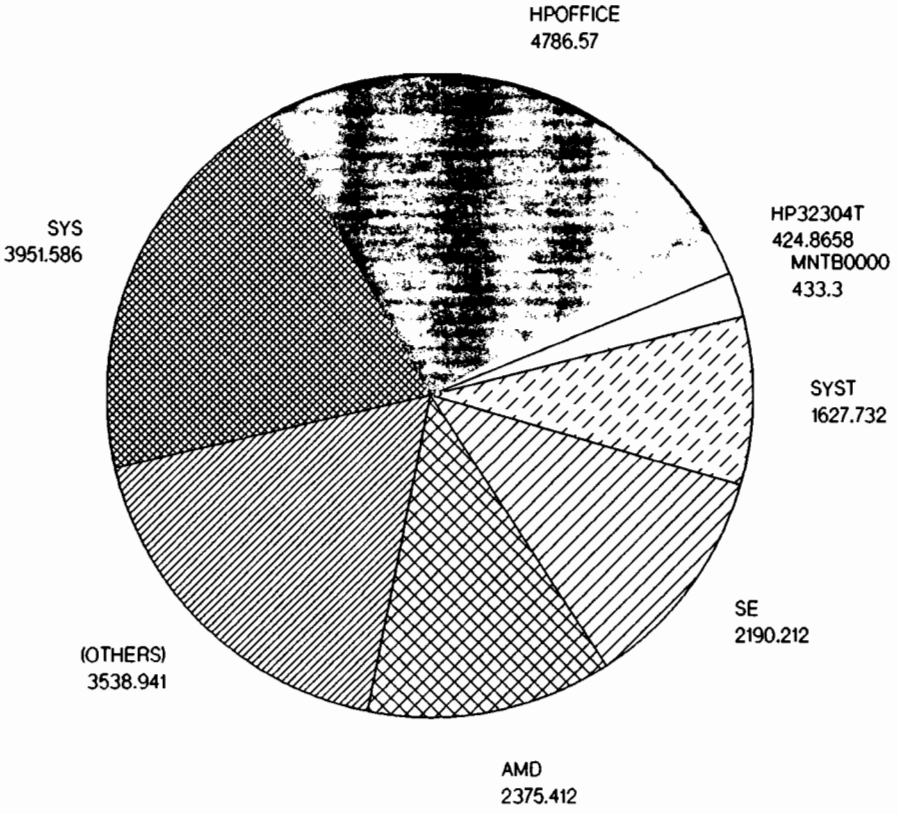
This graph shows day by day how many pages were printed to the spooled printers on your system. Pages are actual physical pages for the newer HP printers or lines divided by 60 for the older printers that do not report pages. Unspooled printers are not included in this graph.

MAGNETIC TAPE USAGE



This graph shows the day by day usage of all magnetic tape drives on your system (not counting the cartridge tapes like the 9140 and 9144 which are included in the DISC I/O graphs). The Y-AXIS is in thousands of I/O's or magnetic tape BLOCKS. Each block may contain one or more logical records.

CONNECT HOURS



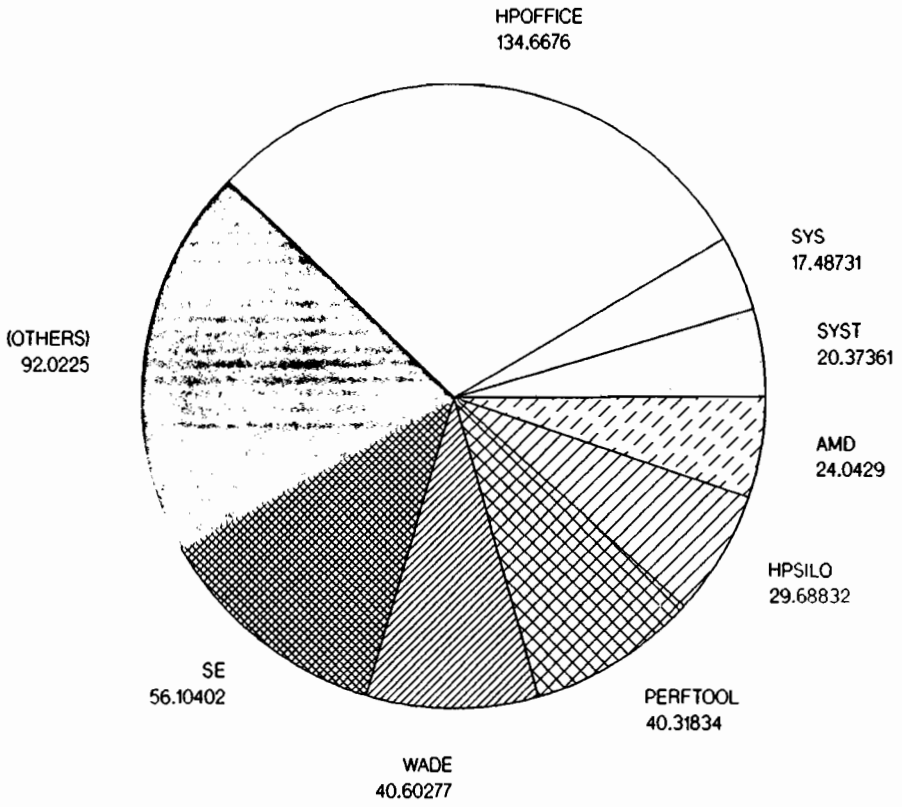
The next series of graphs in the HPTREND report come from the MPE Log Files. This is the ACCOUNT information mentioned earlier. The first few graphs are pie charts showing the names of the accounts used to accumulate the majority of each resource. The top ten accounts will be shown (unless an account used less that two percent of the total resource then it will be suppressed). You may also see a category labeled "(OTHERS)". This category is the combination of all other accounts not shown on the graph.

The pie represents 100% of the resource USED on the system. There is no correlation as to how much of the resource was actually available. Thus the first graph (connect time) shows 100% of the

actual connect time used and how much each account was responsible for. Note that "connect time" is actual log on time and includes batch jobs as well as sessions.

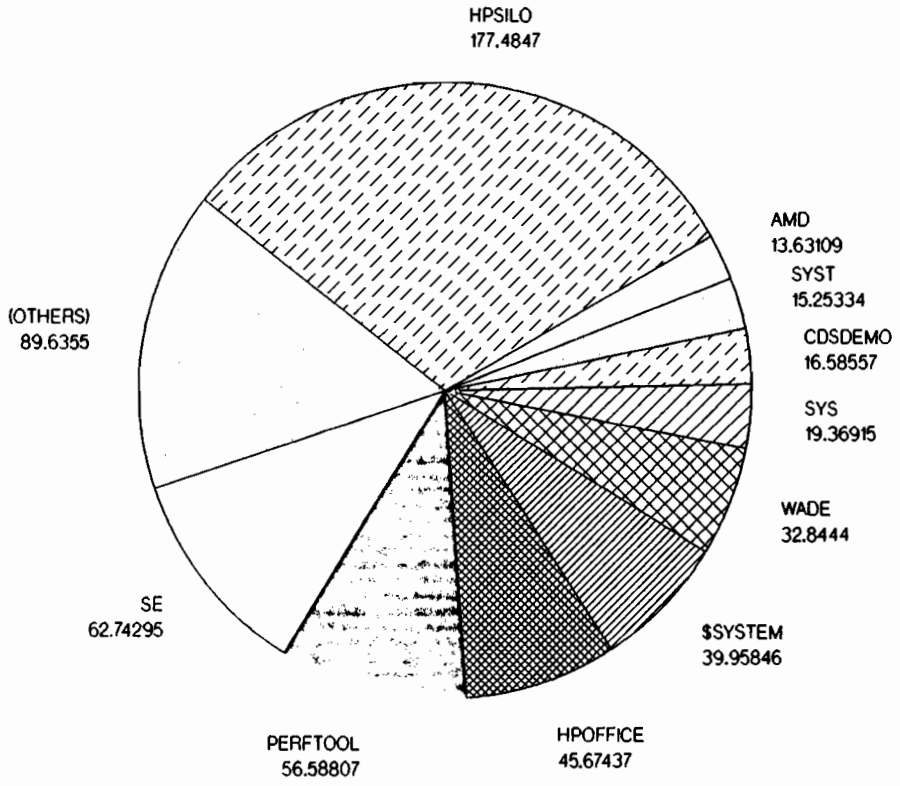
What do you use these graphs for? You might be able to correlate how best to allocate your critical resources if you can tell which accounts are using them. If a resource is not critical then just skip over the corresponding graph. (If you determine your system performance is limited by disc I/O's then you don't need to be too concerned with which accounts used the CPU most).

CPU HOURS



This graph shows the accounts that used the majority of the CPU on the system. Note that MPE overhead is NOT included in this graph. The units displayed on the graph for each account is in hours. In the example above, the WADE account used 40.6 hours of CPU time during the collection period.

MILLIONS of DISC I/O's

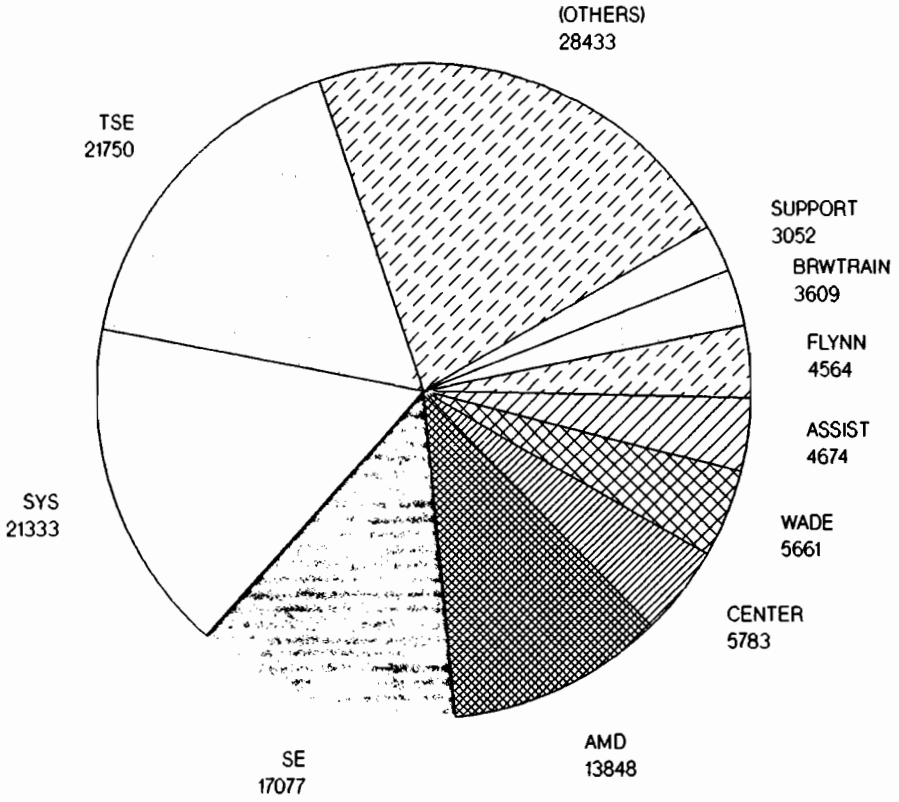


This graph shows the top ten accounts that did disc file transfers. The numbers are actually millions of disc I/O's. Thus 32.84 means 32,840,000 disc transfers. You may also see a category called "$SYSTEM" in this graph. This is not actually an account on your system but rather is file transfers done by MPE itself. The main reason these I/O's are logged to the MPE Log files is due to the LOADER process reading a program file when you RUN it and then reading the SL files in order to link the program for execution. Also in this category is the terminal driver reading the Terminal Type files each time a terminal logs onto the system. If the numbers for $SYSTEM are high then you might look further to determine the cause. Perhaps allocating an often run program might help, or you might find a noisy terminal line which causes the Term Type file to be continually reopened.

What kind of I/O's are shown on this graph, LOGICAL or PHYSICAL? Well, it's probably closer to LOGICAL but actually it is neither one. It is the number of BLOCKS transferred to the file. Normally there is one logical I/O per block but special programs using MULTIRECORD I/O can read many blocks at one time. In this case the number of blocks transferred is neither logical or physical. Usually this is rare so you can count the I/O's as logical. Actually you'd be better off to ignore the absolute numbers and concentrate on the relationships between accounts.

With the UMIT and later releases of MPE, IMAGE has been succeeded by TurboIMAGE. One of the performance enhancements added to TurboIMAGE is the use of Global File Opens. A global file open is a concept where the first time a file is opened, the information is stored in a place where all processes that require access to that file later can use the file without opening the file. The eliminates the very expensive file open operation every time except the first time. The file is not closed until the last process accessing the file closes the file. This is an excellent performance enhancement that has boosted the performance of most HP3000's using TurboIMAGE. Since the MPE Log files only record activity against a file when the file is closed, activity is only reported against TurboIMAGE data bases for the process which closes the file last. This has an impact on this graph when users from multiple accounts are accessing a TurboIMAGE data base. When this is the case, the above I/O counts for a particular account may be inflated because of a user in this account happened to be the last user to close a TurboIMAGE data base. Another account will be reporting fewer I/O's that were actually performed because users in this account were not the last user to close a TurboIMAGE data base.

PAGES PRINTED

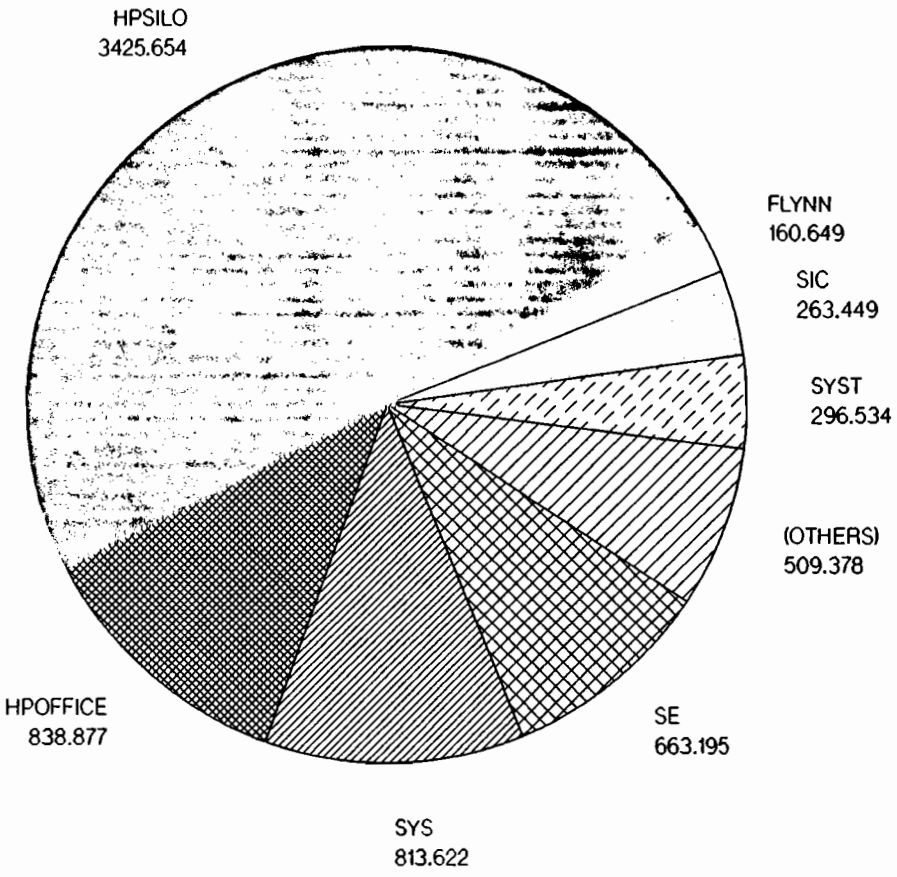


This graph shows the accounts that printed the most pages to spooled printers during the collection period. It includes only spooled printers (including remote spooled printers) but not unspooled printers. Some of HP's newer printers report the number of physical pages printed back to MPE. (The HP2680 for instance can print more than one logical page to a single physical page but it notifies MPE of how many physical pages were actually printed). If this information is available in the log files then it is used. Other printers (2631B, 2617 etc.) do NOT report the number of pages printed but rather only the of lines printed. HPTREND arbitrarily assumes that you are printing 60 lines per

page and calculates pages as lines/60. All printers are combined on this one graph.

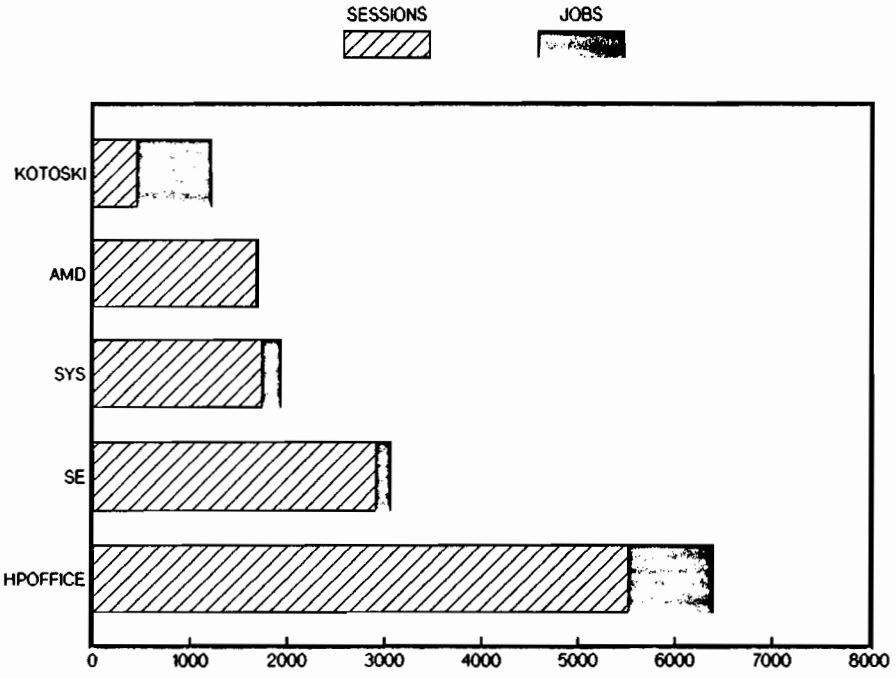
It is possible to get inaccurate results when you let a spool file start printing then do a DELETESPOOLFILE before it finishes. HPTREND has no way of knowing that the spool file was not printed in its entirety and will report it on this graph as if it had completed printing.

MAGNETIC TAPE BLOCK I/O's



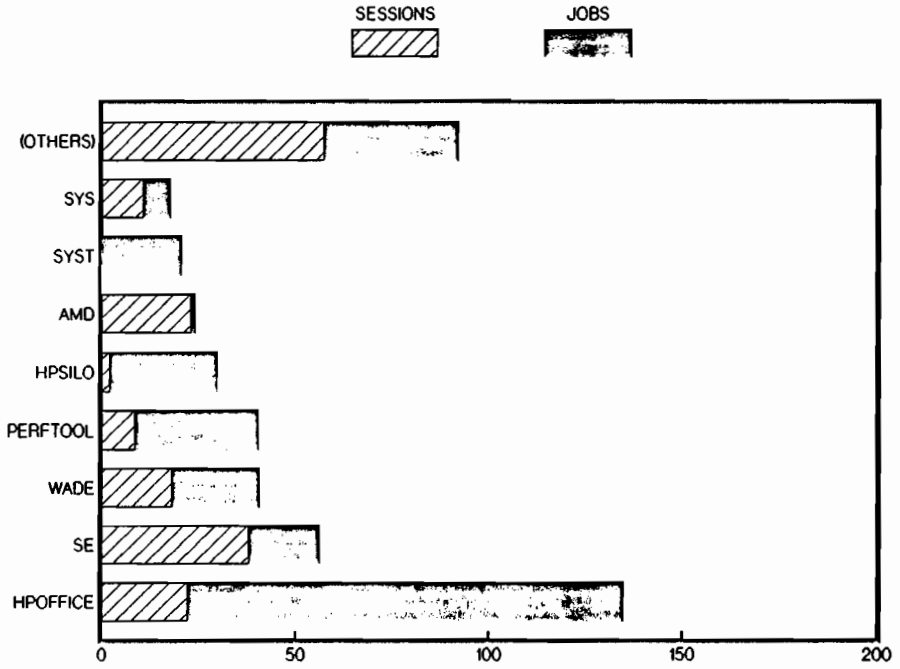
This graph shows the accounts that transmitted the most blocks to or from tape devices during the collection period. The numbers in this graph are thousands of magnetic tape blocks. A block is the smallest amount of data read or written to a tape drive and normally results in a single I/O transfer. It is possible to write multiple logical records in each tape block so the actual number of logical records written to the tape may be higher than the number of blocks reported here.

JOB & SESSION LOGONS BY ACCOUNT



This graph shows the number of logons for the top ten accounts. The number of jobs is stacked on top of the number of sessions so you can see the total number of logons for the account. You will also be able to judge visually the proportion of batch to interactive jobs for each account.

CPU HOURS by ACCOUNT



This bar chart shows CPU time by account again (like the second PIE chart) only this time the CPU used by Batch Jobs is separated from that used by Sessions. You can visually scan the various accounts for their relative proportions of CPU usage in batch vs interactive sessions.

HP-TO-IBM COMMUNICATIONS

RAPHAEL CARTY
HEWLETT-PACKARD
CUPERTINO, CALIFORNIA

INTRODUCTION

IBM connectivity is an important element of Hewlett-Packard's HP AdvanceNet strategy for multivendor networking; accordingly, the company offers a full line of products that integrate HP computers with IBM systems. These products allow subnetworks of HP distributed systems to coexist with IBM systems at corporate or divisional headquarters locations.

A strong commitment to multivendor networking based on international and de facto standards, such as SNA, is central to the HP AdvanceNet strategy.

HP-TO-IBM BATCH AND INTERACTIVE PRODUCTS

HP provides batch and interactive SNA communications on the HP 3000 family of business computers.

The HP 3000 batch communications product, SNA Network Remote Job Entry (NRJE), enables an HP 3000 to appear as an 8100 DPPX/RJE Workstation to an IBM host, when used with HP's SNA Link product. Batch jobs are spooled when communications lines are busy, and automatically processed when the lines are free.

HP's SNA Interactive Mainframe Facility (IMF) is the company's interactive communications product. SNA IMF emulates the main features of an IBM 3270 control unit, using SNA PU2 and LU1, 2 and 3 protocols. HP terminals, printers and applications on the HP 3000 that are running SNA IMF emulate IBM terminal and printer functions.

The SNA3270 Link on the HP Vectra PC and the HP Portable Plus PC provides interactive access to IBM 3270 applications by emulating an IBM 3274 cluster control unit, which is a

PU2 device. This product also provides PC file transfer. In addition, HP supports IBM 3278 emulation via an IRMA card coaxial cable attachment to the IBM 3274 cluster controller for the Vectra PC.

Thus users can enter transactions on the HP 3000, send information to the host's central database, and retrieve information from that database.

HP additionally offers bisynchronous (BSC) HP-to-IBM products, including: 2780/3780 (RJE) batch workstation emulation, HASP workstation emulation, and 3270 terminal emulation.

HP-TO-IBM IN THE OFFICE

Hewlett-Packard offers connections to IBM's PROFS and DISOSS, as well as support of IBM's Document Content Architecture (DCA), Document Interchange Architecture (DIA) and Logical Unit Type 6.2 (LU 6.2).

HP OfficeConnect to DISOSS enables HP DeskManager users to exchange electronic mail with DISOSS users, and to file, search for, retrieve and delete documents in a DISOSS Library, using Final Form Text DCA in most cases. The product also includes a converter that permits exchange of documents with DISOSS in Revisable Form Text DCA.

HP OfficeConnect to PROFS enables HP DeskManager users to exchange electronic mail with PROFS users.

HP OfficeConnect products provide users with transparent access to PROFS and DISOSS hosts completely within the HP network. No software is required for the IBM system.

VERIFIED THIRD-PARTY SOLUTIONS

Hewlett-Packard has verified the performance of third-party solutions for reverse pass-through and SNA-to-X.25 conversion. Reverse pass-through performs the necessary

IBM 3270-to-HP terminal conversion needed to allow IBM 3270 users to access applications based on the HP 3000, such as HP DeskManager, MM, PM and others. Hewlett-Packard has verified products from the following vendors: Forest, Gandalf, Tymlabs, Grampian and Simware.

SNA/X.25 conversion allows the SNA HP-to-IBM communication products to be used over an X.25 network. Hewlett-Packard has verified the XGATE 650 SNA/X.25 protocol converter offered by XMIT A.G.

In addition to the HP 3000-to-IBM communication products described here, Hewlett-Packard offers a full line of IBM connectivity products on the HP 1000 and HP 9000 systems as well. For more information on these products for the Computer Integrated Manufacturing (CIM) and engineering/scientific environments, please contact your HP sales representative.

HP-TO-IBM: FUTURE DIRECTIONS

Hewlett-Packard currently has an entire laboratory dedicated to the development of new products that enable users to communicate within IBM environments; among these products are:

--an LU 6.2 application program interface (API) that enables third parties or end users to develop HP 3000 applications that use the LU 6.2 protocol to communicate with IBM systems.

--HP's software-only, enhanced reverse pass-through product;

--SNA-to-X.25 conversion software.

HP STARLAN SUPPORT OF THE IBM CABLING PLAN

An important HP objective is to make HP StarLAN, which is the company's strategic commercial LAN product for the office, useful in the widest possible range of office environments.

To meet this goal, Hewlett-Packard is testing HP StarLAN over IBM Cabling Plan Type 3 (unshielded twisted pair wiring, which is used by both StarLAN and Token Ring). HP will run tests with shielded twisted pair wiring (such as IBM Cabling Plan Types 1, 2 and 9) as customer need requires.

HP STARLAN COEXISTENCE WITH IBM TOKEN RING

Hewlett-Packard firmly believes that HP StarLAN offers several advantages over Token Ring, but we are also committed to protecting customers' investments in our own office solutions as well as those of IBM. To this end, HP plans to enable Vectra PCs that are linked to Token Ring Networks to access Personal Productivity Center (PPC) functionality on the HP 3000 through an 802.3/802.5 bridge. HP's Vectra PC third-party program will also include a Token Ring adapter card and software. In addition, HP Vectra PCs on the 802.3 network will be able to access systems on the Token Ring Network.

The advantages StarLAN offers to customers include a lower cost per connection than Token Ring. In addition, since HP StarLAN is designed for use over phone wire, customers can feel confident that StarLAN will work with fewer limitations and permit more flexible configurations over existing phone wire (which is unshielded twisted pair).

OSI AND SNA NETWORK MANAGEMENT COEXISTENCE

To provide coexistence between HP industry-standard networking and IBM's SNA networking, HP will support IBM's Network Management Architecture (NMA) on the company's SNA HP-to-IBM products. A distinction, however, should be made between NMA and NetView. NetView is an IBM product packaging and not an architecture. HP is investigating how it will implement support of NMA.

This does not necessarily imply that HP will support NetView/PC. Given the limited functionality currently defined in NetView PC coupled with the present limitations of the PC itself, we are not convinced that NetView PC will be able to provide our customers with

the type of network management capability they need. We believe that our recently introduced Private Packet Network (PPN) offerings can deliver more state-of-the-art functionality than NetView/PC offers.

HP's network management strategy has three phases. Phase 1 covers the variety of products that exist today, which are especially strong in telecommunications management.

Phase 2 includes products that will be introduced within the next year. These will integrate current single-system tools into multiple-system management products. Thus a system operator will be able to issue global commands to execute tools on multiple systems.

Phase 3 looks at the 2-to-5-year time frame. At present, HP has developed a network management architecture that parallels as closely as possible future ISO standards for network management. HP's architectured solution will provide applications that span all HP processors and multivendor environments.

HP-TO-IBM PRODUCTS UNDER INVESTIGATION

Hewlett-Packard is currently investigating development of a PU 2.1 capability, as well as a SNADS product, for the HP 3000.

Hewlett-Packard is firmly committed to making connectivity to IBM easy to implement and use for its customers. This is an important element of HP's commitment to multivendor networking based on both international and de facto industry standards.

Spectrum Case Study: Successful Migration to MPE XL

by
Michael A. Casteel
Unison Software
415 Clyde Avenue
Mountain View, CA 94303

Introduction

Unison Software is an independent software vendor supplying Data Center Management and Transaction Processing software for the HP 3000. In its eight years of existence, Unison has developed four major software products totalling some 150,000 lines of source code (see Figure 1).

Package	Language	Lines of code	
Maestro	SPL	65,000	*Priv Mode*
Radar	SPL	35,000	
Tapes	COBOL	15,000	
Insight	SPL	35,000	

Figure 1. Size and Language Distribution

When development began in 1979, we chose the preferred language for programming systems and utility software which was SPL, HP's proprietary high-level Systems Programming Language. This has proven to be a highly satisfactory choice through the many iterations of HP 3000 processors and many versions of MPE, culminating in MPE V/E.

However, the introduction of HP Precision Architecture machines begins a new age. The new machines implement a different instruction set, execute a new yet compatible operating system, and are provided with a set of "Native Mode" (NM) compilers, which translate programs to the new instruction set. SPL is not one of these compilers.

So as you might expect, Unison Software is deeply concerned with the issue of migration to HP Precision Architecture. We were privileged to begin dealing directly with this problem in early 1986, under the auspices of HP's Fast Start program. This presentation covers our early experiences with the migration effort, which I believe will offer encouragement to others contemplating migration to HP Precision Architecture.

Compatibility Mode Testing

Although the new line of computers is based on a completely new instruction set, HP promises a very high degree of compatibility with existing programs. Almost any existing HP 3000 program is supposed to execute, unchanged, in so-called "Compatibility Mode" (CM) just as it does on a traditional HP 3000. The new compilers are needed only to compile programs into NM for faster execution or to access new features only available in NM.

Given the degree of compatibility which was promised, our first migration step was to test all systems in CM, restoring the programs and data files from a standard installation tape onto a Series 930. We expected this step to test not only the success of HP's implementation of Compatibility Mode, but the viability of essential subsystems such as IMAGE, KSAM, VPLUS and MPE. It was really the crucial step in migration.

We were prepared for the worst. Although HP had an excellent record maintaining compatibility from the original MPE through MPE V, there had never been such a fundamental change in architecture. In addition, we had already been advised that the watchwords for Fast Start participants were patience and *flexibility*.

In fact, we were quite pleased with the results of early CM testing. MPE XL was not entirely bug-free, but all those critical MPE subsystems were fully operational and we successfully tested almost every function of our software by mid-1986 (Figure 2). The only missing element was networking, which was not available until later in the year, and the only notable difficulties encountered were with heavily used Message Files and some Privileged Mode code.

Subsystem	M*	T*	R*	I*	Comment
TurboIMAGE	•	•		•	M/I – Multiple DB's
KSAM Files	•			•	
Message Files	•		•		Heavily used
MPE Files	•	•	•	•	
Process Handling	•			•	CM and NM progs
VPLUS	•	•	•	•	Advanced functions
Terminal I/O			•	•	Bypassing VPLUS
SL Libraries			•		
Networking					
NS/LAN	•	•			
RFA	•				KSAM+Message files
RDBA		•			Also via RFA
Privileged Mode					
ABSOLUTE	•				PCB entries
MFDS/MTDS	•				JMAT / JIT
DIRECFIND	•				Logon passwords
GETSIR	•				JMAT SIR

* M=Maestro, T=Tapes, R=Radar, I=Insight

Figure 2. Subsystems/Functions Exercised

Surprisingly enough, only a few minor changes were required to restore the Privileged Mode code to proper operation. As insurance against further changes in MPE XL, we have since developed an alternative implementation of these functions which avoids access to system tables and undocumented intrinsics. And as you would expect, the Message File problems have been corrected.

Pick a Mode

Migration to NM is not an all-or-nothing proposition. Existing programs can execute on the machine in CM at the same time other programs are running in NM. Every time you run a program, the loader notices whether it is a traditional 'PROG' file or a new 'NMPRG' file and automatically starts it executing in the correct mode.

This means that you can pick and choose exactly which programs you need to convert to NM, and when. Clearly, you can feel comfortable leaving programs in compatibility mode which run every now and then and only for a few minutes. Even when the rest of your programs have been converted to NM, the old programs will still run, processing the same files and data bases as the converted programs.

To many large shops, this approach holds a special appeal: as long as the company still operates any MPE V-based system, it will be necessary to maintain CM programs. Any program which is run in CM on the Precision Architecture machines will still run on MPE V-based 3000's; meaning that *only one version of the program need be maintained.*

At Unison Software we have taken further advantage of this capability when translating packages written in SPL. By choosing to translate only the performance-critical programs, we avoid the cost of complete translation while reaping most of the benefits of NM performance. Over time, we expect that less critical programs will be translated as well in the course of system maintenance and enhancement.

For example, our Maestro package consisted of something over 60,000 lines of SPL source code. Migration of the complete package to CM was accomplished with less than two days' work, mostly spent resolving the incompatibilities in Privileged Mode calls to DIRECFIND. We have since translated the job controller program (over 10,000 lines) to NM. This program was selected because it runs continuously during the batch processing cycle, therefore raising the greatest performance concern. It is now possible to run the complete package in CM, or run the job controller program in NM and the rest in CM, with complete compatibility.

Mixing Modes

HP's CM processor not only allows different programs to run in different modes on the same machine, it allows a program to switch modes during execution. This is excellent for systems with extensive libraries of common routines such as those found in a local SL (Segmented Library). Since these routines are used by many programs, they might be needed in both NM and CM programs. The effort required to migrate these libraries could add significantly to the cost of migration the very first program to NM.

Once again, we can gain flexibility by taking advantage of the multi-mode execution capability. Rather than having to translate library routines to support NM programs, the NM programs can call the CM library routines from an SL. This only requires the provision of an intermediate routine called a "Switch Stub". HP provides a utility program which makes creating Switch Stubs a fairly easy task, although you might find it just as easy (considering future maintenance) to migrate short library routines to NM. When the library routine is eventually migrated, it is only necessary to replace the Switch Stub with the complete routine to achieve full NM execution.

In our translation efforts we took advantage of this ability to leave two major SPL routines in a CM library until the rest of the program was completely converted. These two routines were especially troublesome to translate, because they were privileged procedures using undocumented intrinsics and privileged machine instructions.

One routine used privileged machine instructions (MFDS) to access the JMAT, one of MPE's system tables. The other called an undocumented system intrinsic (DIRECFIND) to read system directory entries and obtain logon passwords for the jobs being streamed. All those tables were still accessible under MPE XL in CM (once we had accommodated the changes alluded to previously), and the privileged routines still worked, even when called by an NM program (Figure 3). For future safety, those routines were redesigned to use documented MPE interfaces when they were ultimately translated.

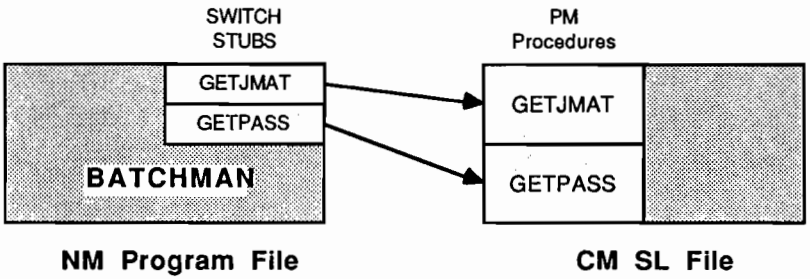


Figure 3. Mode Switching to CM SL

Staying Compatible

As a software supplier to many HP 3000 users, our prime concern when translating programs and routines to an NM-capable language is maintaining source code compatibility with the MPE V-based HP 3000. Our objective was to produce new programs which would only require recompilation to move from CM to NM. This is one of the reasons our language of choice is Pascal: HP offers a Pascal compiler for both CM (Pascal V) and NM (Pascal XL).

This strategy allows us to support users with existing HP 3000's as well as Precision Architecture machines with only one set of source code, keeping maintenance costs to a minimum. Although we feel we have achieved this goal without severe compromise, a number of problems surfaced along the way.

The obvious and well-documented problem is the difference in data representation. Although of little concern within the confines of a single program, it is essential for compatibility that record structures not be affected by the choice of a CM or NM compiler. After all, our NM programs nee to read records written by CM programs.

Floating Point Differences

A fundamental difference in data representation is the switch to IEEE floating point in the new machines. This was transparent to our SPL packages: our only use for floating point is the MPE PAUSE intrinsic. Since the pause intervals are always calculated internally by the program, never written to files, the change is automatically taken care of by the compiler.

Resolving the difference in floating point representation required some effort in the COBOL package. Since the COBOL compiler did not handle floating point, the PAUSE intervals were hand-coded as constants in the data division. These constants need to differ depending on the compiler being used (NM vs. CM). We resolved this by removing the constants from the Data Division to a library routine written in Pascal, so the Pascal compilers would accommodate the difference in floating point formats. The COBOL program simply calls the Pascal routine during startup, and returns the needed constants to the Data Division in the correct representation.

Storage Allocation

The second more significant change in data representation is in the storage alignment assumed by the compilers. Pascal V takes the 16-bit word as a fundamental unit, while Pascal XL assumes a 32-bit word. HP has provided for this with a compiler directive which forces the Pascal XL compiler to use Pascal V storage alignment. This may result in some loss of efficiency due to poorly aligned record elements, but it ensures compatibility with records in files accessed by MPE V (CM) programs. It is interesting to note that the Pascal XL compiler lacks the obvious option: to perform 16-bit alignment only on those records which are intended to communicate with CM programs, preserving 32-bit efficiency for the program's internal data.

Perhaps because we carefully prepared for problems in this area, we experienced relatively few problems with data alignment. We have not used the compiler directive forcing 16-bit alignment, although in a few instances we had to experiment with Pascal record definitions to achieve alignments compatible with the actual record layouts.

Our success was achieved in part due to two Pascal type definitions which we incorporated in translated programs from the beginning (see Figure 4). These two definitions are in a small file which is $INCLUDEd in each source module. It is a simple matter to replace this file when moving from Pascal V to Pascal XL.

Pascal V

```
TYPE
  SHORTINT =  -32768..32767;
  INTPTR   =  SHORTINT;
```

Pascal XL

```
TYPE
  { SHORTINT predeclared }
  INTPTR   =  INTEGER;
```

Figure 4. Essential TYPE Declarations

SHORTINT is the only true 16-bit integer type recognized by the Pascal XL compiler, stored in 16 bits and aligned to a 16-bit boundary. Since we were coding our Pascal from scratch, we were able to use this special name all along. In Pascal V, we declare it to be the subrange -32768..32767.

INTPTR is an integer type of the proper size to hold a pointer value. It is declared equivalent to SHORTINT for Pascal V, INTEGER for Pascal XL. This is

essential for those new MPE intrinsics, such as CREATEPROCESS, whose input consists of arrays which can contain either pointers or numbers. The BADDRESS and WADDRESS functions are used to generate the pointer values (see Figure 5). The above declarations accommodate the fact that MPE V uses 16-bit pointers, while MPE XL needs 32 bits.

```
VAR
  ERR, PIN     : SHORTINT;
  PROGNAME     : CHAR16;
  INFO         : CHAR80;
  ITEMNUMS     : ARRAY[1..5] OF SHORTINT;
  ITEMS        : ARRAY[1..5] OF INTPTR;

BEGIN
  PROGNAME := 'FCOPY.PUB.SYS ';
  ITEMNUMS[1] := 3;
  ITEMS[1] := 1;                    { REACTIVATE ME ON COMPLETION }
  ITEMNUMS[2] := 10;                { SUSPEND ME NOW }
  ITEMS[2] := 3;
  INFO := 'FROM=INFILE;TØ=OUTFILE';
  ITEMNUMS[3] := 11;
  ITEMS[3] := BADDRESS(INFO);
  ITEMNUMS[4] := 12;
  ITEMS[4] := 22;
  ITEMNUMS[5] := 0;                 { END OF ITEMLIST }
  createprocess(ERR, PIN, PROGNAME, ITEMNUMS, ITEMS);
END;
```

Figure 5. Example of SHORTINT and INTPTR

Compiler Compatibility

Finally, there is the question of compiler compatibility. Pascal XL supports a superset of the Pascal V language extensions, meaning that nonstandard elements available in Pascal V (such as 'INTRINSIC'), are also available in Pascal XL. However, a few very attractive Pascal XL extensions, such as the ability to freely manipulate address pointers and cast their types, are not included in Pascal V. Our simple solution to this was to forego the extensions, essentially developing in Pascal V and recompiling in Pascal XL for NM.

It turns out that compiler options themselves posed the only bothersome incompatibility. Any Pascal V program which replaces a 10,000 line SPL program must be segmented, in order to fit in HP3000 code segments under MPE V. This involves the Pascal V '$SEGMENT$' directive. The Pascal XL compiler issues a warning for each of these statements, and the profusion of useless warnings can obscure other warnings pointing out questionable language constructs. We hope that programmers will be able to suppress these in the future.

Summary

Our experience to date indicates that HP's migration strategy has much to offer the HP 3000 installed base. The high degree of compatibility, the ability to run different programs in different modes (CM or NM), and the ability for a given program to call routines executing in the opposite mode are the highlights of this strategy.

Together with the availability of common high-level languages, we have found that this strategy is successful in providing the power and flexibility needed to support not only migration but ongoing development of compatible software systems.

#

AN ENVIRONMENTALIST'S COMPUTER ENVIRONMENT

by: Allan Chalmers

Sierra Club
730 Polk Street
San Francisco, CA 94109

This paper is directed toward the computer center manager who would like to know what someone else has done with the opportunity to design a building wiring scheme and a computer room from scratch. I particularly would like to pass on warnings on mistakes that I made and hope that this paper can save someone from doing some of the same things wrong.

I also want to share with you some hints on running a data processing operation in an environment where money is carefully spent and personnel are at a minimum.

Please note: It is not my intent to try to bring some new, technical concept to you in this paper. Its intended audience is not the programmer, bit-fiddler or general techy.

BACKGROUND

The Sierra Club, an environmental organization started in 1892 by John Muir, has been computerized since 1978, when the Club contracted with a best unnamed Hewlett Packard distributor to install a Series II and develop a membership system for its use. The Distributor failed miserably, leaving no software to be used on this new computer. The Club then went to its then current service bureau and had them develop a system whereby the club supplied the service bureau with tapes of membership updates three times a month. The service bureau then supplied an updated membership file to be loaded onto the Series II. The input data was supplied by a data entry crew using an Inforex dedicated system. The Club's programmer at that time wrote a system so that the membership staff could access information for member services on a read-only basis. This system also had a donor information file and a dropped member info file. All in KSAM.

The Club then went to a big 8 firm to write specs for a new system. This took a year, and at the beginning of 1980 when I was hired, we were going out with an RFP for someone to write the system. The contract was awarded to Gentry, Inc. of Oakland, California. This process took about a year, with a project manager and three to five programmers working on site. We brought the system up in March 1981 and except for the addition of a Donor information sub-system written in-house, the software is much the same as it was then. We added a financial system from Smith, Dennis & Gaylord, a reservation system

for our Outings Department, written by a former member of the membership system programming team, John Alleyn-Day, who is also speaking at this conference. An activist system and a Catalog Department Buyers information system have been written in-house using Relate/3000. Two other small systems have been replaced by microcomputer turnkey packages; Personnel and magazine advertising.

We have moved from an environment of a Series II with a half Megabyte of RAM and 200 Megabytes of disc, with a few HP2640B terminals to our current configuration of a Series 48 with eight Megabytes of RAM and one and a half Gigabytes of disc with somewhere around 90 terminals/computers connected or connectable to the HP3000. Future plans are for the installation of a Series 68 next year if we can find the money. I don't believe that a Series 58 upgrade will be cost effective, but if anyone in my audience has gone through a similar upgrade I would be pleased to hear from them whether they saw a substantial increase in horsepower or not.

We use Relate/3000 almost exclusively as a programming language and for database development. Our membership system is written in COBOL using IMAGE, but the donor information system, which ties in closely to it, utilizes relational files. Due to the nature of a system that is used to select and analyze members based on criteria that constantly varies and to develop creative donation appeal strategies, the only software that fit our needs when we were searching five years ago was Relate/3000 because of its unique ability to access IMAGE databases. We are still very pleased with its flexibility and the productivity in programming it gives us. As with any 4GL, one has to be careful when developing programs so that they don't run off with your system. I would call your attention to the paper given by John Alleyn-Day on this subject at the BARUG meeting in May of this year.

The Smith Dennis & Gaylord Accounts Payable, General Ledger and Financial Report writer software runs in compiled BASIC. SD&G uses a modular programming system that is very effective, allowing their customers to get custom coding created at a reasonable cost. Proficiency using this system, called TRM, is required however, and I find it best to let them maintain their systems.

Software tools used at the Club are minimal - we use the following:

SYSVIEW - An OPT/3000 competitor for monitoring the system operation
ADAGER - Has never let me down
QEDIT - What can you say about this product that hasn't been said?
SUPERTOOL - With this Robelle program I went from four hour serial reads through my membership detail file to 15 minutes per.
TEXTPRO - A word processor on the HP3000

We will also be using a package, unchosen as of this writing, to do job accounting.

I don't use too many contributed programs, but those I do use are usually the ones that come with the ROBELLE and REGO accounts, the Boeing TECH account and TELESUP. DIRK in the TECH account is a very useful tool for file maintenance.
The staff is composed of myself, a Programmer/Analyst, a computer operator, a Programmer Trainee, Telecommunications analyst, and a part time Training Specialist (micro and HP3000 software).

THE MOVE

The Club had been operating its national headquarters from leased space on Bush Street in San Francisco but was coming to the end of its lease in late 1985. A decision was made to purchase a building and rehabilitate it for our use. The selected site was a 60+ year old 40,000 square foot building on Polk Street in San Francisco.

Here was an opportunity to design a building for computers! Not exactly a "smart building", but one that would be "computer savvy". I had done a great deal of climbing in the rafters stringing cable for every terminal that came along (remember this is a low budget operation), and I was very keen on not having to do that anymore. When you have to do it for each addition you find it is quite expensive to bring someone in and purchasing a reel of cable and doing it yourself is very cost effective. (Do others DP managers find that money for cabling never seems to be in the budget?) Ideally you will have someone on your staff who will do this work.

The plans for the building included a major structural project to bring it up to code in earthquake aware San Francisco along with new construction on most floors. For reasons of economy, walls, electrical outlets, doorways were to be retained on two of the floors. This was really more trouble than it was worth and led to our use of inefficient space. Office space designed years ago does not adapt well for these times. Retaining original structures also led to skimpy electrical provisioning. The person holding the pursestrings decided that one outlet per office, or workspace should be good enough, seeing as it was good enough for the last inhabitants. Consequently we have countless power strips around the building.

The computer room was placed in the center of the building and on the third of four floors for several good reasons, but mainly for security. You don't want easy access from windows, the roof or the street; the computer should be centrally located for cable length (RS232 certainly works past the artificial maximum distance of 50 feet, but don't stretch your luck). Whatever you do, don't put it in the basement, although that appears to be a popular locale for HP3000's. If there is a fire in the building - anywhere in the building - the water poured onto it by the fire laddies will end up you-know-where. If you are anywhere near a body of water or a river there is also the possibility of flooding. Now if you are already in the basement, please don't use soft tip pens for tape labeling - it

will wash off when you do have the flood and you won't be able to identify your tapes after drying and washing them.

TELECOMMUNICATIONS

For the same security reasons put the telephone switch, in our case a Telex Computer Products LEXAR 1001S, in the computer room. Other reasons for doing this are becoming more obvious as time goes on. The switch is a complex and powerful minicomputer and as the two technologies move closer together in function and importance, so will the departments. If you don't already have responsibility for telecommunications, you possibly soon will in a small organization. Other reasons for having the two computers together are that you only need one air conditioned, secure environment and all wiring terminations are in the same room.

If you are moving or rehabilitating, you will undoubtedly be looking at a new telephone system. Our experience in this area was slim to non-existent, as will be most organization's; we just don't do it often enough, plus the technology is advancing at a dizzying pace. The Club hired a consulting firm and we were quite satisfied with the results. They knew how to put together an RFP so that we were covered in any eventuality, and they were available for weekly progress meetings that took place between the phone company, the phone switch company, the building communications wiring contractor and Sierra Club staff. Their representative at these meetings was knowledgeable, intelligent and reliable and is still consulted by our telecommunications analyst.

The staff person at the Club was hired to manage the old uncomplicated system and was instrumental in bringing everything together in the process of going high-tech in phones. She spent long hours bird-dogging the wiring prior to installation, testing each location individually, training all the personnel in the use of these phones (Each station is controlled by a Z80 processor). You must have someone on staff dedicated to the communications functions of the organization. Either that or let the phone companies do it all and pay the price. Obviously this same person should also control data communications in an organization the size of ours.

DATA WIRING

My initial assumption was that I would be wiring the building for a data/voice switch, which at that time was an available feature on most switches, however when I found that the least expensive vendor's add-on price was about $60,000, I looked at the alternative, the data switch. We put out an RFP, with the assistance of the consultants, and received back about twelve proposals. It was clear to me that one switch, the EQUINOX, was far superior and it was the one I chose. By adding wiring for data at the same time you are installing telephone wiring you are increasing the installation cost minimally.

I installed the switch at the old location to become familiar with it and used it there for about four months on about 20 of my terminals. I configured it with 120 ports and soon after moving found I was going to run out. You need a port for each port on the HP3000 plus a port for each terminal, computer, printer and modem. I decided that there wasn't much point in running the data entry workhorse terminals through the switch when the operators signed on in the morning and stayed on all day on the same session. I ran twelve stations past the switch, directly to ATP ports. The main console and the auxilliary are also directly connected, as are two incoming modem ports and a spooled HP2631b used for printing miscellaneous screens.

The Equinox distributor also contracted with me to install a switching system utilizing TRW patch panels that consist of two fifty pin female connectors and 24 RJ11 connectors. We designed the system so that all data lines, which terminate in the computer room on standard telephone punch-down blocks, are connected by fifty pin cables to the 16 TRW panels dedicated to the incoming ports. From these 384 connectors, RJ11 patch cords route the incoming ports, as required, to the 96 input ports on the data switch. They are in turn switched electronically to 24 non-dedicated ports on the HP3000. In other words:

384 locations in the building are accessible through 96 data switch input ports to 24 ATP ports on the HP3000.

Obviously, this is only going to work up to a certain number of terminals, but as of this date, I can count on one hand the number of times a user has had to wait for a session. Don't forget, the heavy users are directly tied in to their own dedicated ports. We usually run no more than 32 concurrent sessions, so you can see the reason for port availability. Just as obviously, if I get more than 96 connections to possible users, I will have to expand. The Equinox switch will accomodate 1320 ports on 55 cards. I have 5 cards now. No problem. Each 24-port card is about $1800. I don't have a contract on the switch; Equinox has a plan whereby you can get a card in 24 hours. You can also elect to purchase an extra card and you can install a redundant processor. So far, no problems on ours.

The switch also allows the use of a modem pool. We have three Hayes modems in an outgoing pool connected to switch ports. Most people want their own modems so that they can avoid perceived contention, but digital telephone switches won't allow that will they? You can't MODulate/DEModulate a digital signal. There will always be folks who will insist though, so you put in a separate line and modem for them. Equinox has optional port boards that support modem control signals, but with four wire RJ11 connectors, they aren't going to be of much value and besides, they are quite a bit more expensive. The Equinox switch is programmable for length of idle time before timeout so you can avoid hung modems when someone forgets to drop the connection. The normal method of disconnection is by three rapid breaks. In the case of break-less devices like TRS80 Model 100's, a short period

until time-out is essential. There also must be a discipline required for Hayes protocol resets to the modems because of the lack of control signals. The three modems are set up so that the operator and I can see them from our desks which aids in troubleshooting. Our experience with this setup has been very satisfactory. The pool is used extensively for electronic mail.

In addition to allocating ports on the HP3000, the switch can be used for routing users to printers. I initially had planned to do this but found that buffering printers and connecting them by patching through the panel is a more viable, transparent method. I have installed multiple DATA MANAGER 256K buffers in the computer room, each of which buffers five computers to one Laserjet, bringing the computer lines to the buffer and passing the output back. This way I avoid having a jumble of lines in the user area and also have centrally located equipment. Of course, there are not enough remote ports in any one location to do this in any case. With the Buffer in the computer room, only one line runs to the printer at the user station.

The fourth lead in the RJ11 connection is used for the Laserjet's Data Terminal Ready signal. Impact printers could also be connected in this way, but they do not lend themselves as well to this usage. Parallel printers are not going to work here of course.

Another feature of the wiring scheme is the lack of spaghetti-like wiring all over the walls, floor and ceiling. If, for instance, a computer is dedicated to a printer that is not in a location that allows an unseen connection, the wiring can be routed through the patch panel. Ideally, you would want as many remote ports as possible for future usage, but reality sets in when you price the job out. Plus, workstations tend to cluster and you can drop multiples in locations that you anticipate will require them when you are designing the system. We used four pair cable for data and three pair for telephone. If there is a requirement for another port in any location, the second four wires in the data cable can be split off and another port is available once that pair is punched down. You do need to have spares on a TRW panel for these inevitable extras.

The Old fifty foot distance restriction hasn't been a problem except possibly in one instance where HP2640B terminals were being used through the switch in a Relate/3000 reservation system. I added a set of four used ADCC ports to the ones I already had and connected the 2640B's directly, bypassing the switch, and the problem went away. I've since replaced the old terminals. Our Conservation department has a TOPS network with a couple of Mac SE's, a Laserwriter, an AT compatible and a couple of XT compatibles, and they are also running through the building wiring without a problem.

I have laid out a number of benefits of the wiring plan at the Club and I suppose I had also better set down some of the real and possible problems or concerns that might come to mind. First, of course, is that you might want to consider coax in wiring the building. Benefits

there could be faster data speeds and, if your company surrendered to the defacto standard bearer IBM, you would be in trouble. On the other hand that would be a lot more expensive and really isn't justifiable for some possible future need. As mentioned we are using a TOPS network through the twisted pair RJ11 system and it is working well.

An obvious caveat to this wiring and switching scheme is that it should be documented well. This is a case of "do as I say not as I do" however. Between writing this paper and delivering it I promise to catch up on that! However the EQUINOX switch has menu driven software and is pretty much self-documenting. Connections through the patch panel are easily traced although the sheer number of patch cords might be daunting.

WRINGING OUT THE LAST OUNCE

I want to pass along some of the practices in use at the Sierra Club so that you can weigh the pros and cons and perhaps use our experience in your own shop. The first International conference I went to, San Jose in 1980, was directed toward those new to the HP world. It was of great benefit to this new HP manager in getting started on the right foot.

The Sierra Club's capital budget has historically been funded by income; no borrowed money except in extreme cases. I went to the Board with a request for a Series 48 because the difference between it and a Series 68 was $120K and at the time the 48 was more than adequate. In retrospect, I should have gone for the more powerful machine, because I now need it. Having an underpowered computer is somewhat akin to driving a small four cylinder car; If you don"t keep it in tune, it"s going to perform poorly, unlike a big V8 with lots of extra horsepower which will run adequately when badly in need of a tune-up. some of my recommendations are:

Run with as much memory as you can afford. We have eight megs and are using about seven. Caching will take advantage of just about all the memory you will give it. I have a one Meg board from EMC and another four meg board from a supplier who wishes to remain anonymous. Neither board has hicupped once. I am sure you all know the difference in price between HP and third party memory sellers, and that there is no maintenance charge on the latter.

Turn off Console Logging. On advice from Steve Cooper of Allegro Consultants I have effected a few I/O reducing tricks including installing an HP2621P console, turning off logging, turning on the printer at night. Saves lots of I/O's and you check the console log daily this way. I bought the 2621 for $95.

Make sure your peripherals are distributed across GIC's correctly Your CE should have done this without prompting from you.

Run the program FILERPT from the contributed library against file opens and closes to determine your most used files and spread them accordingly. If you have one Eagle, as I have, and a REALLY heavily used file, as I also have, be sure it's on that drive. You have to turn on logging for these opens and closes long enough to get an accurate picture. Turn it back off as soon as you can - log files fill fast while you are doing this. You are also defeating your I/O reducing purpose while this is going on.

Use a 7978A or B tape drive. Cut your stores to a quarter of the time. If you have a 1600 BPI drive, get one of the fast store programs.

Keep a close watch on IMAGE efficiency by running HOWMESSY or DBLOADING. There is a good write-up by David Greer on interpreting this program, but pay particular attention to elongation. DBLOADING is so slow that you won't run it often enough. Get HOWMESSY from ROBELLE.

Hire a consultant annually, to come in and evaluate your operation. Whether you use the HP Performance Evaluation service or a private party depends on how much you can afford and whether you can find the latter. If your operation is running smoothly and efficiently you send the report to upper management, if not you sort it out based on the analyst's recommendation and send the report next year when all the "gigs" are corrected.

I am a proponent of Management-by-Walking-Around. If you don't get out with the people on the terminals you often don't hear the complaints, not to mention getting to see the response times. My standard is the membership data entry screen: How long does it take to enter a new member. I know how long it takes on an unloaded computer, so I stand behind operators and count the seconds often.

BEING TIGHT WITH THE BUCKS

Think about the way you are doing things if you really want to watch expenses. You don't have to jeopardize your Data Center or your career by cutting down on the cost of your operation. For example:

Consider Next Day Service. You are saying, "But the CE comes out tomorrow with that contract, I can't live with that!" Well folks, did you know that if you pay a one time charge of $300 you can get two hour response on any call. Think about that in relation to your operation. How many emergency calls do you make in a year. The difference between the two versions last time I looked was about $300/month on my contract. HP obviously doesn't go out of its way to publicize this option. In over two years I have paid the deductible only twice.

How many HP terminals, computers, plotters, printers do you have? When did one last break? I took all my peripherals, except for line printers, off contract. I haven't called HP for a time and materials

repair in almost two years. Some say that the line printers should also be off contract, and in my case they soon will because HP2617's aren't going to be supported after next April. It helps to have a Mr. Fixit type on staff of course, because most of the problems that come up are either configuration, usage, or minor mechanical failings.

Don't turn off micros and terminals at night, just have folks turn down their screens to save the phosphor. The money wasted in electrical use is more than made up for in computers that don't come back on in the morning. HP150's are particularly aggravating when they lose their configurations. You're right, wasting electricity is anti-environmental, but it's hard to be a purist 100% of the time.

We have quite a few non-HP micros around, including about 20 generic clones and none of these are on a contract either. Keep a set of spare boards on hand for problems that might come up with these devices: They don't hum along as well as HP equipment does.

Buy used HP equipment. This is quality stuff and you won't get burned. I have bought about a dozen HP2645's for $200-300 each. Cleaned out Leasametric. Theirs came with the dual tape drive option. This terminal apparently won't work with a 930, which I am told will not support Xon-Xoff but for 200 bucks you can throw them away at that time. Besides they are a great terminal with keys placed more appropriately for MPE commands than those with the "standard" layout.

Drop your SE. I was paying about $4000/year for a quarterly visit and handholding during MIT installations. Switch to Response Center Service. If you need an SE call one out on Time and Materials.

Last but not least, pay your staff well with all the money you've saved.

WORDS TO THE WISE

I particularly want to pass on some advice based on my experience that could benefit people when planning a computer center, whether from scratch or as a redesign:

You should obviously install Halon; Your computer has an excellent chance of surviving a fire, whereas even the most minor fire doused by water is going to render your hardware scrap. However, please remember one thing, and this pertains in any Halon protected installation whether new or existing: Know what to do when bells start clanging! The first thing that came to my mind when this happened to me one Sunday morning was, "I don't remember what to do!" Panic is what you don't need. Do you know and your staff know exactly what to do in all instances other than normal? Find out. In my installation and probably most others, the first alarm indicates that one head has picked up combustion, or thinks it has. If another head does the same, you then have twenty seconds to get out before Halon triggers, or you can hit the abort button for a 20 second delay. If

there is no second pickup, you probably have a false alarm. Know how to stop the clanging and how to reset. Also, the alarm company people will want to know which head is the culprit, so before resetting look at all the heads. The one with the light lit is the one. If you don't do this, you will surely get another false alarm that will again scare the blazes out of you.

The HP3000 really doesn't require false flooring to route cabling. I decided to cut expenses by not installing it. Unfortunately, I didn't count on the contractor installing an air conditioning unit noisier and windier than my old one. Consequently, it is an inhospitable environment. I had planned on an isolated processor, with the operator and two line printers in a different room and switching the console to another terminal in that room. That's fine, however there is a certain amount of work that has to be done in the computer room no matter how you try to isolate it. The second mistake you should learn from my experience is that I didn't get to specify my air conditioner. Don't let a contractor get away with that; you be sure you are satisfied with the unit.

Don't take anything for granted in the design phase. I assumed that, because the rest of the building was getting dropped ceiling, so would the computer room. Wrong again, I had to install insulation after the fact. I couldn't tell from the plans that the electricians were dropping the fluorescents to different levels in the computer room because of piping, therefore not allowing post-construction dropped ceilings.

The floors in this building were the original seven inch thick fir beams, dense as concrete. These were to remain, so I wasn't going to have a problem with floor loading. Wrong again. Directly under the printer room I found a large crescent shaped hole, from a long gone spiral staircase. It was covered by a couple of sheets of half inch plywood, then by the old flooring. If I had rolled my tape safe across that spot it could have been exciting. It would also have negated my security-minded design work if I hadn't corrected the problem.

Be sure to install a thermal cutout so that if the A/C fails, power will drop in the computer room. The processor has a sensor that will shut it down, but the drives don't have this safeguard and you could fry things. HP makes you pay for that kind of damage. Because the phone switch is much less forgiving of power loss than the HP3000 and because all the Direct Inward Dial lines drop and must be brought up by the phone company, we don't want this to happen and bought a dandy little inexpensive sensing device. It constantly monitors temperature, noise level and moisture level on the floor, and calls four different numbers in rotation until it gets an answer, then in an insipid electronic male voice, it tells you its readings and lets you listen in to the room. The price on this was $250 plus a telephone line. It has already paid for itself when the A/C unit threw a belt. I am only fifteen minutes from work and was able to get there before

the temperature reached the cutoff.

A Data Processing manager should always remember that he/she is ultimately responsible for anything that occurs within the computer room and with any security breaches that might occur. Don't rely on other departments to protect the integrity of your operation. Building security may not be in your domain, but you had better be checking it. Your building should be under surveilance after hours, preferably by an on-site, full-time warm body. Most companies understand the importance of computer operations to their continued existence and you are the one responsible for seeing to that.

Check the entrances, alarms, windows yourself. Report problems and be sure that someone follows up. Do you let the janitorial service into your data center unattended? You shouldn't. They aren't even company employees in our case. Check your console log every morning for unauthorized or suspicious activity over your modems. Change your passwords on any account with System Manager capability whenever Someone from HP accesses the Telecom account. Make it very plain to Personnel and the user departments that you or an appropriate member of your staff is to be informed immediately when someone is let go for cause. Don't PURGEUSER the late, possibly dangerous employees, just change the passwords they used, so that files they created are available to others easily.

Know your vendors. Try to find out about them from their current and former customers. I had a bad experience with a service provider that might have been avoided if I had investigated further. Obviously the references given will be well chosen by the company, so you must look for disgruntled former customers.

Watch the personnel in the user departments for likely candidates for your staff. I have had exceptionally bad luck with "experienced" computer operators, but just the opposite with people who I recruited from within the Club and trained.

AND TO THE DATA-WISE

It may not be very sophisticated, but we back up our databases nightly. With Relate/3000 files keyed to our IMAGE databases it would be most difficult to use logging. We fortunately are able to recreate our on-line activity at any time, so in case of a System Failure we restore suspect databases back to the night before the re-enter. Having suffered a broken chain in our most critical detail set at the most critical time of the month, I am not anxious for a repeat performance.

Be careful in creating new IMAGE systems, that all data is kept within datasets. Our membership system has the serious flaw of numerous flat files being keyed to and updated along with IMAGE sets. If you want to do IMAGE logging this will be a problem.

Don't be in a great hurry to install the latest MIT. I used to be the first one on the block to have the latest when there were really neat features coming along, but I have since re-thought that philosophy, especially if there is a performance penalty. We are still on old, regularly-aspirated IMAGE but I guess HP will force us onto the Turbo model soon.

MISCELLANEOUS

Go to conferences and RUG meetings, join Special Interest Groups, like SIGSYSMGR (meeting during this conference). If you are the manager of a small operation you are undoubtedly the System Manager too.

Develop user department microcomputer specialists. With the proliferation of these things, their printers and software, you can be run ragged fighting fires. This is another reason for watching user department people for talent.

CONCLUSION

I hope that this paper can be of use to managers who try as I do, to give your employers value for dollar. As I have stated before, I am only giving you my opinion on running a data center and I would not presume to tell you that anything herein is gospel, it's only what we are doing at the Sierra Club. I would like to hear from anyone who may have comments one way or the other on the way we are operating. I have benefited greatly from writing this paper and perhaps will come back again with an update at a future conference.

HP StarLAN Networking

Mary H. Chay
Hewlett-Packard Co., Inc.
Colorado Networks Division
3404 East Harmony Road
Fort Collins, Colorado 80525

1. Introduction

HP StarLAN, which has been shipping since early this year, is Hewlett-Packard's unshielded twisted-pair cable local area network (LAN). HP StarLAN can be used in office environments with HP Vectra, IBM (R) PCs, and/or HP 3000s. HP StarLAN is low cost and it is scalable since the network can grow along with an organization.

By providing access to HP's distributed data processing and Office Productivity Services, HP StarLAN supports a complete business office automation solution for commercial environments. The LAN is a flexible networking solution that integrates PCs and HP 3000s. The network uses unshielded twisted pair wiring, or telephone cable, for communications and often can be supported on installed wiring.

Definition of HP StarLAN

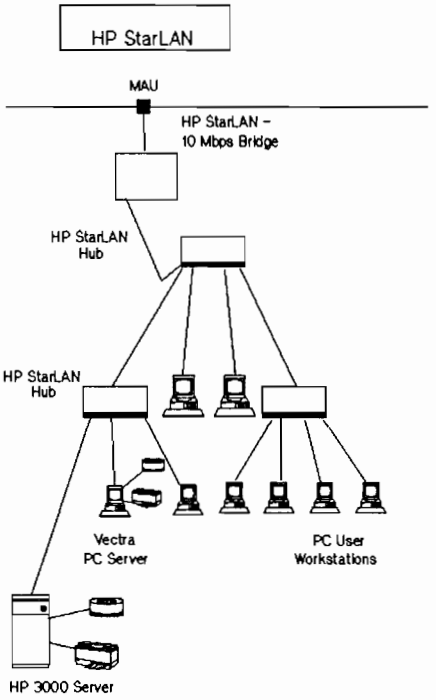


Diagram 1

HP StarLAN

mc61187a

HP StarLAN is physically configured in a star formation. (See Diagram 1). A member of the HP AdvanceNet family of products, HP StarLAN supports supports industry standards. HP StarLAN conforms to the IEEE 802.3 (Institute of Electrical and Electronic Engineers) Type1BASE5 specifications. This standards support will permit inter-vendor communication in the future. HP StarLAN support of the AdvanceNet strategy ensures a secure growth path to the future. AdvanceNet support means future access to other Hewlett-Packard systems,

including new HP Precision Architecture systems.

Implementing the Open Systems Interconnect (OSI) reference model for networks, HP StarLAN provides an interface which conforms to the OSI model for networks.

HP StarLAN also supports Microsoft (R)-Networks, also known as MS (R)-Net, a de facto standard for application software interfaces on LANs. MS-Net support means that many software programs supported on other vendors' LANs are supported by HP StarLAN and that applications can be easily ported to the network.

HP StarLAN provides the ability to share resources on servers. With HP StarLAN, users have access to transparent disc and file sharing, printer sharing with spooling, unattended backup and restore, and terminal emulation with file transfer from personal computers to HP 3000s. Users can transparently share discs and files and obtain printer and plotter sharing with spooling on a personal computer server. HP StarLAN supports HP Vectra PCs and IBM PC/XT/ATs as user workstations. The LAN also supports HP Vectra PCs as personal computer servers and HP 3000s as minicomputer servers.

2. Benefits of HP StarLAN

HP StarLAN provides a great deal of functionality, with many benefits including low cost and scalability. Users have: the ability to share expensive peripherals, transparent use,

access to multiple hosts, and investment protection.

HP StarLAN is a low-cost solution, providing LAN capabilities at a low per node cost for the networking products as well as the supporting cabling system. HP network hardware, software, LAN attachments and network components such as the HP StarLAN Hub provide an office network that typically costs less than other competitors' offerings. Depending on the age and the condition of the cable, customers may be able to use the existing telephone wiring. When existing wiring can be used, substantial savings on cable installation can be realized. Additionally, problems with a shortage of building space available for cables and cable housing can be avoided.

Even when the existing unshielded twisted pair cabling cannot be used, the costs for installing this type of wiring will still be lower than the costs for pulling coaxial cable. HP StarLAN can reduce the cost of implementation by protecting previous investments in systems hardware, wiring, and in application software and user training.

Moves, adds and changes can be easily accomplished with HP StarLAN and the unshielded twisted pair cabling it supports. A simple change in connections in the cross connects of a wiring closet will accommodate a change in the location of workstations. Easy moves, adds and changes are economical since they reduce the amount of resources required for implementing the changes.

The HP StarLAN supports distances of up to 250 meters between user workstations, servers and hubs. These distances are ample for typical business office environments, and meet the distance requirements of the vast majority of offices.

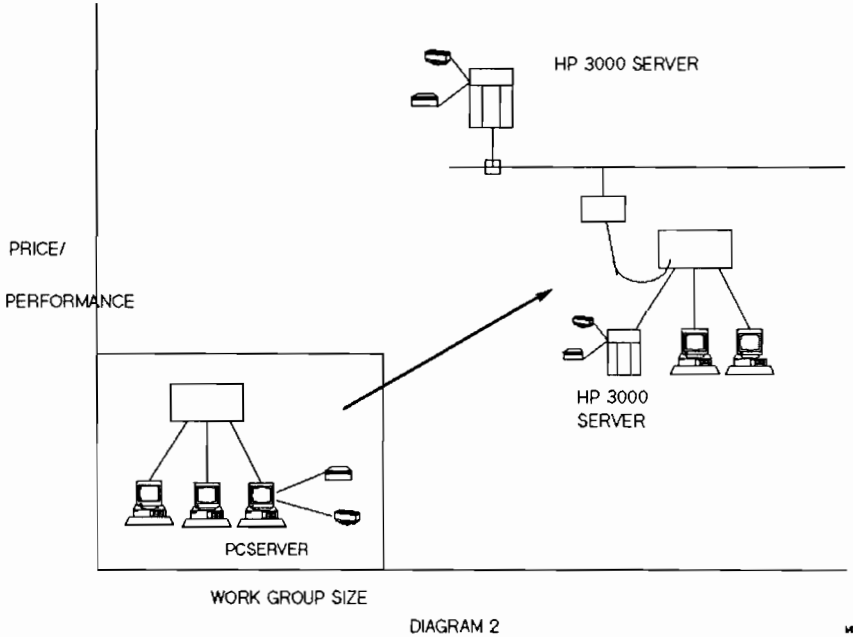


DIAGRAM 2

An HP StarLAN network can grow as a workgroup grows, providing scalability. The number of users can be increased by adding a PC user workstation with a connection to an existing server. If the workgroup outgrows the existing PC server, that customer can continue to use the PC server and add an HP 3000 server so that the workgroup has access to both a PC and an HP 3000 server. Or, the PC server can be made into a user workstation and an HP 3000 server can be acquired. Because the HP 3000 Series 37XE through 70 are compatible, if the workgroup needs a larger HP 3000 server, upgrades are easily accomplished. With each change to accommodate growth, the impact on the users is minimized since little re-training is

required.

With a network, firms can avoid the sharp increases in costs often experienced when a central processing unit must be upgraded to support additional terminal-based systems. Often, the additional central processing capacity is not fully utilized, since more terminals could be added. With a LAN, computing power can be added incrementally in amounts that can be fully utilized by end users. As the firm and its information management needs increase with growth, the needs can be met with a network that provides a framework for systems and peripherals.

With HP StarLAN, users can share high quality and high capacity peripherals. From his or her workstation, a user can print documents on an HP 2680 Laser Printer or an HP LaserJet Printer attached to the HP 3000 server. Similarly, users have access to the large amounts of disc storage space available on HP 3000 servers. They can share information with each other quickly and easily by placing files in public directories. The network administrator can ensure the security of files by using unattended backup and restore capabilities on the HP 3000. A PC server supports printer and disc sharing and also provides plotter sharing capabilities. With the peripheral sharing capabilities provided by HP StarLAN, customers can share high quality peripherals such as HP LaserJet printers. Providing a printer for each user would be expensive; sharing a printer among users in a workgroup without a LAN would be inconvenient.

HP StarLAN is easy to use: it provides a transparent user interface and minimizes productivity loss which might result from implementation. The transparent user interface enables a user to store files on an HP 3000 disc exactly as if the disc were connected to his or her workstation. For example, the HP 3000 disc might appear to be the user's E:\ disc for applications using MS-DOS. To print a document on a Laser Printer attached to the HP 3000, the user might use the steps normally used for printing to a local printer.

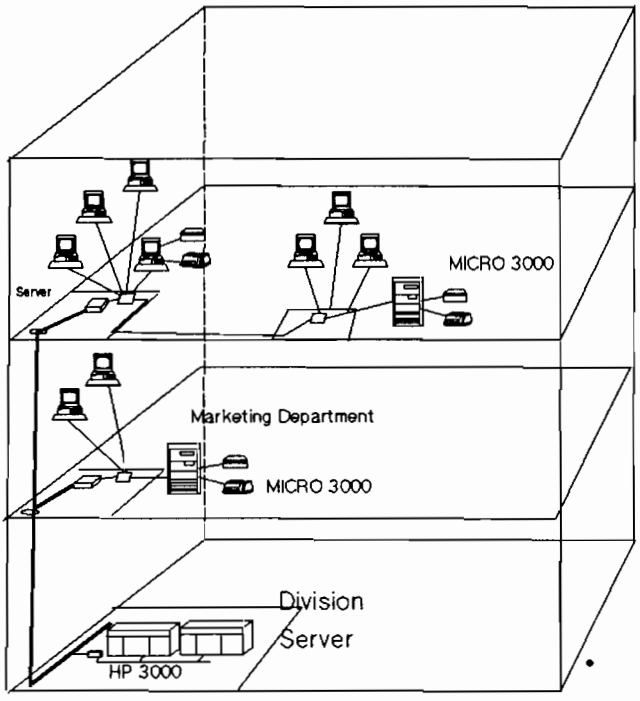


Diagram 3
Multiple Servers

MC6187c

HP StarLAN users can utilize the information and peripherals located on multiple hosts, instead of on one host at a time. Users can access current information located on different HP 3000 discs for support of better and faster decisions. For example, a financial analyst could transparently download information from a departmental or corporate database located on a division HP 3000 server. Combining that information with the forecasts located on a database on the marketing department's HP 3000 server, the financial analyst could generate division projections using a spreadsheet application.

(See Diagram 3). With the different levels of security provided by the network and on HP 3000 accounts, user access can be made secure. With HP StarLAN, users also can share a variety of printers attached to different HP 3000 servers. For example, a user could send a file to a MICRO 3000 in his or her workgroup for printing on an HP LaserJet Printer. The user could also send the same file for volume printing on a high speed HP 2680 Laser Printer attached to an HP 3000 Series 70 located in the computer room, in a different area of the building.

HP StarLAN protects investments in application software and user training. The LAN supports many popular programs using MS-DOS 3.1. Users who are familiar with a given application will be able to continue working with it or upgrade to the network version, after HP StarLAN is installed.

HP StarLAN provides additional benefits that could be obtained using point to point connections. Because these benefits are important for all information systems, they are noted here. HP StarLAN enables users to access other vendors' products, permits centralized information systems control, and has network implementation and support by HP's high quality support organizations. HP StarLAN supports access to IBM environments, with HP products for connecting to SNA (R), PROFS (R) and DISOSS (R). Through network access restrictions for security, as well as HP 3000 account schemes, systems managers can administer secured user access to information. HP can assist a

firm with the planning and smooth implementing of a network. Available support services include courses for the network manager as well as installation services and assistance with problem resolution. Productivity improvements can be realized as network users take full advantage of the productivity applications and shared peripherals supported by the network.

3. HP StarLAN Configuration

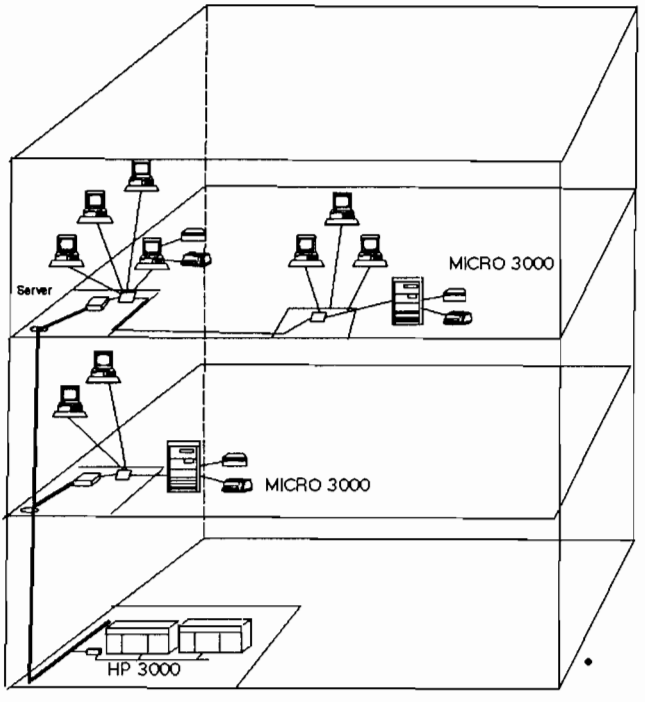
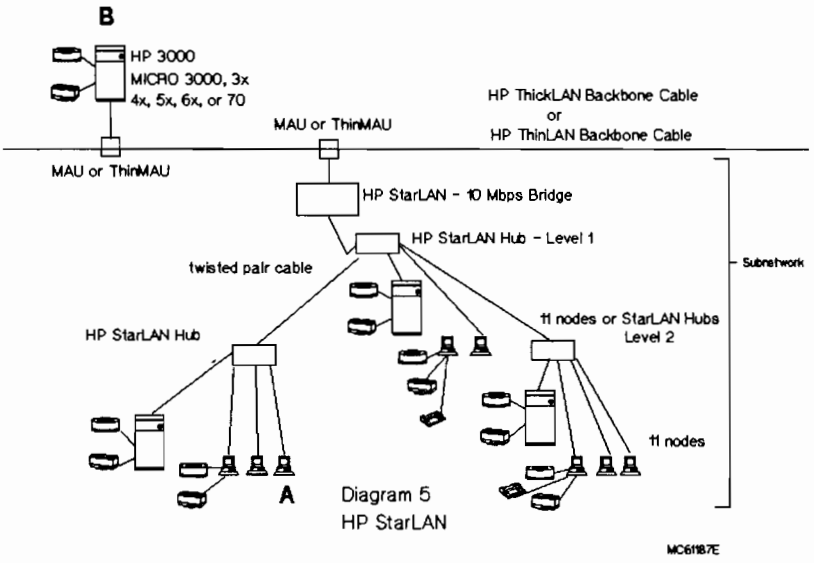


Diagram 4
Building Wiring

MC6H87d

Hewlett-Packard recommends a wiring strategy which can be designed to meet a customer's specific needs both now and in the future. Wiring systems are tailored to a customer's situation. In general, HP advocates coaxial backbone cable for connecting the floors of a building and larger HP 3000s (Series 42-70). On a given floor of a building, unshielded twisted pair wiring should be used to interconnect workgroups. HP StarLAN communications require two spare pairs of wires. Note that data transmissions will not interfere with the PBX

voice system.



The components that comprise an HP StarLAN are shown in Diagram 5. The HP 3000 Series 37XE, MICRO 3000 or MICRO 3000XE can be attached directly to unshielded twisted pair cable with a LAN Interface Controller and HP StarLAN software supporting connections to the network. Each PC user workstation and PC server is equipped with an HP StarLAN interface card and HP StarLAN software. User workstations, HP 3000 and PC servers (nodes) are tied to the HP StarLAN Hub with unshielded twisted pair cable. The maximum distance between a node and an HP StarLAN Hub is 250 meters. Hewlett-Packard supports two levels of hubs (see Diagram 5). The Level 1 hub, or header hub, can support connections to a maximum total of 11 nodes or Level 2 hubs, intermediate hubs. Each Level 2 hub can support connections to 11 nodes.

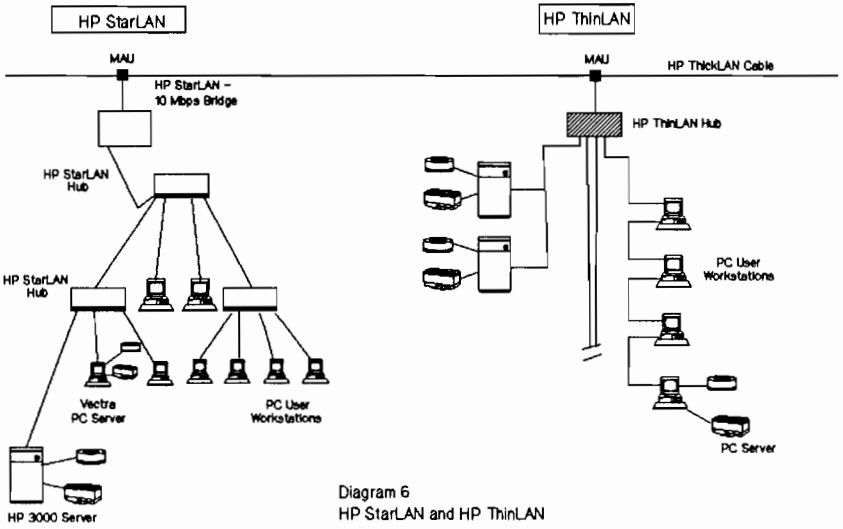


Diagram 6
HP StarLAN and HP ThinLAN

mc61871

HP StarLAN subnetworks, which are networks that include the header hub, intermediate hubs and the nodes below the header hub, can be attached to coaxial backbone cabling with the HP StarLAN-10 Mbps Bridge. The bridge supports communication between the StarLAN subnetwork nodes and systems on the backbone cable. A user on workstation A could transparently store an MS-DOS file on a shared disc on the HP 3000 server B (an HP 3000 server, Series 37XE-70) connected to the coaxial cable. (See Diagram 5). Nodes on HP StarLAN subnetworks can communicate with nodes on HP ThinLAN (coaxial cable) subnetworks that are attached to the same backbone cable. (See Diagram 6). The HP StarLAN Bridge also provides important address filtering, which causes traffic which is not intended for a subnetwork to by-pass the subnetwork.

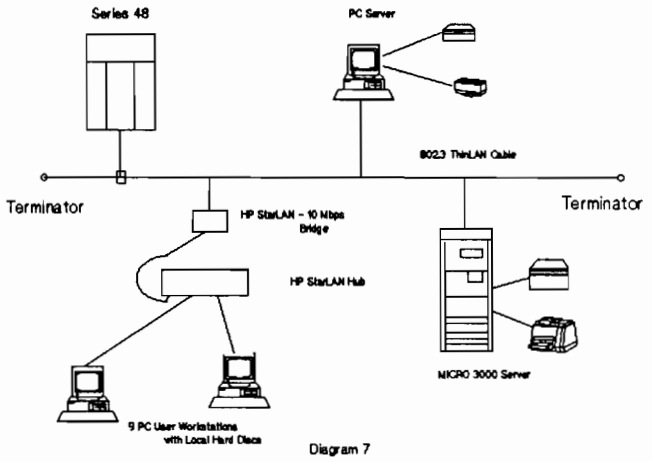
4. Performance

HP StarLAN provides the performance that users expect, with response times generally ranging from local hard disc speeds to local floppy speeds. Response times the end user sees will vary, depending on the application used and the operation in question.

HP StarLAN, with the 1 Megabit per second link speed, will support performance requirements for now and in the future in office environments. Many factors affect performance, including: the type of workstation the user has, the application software, the system that is being used as a server, the type of shared resources used, the users' work activity patterns and the type of link.

On networks, the constraining factor for performance is the network software on the PC user workstation and the server. A 1 Megabit link provides very ample capacity that typical office user activity would not consume. One result of this fact is that in office environments the performance on an HP StarLAN 1 Megabit link will be very close to that for a 4 Megabit link supporting token ring access and for a 10 Megabit Ethernet link.

5. An example implementation of HP StarLAN



Installed HP StarLAN Site

HP StarLAN was installed as a replacement in a site where the customer had been using the HP OfficeShare Network on coaxial cable. The customer is using its existing telephone cabling, unshielded twisted pair wire, for its HP StarLAN network. The network manager, who has extensive experience with HP 3000s, personal computers and the HP OfficeShare Network, found the installation process to be relatively simple. The customer's configuration has PC user workstations attached to unshielded twisted pair cable with access to shared discs and printers on PC and HP 3000s, servers and hosts for virtual terminal capabilities, attached to ThinLAN coaxial cable. The end users transparently access server resources through the HP StarLAN-10 Mbps Bridge, which connects the HP StarLAN Hub to the ThinLAN cable.

The end users at this HP StarLAN customer site use the network for a variety of activities and applications. They use the network primarily for heavy printer sharing, document sharing, integrated document management and electronic mail. With a mix of office application software, this customer uses Word Perfect, other personal productivity applications, and an internally developed database manager.

Several key attributes make HP StarLAN a suitable solution for this customer. Ease of installation, the ability to transparently share peripherals, scalability across the family of HP 3000 computers, reliability, and HP StarLAN support of IBM PCs have all made HP StarLAN a suitable product for this customer.

Summary

HP StarLAN provides a low cost and scalable network for office environments which permits the sharing of high quality peripherals and provides easy to use network capabilities. The LAN supports access to multiple servers, allowing users to transparently share files and peripherals located on different HP 3000 or PC servers. Additional benefits HP StarLAN provides include access to IBM environments, secure files, and Hewlett-Packard's high quality support and service.

Microsoft (R)-Networks and MS (R)-Net are registered trademarks of Microsoft Corporation.

IBM (R), SNA (R), PROFS (R), and DISOSS (R) are trademarks of the International Business Machines Corporation.

The Thinking Behind the Design of a 4th GL

R. S. Ching
Comprehensive Systems, Inc.
720 King Georges Road
Fords, New Jersey 08863

I am the author of a fourth generation language. The purpose of this paper is to share with you some of my experiences in writing a fourth generation language. Through the late 1970's and into the '80's, I had been programming within the HP/IBM programming environment. I had worked on many projects using the conventional languages such as COBOL and DEL (the predecessor of VIEW). As my application tasks had grown in number and complexity, I felt that the older methods were very cumbersome and took quite a lot of time and effort to apply to the tasks at hand at the time. Like many of yourselves, I became interested in finding a better way of programming. The older languages no longer had the same appeal in light of the many new ideas that were emerging as possible solutions to my dilemna. Maintaining an open-minded attitude, I proceeded to attend a number of seminars and conferences which covered a broad range of possibly easier methods. Various books and magazines provided a wealth of information on what were two particularly hot topics of interest to everyone at the time. These were:

Structural programming and
Non-procedural languages.

At that time the concepts behind fourth generation languages were just emerging. In fact, there is no universally accepted definition of the fourth generation language to this day. My primary concern was in finding an easier and quicker way of completing my application tasks. By coincidence, it seemed that the definition of a fourth generation language followed from its overall purpose-- to simply provide an easier and quicker approach to programming.

I found out quickly that the concept of structured programming was not directly related to the fourth generation language. Structured programming could be done in any generation language and in this context serves as an excellent organizational software design tool. Although structured programming has no relation to the fourth generation language, my investigation of structured programming has helped me in the development process of my fourth generation language which I will come back to later in this paper. The other hot topic was non-procedural programming, sometimes called command language. This allows the user to specify what he wants, but not how he wants it. This seemed to be a much better approach to application development. I would be able to get the job done simply by stating what I want, letting the fourth generation language do the job for me. However, my greatest reservation was that I would lose too much time if something wrong turned up in the new idea. I solved this problem by inventing a pseudo-language on paper. Initially, the pseudo-language model

provided good results. Given a simple test application, the pseudo-language worked in providing a quick and simple solution. However, when faced with an actual task, which was not contrived, the pseudo-language was ineffective and was not able to solve the problem. My observation was that as application tasks had become more complex, the language became less effective. Further evaluation of this observation led me toward two conclusions regarding the system development process. First, if the increased complexity of test applications reduced the effectiveness of the pseudo-language, then each of the shortcomings of the model must be clearly identified and the model's capabilities expanded accordingly. Second, improvements of the pseudo-language must be directed toward two areas, 1.) additions and improvements must be made to the language statements, and 2.) the relationships between language statements must be more clearly defined as they become more critical in use. Thus we see that in using our pseudo-language, non-procedural model for complex programming the system simply fails.

My first question asked, Why? Everyone that I had spoken to, and all of the sources I had investigated had told me that the non-procedural language was the best way to go. Why, then, did the system fail when asked to perform a complex task such as those I encounter daily? Why were they wrong? I decided to make a closer examination of all of the things I had been led to believe in. This led me back to the concept of structured programming. I had been told that structured programming involved no GOTO statements. Rethinking this concept, I realized that this was not true. Examination shows us that this definition is wrong. If structured programming was based on the absence of GOTO statements, then it would simply be called non-GOTO programming. After further consideration, I found out that the objective of structured programming is better organization of the program so that people can read and maintain programs more effectively. This organization results from what is called functional modularization of the program. Functional modularization is arranged so that functionally related lines of code are grouped together in the same place. Therefore, one does not have to search in many different locations for related lines of code which pertain to the same function. This results in more organized programs.

Given the following example of a credit check:

```
IF CREDIT-RATING="GOOD"
   THEN
     DO GOOD-1
     DO GOOD-2
   ELSE
     DO BAD-1
     DO BAD-2 ;

FINI:
     ....
```

```
IF CREDIT-RATING="GOOD" THEN GOTO GOOD
                        ELSE GOTO BAD;
GOOD:
                DO GOOD-1
                DO GOOD-2
                GOTO FINI;
BAD:
                DO BAD-1
                DO BAD-2 ;

FINI:
                ....
```

We see that the pseudo-program on the left was written using a well accepted structural organization. The program sample on the right implements the GOTO statement. At first glance, the program on the left would seem to be a better example of structured programming as it contains no GOTO statements. The program on the right contains GOTO statements, however, actually exibits superior structural organization. Though there are more hardware branches in the right example, it is more functionally modulated than the example to the left.

This example consists of three functions,

1.) determination of good or bad customers,
2.) action for good customers,
3.) action for bad customers.

The example on the right implements GOTO statements and distinctly modularizes these three functions. The example on the left implements the well-accepted NO-GOTO structured programming style, however, fails to modularize the determination of the good and bad customer because the THEN and ELSE statements are separated. This problem becomes more obvious if the THEN and ELSE statements are separated by 25 pages of code which contain many levels of nested IF statements. Have you ever tried to match an ELSE statement with its THEN counterpart in 25 pages of code?

To strengthen the point that the NO-GOTO style does not automatically warrant a structured program, suppose the above program also includes address checking. Someone could write the program in the following manner.

```
IF CREDIT-RATING="GOOD"
   THEN
     DO GOOD-1
     DO ADDRESS-CHECK
     DO GOOD-2

   ELSE
     DO BAD-1
     DO ADDRESS-CHECK
     DO BAD-2 ;

FINI:
     ....
```

```
IF CREDIT-RATING="GOOD" THEN GOTO GOOD
                        ELSE GOTO BAD;
GOOD:
                DO GOOD-1

                DO GOOD-2
                GOTO FINI;
BAD:
                DO BAD-1

                DO BAD-2 ;

FINI:
                ....
                DO ADDRESS-CHECK;
```

Everyone would agree that the left side exemplifies poor programming and is non-structured. Nevertheless, it has NO GOTO !

My intent in bringing forth this example is to illustrate what I came across only after careful study of what I had been instructed to do. I finally realized the reason why those people were wrong. They started to provide us with a mechanism or method to write structured programs

(the NO-GOTO programming style), yet they miss the objective or reason for structured programming (functional modularization). They place the GOALS and the MEANS in the wrong order.

These observations led me to great confidence. I began to re-examined the non-procedural languages and discovered that the logic supporting them was as incorrect as the logic behind the NO-GOTO approach to structured programming. They provide you with a MEANS, and then ask you to determine your GOALS. More specifically, they provide you with a non-procedural language. Whether or not you can solve your application task with this non-procedural language is determined by your luck.

I found that a more correct approach is to first determine your GOALS. Secondly, you must look for a MEANS to satisfy these GOALS. My number one priority is to determine my goals. I must solve all of my application problems, including the simple and the complex. Then I must find a means to do the job. With this in mind, I investigated the third generation languages. Though these languages can be clumsy, I know they can get the job done. They can solve the application problems at hand and thus achieve the goal. With these priorities in mind, I took a closer look at the COBOL language.

```
IDENTIFICATION DIVISION.
 ......
    .......
ENVIRONMENT DIVISION.
 ......
    ........
DATA DIVISION.
 ......
    ........
PROCEDURE DIVISION.
HOUSE-KEEPING-SECTION.
 .....
    .........
    IF THIS THEN
       PERFORM MY-BUSINESS.
 .....

    STOP RUN.
```

Referring to the COBOL language example, we see that from the first statement, of IDENTIFICATION DIVISION, up to the last statement of DATA DIVISION, statements are preparatory for the program. Looking at the PROCEDURE DIVISION, we see that there is quite a bit of housekeeping to do. From a business-related standpoint, however, COBOL is not bad in that it is,

1.) very precise and english-like,
2.) well accepted and tested.

If we can remove the bad aspects of COBOL and retain the good, then

this is the type of fourth generation language we are looking for. Procedurally, it has the power of a proven language in that it allows us to handle complex applications. Yet, it has the simplicity of a fourth generation language in that it provides an easier and quicker way to program.

Thus, I have come closer to the idea of a concise and condensed COBOL which eliminates the overhead of the third generation language while maintaining the capability to handle complex jobs. The program should start with the PROCEDURE DIVISION as its first statement. This allows the user to focus attention on the problem at hand and on that alone. This is illustrated by the following example:

```
PROCEDURE DIVISION.
    IF CREDIT-RATING = "AA" THEN
       PERFORM GOOD-CUSTOMER
                              ELSE
    ......
```

However, the non-procedural languages do have merit in their ability to handle simple applications. Therefore, whenever the application is simple such as in prototyping, the concise COBOL program should be optional and the user should be able to issue commands. The fourth generation COBOL is included in the application only when the added application complexity dictates its use.

As applications become more complex, one cannot avoid the use of I/O statements. More precisely, the control over I/O becomes more critical given added task complexity. One of the greatest weaknesses of non-procedural programming is the fact that the user applications must work within a fixed I/O pattern which is controlled by the system. Using the concise COBOL, the user may tailor the I/O statements around the application task at hand. We must, however, eliminate the old, cumbersome HP I/O environment. I have accomplished this by removing the VIEW and IMAGE calls, replacing and assimilating them into the COBOL ACCEPT, READ, and WRITE statements. This can be illustrated by the following example:

```
PROCEDURE DIVISION.

  START-READ.
       ACCEPT WITH F-KEY.
       IF F-KEY = 8 STOP RUN.
       READ MY-SET INVALID KEY
           DISPLAY "RECORD NOT FOUND"
           GO TO START-READ.
       DISPLAY "ENTER NEXT RECORD KEY".
       GO TO START-READ.
```

The above example, although simple for illustration purposes,

demonstrates three important points regarding I/O in the fourth generation language programming environment. First, it is complete and executable. Second, the I/O statements are very simple and readable. Third, and perhaps most important, the control of I/O is now in the user's hands. If the program becomes more complex, the user may have the option of specifying the I/O statements necessary to solve the business applications involved, without being limited by the fourth generation language system.

In conclusion, the purpose of the fourth generation language is to help the user to simplify application programming. This does not grant license to the system to take control away from the user where it is needed. Because of this, the non-procedural language can only be used as a prototyping tool. It cannot be seriously considered as a candidate fourth generation "language". The only way the user can get the simplicity of a fourth generation language, yet maintain control over applications, is through the power and flexibility of a concise and condensed procedural language. I decided to use COBOL because it is well known in the commercial environment. I hope that someday, someone will invent a concise FORTRAN, PASCAL, or other concise procedural language to serve the computer programming community.

THE CIM SOLUTION

BRICE L. CLARK
HEWLETT-PACKARD
ROSEVILLE, CALIFORNIA

INTRODUCTION

In today's business environment, failure to respond to changing and tougher competition for world markets can mean the permanent failure of a business or even an entire industry. Corporations are increasingly turning to their manufacturing managers to improve profits, product quality and costs to help stay competitive.

These objectives lead manufacturing managers to seek solutions with quality programs, productivity improvements and flexible manufacturing. Perceptive managers realize that information is the key to making these changes, and that CIM (computer integrated manufacturing) is the key to the productive utilization of information resources.

The benefits of CIM can be summarized as follows:

-- higher quality -- CIM means reduced scrap and rework, lower warranty costs and higher customer satisfaction.

-- increased productivity -- With improved resource utilization, reduced cycle times and lower overhead, productivity improves.

-- greater flexibility -- CIM translates into lower inventories, faster response to market demand and less time to get products to market.

In short, CIM means lower manufacturing costs, higher profit margins and better positioning for growth -- a healthier bottom line.

Among the specific applications that deliver these benefits are:

quality -- Quality Decision Management (QDM) from HP;

(RQM) from

Automated Technology Associates.

productivity -- HP's Maintenance Management Package (MMP);

Materials Management (MM) and Production

Management (PM)

Dispatcher from Logisticon.

flexibility -- MM and PM from HP;

Monitrol from Hillco;

StarNet from Denniston and Denniston;

AIM from Billes and Associates;

HP's Just-In-Time software.

As today's manufacturers develop and implement their CIM plans, computer networks, and the information they manage and deliver, are increasingly important.

This paper provides an overview of Hewlett-Packard's CIM Networking Solution. The CIM solution is based on HP AdvanceNet, HP's overall networking strategy, which delivers flexible, scalable solutions and embodies a strong commitment to standards-based networking. Any successful CIM implementation requires an efficient information network as the communications foundation upon which the solution is based.

CIM ACTIVITIES AND ENVIRONMENTS

A discussion of networks begins with a look at basic business and environmental factors. The primary activities that go on in businesses today include administration, planning and control; manufacturing; marketing and sales; engineering; and facilities administration. The activities performed in a given area establish the information needs of that area, and information needs help determine the most appropriate networking technology.

Since networks operate over physical media (at least within buildings), the physical environment will also have an important effect on the choice of networking technology. Office settings tend to be clean, quiet, and include phones on every desk. Computer centers are custom environments designed for computers from the outset, while factory production areas are typically noisy, dirty and can be quite large. Unlike the office, production areas tend to have few phones.

ISLANDS OF AUTOMATION

One of the key problems manufacturing managers face today are the "islands of automation" that exist on their factory floors. These islands are the result of the efforts of many manufacturers to streamline different parts of the manufacturing process. Frequently, each automated process has been designed and engineered without much thought of integrating it with other processes.

Many companies are starting to see the benefits of integrating their information sources, but are having problems communicating with a wide range of "automation islands." Advanced manufacturing companies are looking to information automation as the key to moving beyond "islands of automation" toward an integrated automation environment.

Networks can often make important contributions to control and integration of the physical automation process. Islands of automation merge as individual processes are refined to match the needs of upstream and downstream processes. This merging creates the need for "real-time" communication from machine-to-machine or control point-to-control point.

As the whole process becomes further automated, a network enables information to flow fast enough to permit real-time corrections that can either prevent errors before they happen or spot them fast enough to correct them at minimal cost.

Unfortunately, the manager who recognizes the potential of information-handling networks often encounters a horrible sight just beyond the horizon. At HP, we call this the "CIM Barrier."

THE CIM BARRIER

It's sad but true: the manufacturing manager who wants to improve effectiveness through CIM is often held back by the limitations of poor or non-existent wiring systems and the lack of multivendor compatibility that have characterized the past.

There are several key problems with the traditional approach to using computers in manufacturing. These are:

--sparse connectivity with point-to-point connections to large, inflexible systems in the computer center;

--slow information flow via paper;

--lack of communication among the computers of different vendors;

--low networking expertise in manufacturing companies;

--no systematic approach to wiring.

Collectively these problems make up the CIM Barrier. HP's CIM Networking Solution is designed to enable manufacturers to break through this barrier. The solution is divided into several modules, each of which addresses a particular communications problem. In addition, the HP AdvanceNet solution enables manufacturers to implement their networks in affordable, manageable stages.

CIM NETWORKING MODULES

The modules that comprise the CIM networking solution reflect the hierarchical structure typical of manufacturing sites. This structure starts at the level of shopfloor devices, and moves up to cell controllers, area managers, plant hosts and corporate hosts. Establishing effective communications among the levels of this hierarchy is addressed specifically in the Plant Area Management Module, but such communications can also be seen as the overall goal of CIM.

HP SiteWire Module

Within a manufacturing facility, two kinds of networks are generally found: the site backbone network and subnetworks. The backbone is a common communications channel that connects different workgroups throughout a facility. Subnetworks provide the specific functionalities needed in offices, engineering departments and production areas.

HP SiteWire is the name of the HP AdvanceNet communications wiring infrastructure. CIM networks requires a plant-wide communications backbone to connect people with information; subnetworks need easy access to the backbone from anywhere in the plant. In addition, the backbone network should be able to accommodate moves and changes easily.

The two HP SiteWire backbone options are based on industry standards. The primary and most versatile backbone option is based on the IEEE 802.7 broadband standard. It allows multiple information channels of voice, video and data. It also lets users mix terminals, point-to-point links, LANs and more on a single cable, and it supports an important new industry standard: the Manufacturing Automation Protocol (MAP).

IEEE 802.3 baseband is the second alternative, better suited to less complex situations that don't require video, voice or specialized data services.

End User Workgroup and Access Module

HP AdvanceNet offers a wide range of end-user solutions to enhance local departmental productivity while providing access to information throughout the plant. Options provide for multivendor terminal clusters and the latest in industry-standard LANs. Each option focuses on a specific area of the plant: planning and control, production engineering and the shop floor.

Thanks to terminal cluster solutions, a common problem of the past -- connecting terminals to the systems of different vendors -- isn't a problem of the present. Hewlett-Packard has entered into a special marketing agreement with Ungermann-Bass, the leading independent vendor of terminal servers, in order to provide this capability for its customers. The UB terminal servers operate over both broadband and baseband, and handle systems and terminals from a wide variety of vendors.

A planning and control staff can enjoy greater productivity and effectiveness using industry-standard StarLAN to connect PCs with information systems over low-cost twisted pair. HP StarLAN subnetworks can connect to either CIM backbone; to allow for connection to the broadband backbone, HP has extended its agreement with Ungermann-Bass to include the UB Buffered Repeater. Connecting the HP StarLAN Bridge to the UB Buffered Repeater connects a planning and control HP StarLAN subnet to a broadband backbone.

For production engineering, industry-standard IEEE 802.3 with ARPA and Berkeley networking services connects UNIX workstations. Production engineers can thus share files and peripherals, access mainframe resources and product design groups. Those capabilities can mean getting products to market faster -- a key benefit of CIM.

In the future, X-Windows will become increasingly important for standard graphics, multi-user and multi-application access. It will run on bit-mapped workstations, and,

because of its greater flexibility, will accelerate the replacement of terminals by PCs and UNIX engineering workstations. This will increase the use of LANs, replacing terminal servers.

Production Workcell Module

A key step toward CIM is establishing effective islands of automation. Clearly, CIM is easier to achieve if these islands are designed to communicate with the rest of the plant from the beginning.

There are two major concerns in building effective workcells. The first, and generally the one that receives the most attention, is connecting the cell controller to the shop floor devices that perform the work. The second concerns the linking of workcells together in groups or areas.

HP computers commonly used as cell controllers are the HP 1000, 9000/200 and 300, and the Vectra PC. Which machine is best is a function of cell complexity, real-time needs, programming expertise and interface flexibility.

By far the most common interface to shop floor devices is EIA RS-232. Each HP computer used in cell control provides basic link level interfaces to this important standard. While not as common as RS-232, IEEE 488 (HP-IB) is more important in certain kinds of cells, such as product test and data acquisition workcells.

Once a group of workcells has been set up, users often connect the cells and establish an "area management" function. Area management is implemented to collect data from cells, store and control cell software releases, manage program maintenance and development, and to provide shared resources (discs and printers). Workcell clusters can be created in several ways: by a local subnet that links the cells with a locally owned and operated area manager, or by connecting the cells to a plant-wide backbone to access area management resources in another department or plant data center. HP AdvanceNet offers both options.

While the HP 1000 is an important cell controller today, the future of HP cell controllers is in UNIX systems. When coupled with X-Windows and low-cost PCs, UNIX will offer a powerful range of cell control solutions.

As MAP becomes increasingly well established, a low-cost link for subnets, called carrierband, will link clusters of cell controllers. Cell controllers, the equipment they control and area manager systems will be connected via this low-cost MAP subnet and gain backbone access via a carrierband-to-broadband bridge.

Computer Center Module

Getting more from a data center is easy with IEEE 802.3 ThinLAN subnets for HP and DEC equipment and SNA products for connecting to IBM and compatible mainframes. The HP ThinLAN Hub connects HP 3000s via ThinLAN coax. In addition, HP Network Services provides the capabilities to improve information access, reap the rewards of resource sharing, improve utilization of processors and communicate with other HP systems around the plant.

The future will bring more OSI/ISO networking in the form of direct MAP connections to the backbone or a subnet using TOP (Technical Office Protocol) connected via a bridge. We also expect that UNIX will begin to play a role in data center computing for off-line area management and, evenutally, more traditional applications.

Plant Area Management Module

This is the heart of HP's CIM solution. The major benefit of CIM is information access and integration, and HP AdvanceNet provides it plant-wide and among multiple vendors.

For industry-standard multivendor networking, MAP is available on some HP computer systems now, and will be on all of HP's factory systems in the future. For multivendor applications in production engineering, and for links to product design, HP offers

industry-standard ARPA and Berkeley networking services for all our UNIX workstations and systems. While the ARPA/BSD services will remain important for some time, TOP will become increasingly important, integrating production and design easily into the MAP manufacturing.

Company-Wide Access Module

Here's the module that keeps a manufacturing plant in touch with the outside world. When it's important to communicate with headquarters, suppliers or customers, the alternatives within this module offer different options for doing so. In addition, HP AdvanceNet provides worldwide electronic mail, even if a customer has an SNA company-wide backbone network.

The future again includes OSI/ISO, specifically the X.400 Message Handling Services (MHS). MHS will help provide a standard foundation for Electronic Data Interchange (EDI) for connecting the factory directly to suppliers and customers.

CONCLUSION

HP's CIM networking solution is divided into modules, each of which addresses a specific aspect of the manufacturing process. But it is important to note that the overall CIM solution is created by the integration of these modules. This modular structure enables users to implement their CIM solutions in manageable, affordable stages, at the rate that is ideal for them.

In the future, UNIX and DOS, linked with MAP and TOP, will become the dominant operating systems in HP's CIM solution.

CIM is an evolving technology that has the potential to deliver dramatic improvements in manufacturing productivity and efficiency. HP has the networking experience and expertise users need to put this technology to work for them.

Overview: Business Graphics and Desktop Publishing

Ames Cornish
Hewlett-Packard
3410 Central Expressway
Santa Clara, CA 95051

This paper will give a brief overview of how Hewlett-Packard is addressing business graphics and desktop publishing. Desktop publishing (DTP) is the fastest-growing new application in the PC industry. Many consider desktop publishing to be the first truly new application for personal computers since the invention of the spreadsheet. DTP has generated such excitement with good reason -- the personal computer can vastly improve the quality, clarity, readability, and impact of office documents. Aside from reducing typesetting costs and turnaround for those graphics artists who are currently using older methods, DTP is allowing a whole new category of people to reap the benefits of high-quality document production.

This new category of users consists of managers or professionals in medium-to-large organizations. These users have typically been using less sophisticated word processing, and spend more than half of their time on written and verbal communication to others. Small businesses are also taking advantage of desktop publishing to improve the image that they present to customers and clients, so that they can seem bigger than they really are.

Who Uses DTP in the Office?

- **Manager or Professional**
- **Medium to Large Company**
  OR
  Small Client-Service Firm

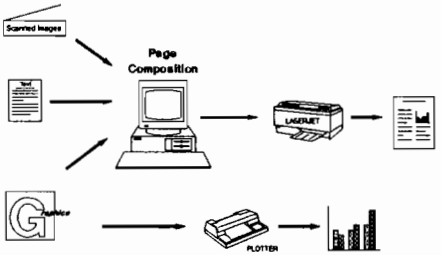
These same people can use business graphics to vastly

improve the quality, clarity, and impact of their presentations. Both business graphics and desktop publishing use the power and interactive nature of the personal computer to make communication more effective, more professional, and more approachable.

HEWLETT-PACKARD'S STRATEGY

Hewlett-Packard is addressing desktop publishing and business graphics by offering a complete solution spanning peripherals such as the LaserJet family of printers, to PC's such as the Vectra PC, and to software such as Graphics Gallery. In providing for the needs of the office user, Hewlett-Packard is focusing on four major factors.

Hewlett-Packard delivers DTP



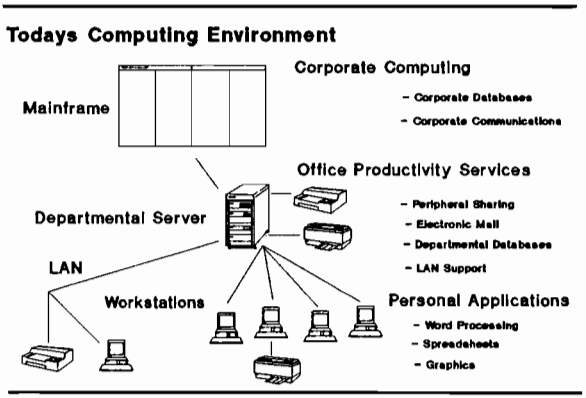
First, Hewlett-Packard is committed to providing solutions that can be easily integrated into your current environment and easily integrated with each other. This implies a host of features, such as support of IBM and IBM-compatible PC's, quick and easy transfer of graphs from Lotus 1-2-3 to Gallery, support for third-party software such as word-processors and page-layout products, and easy access to database information on host systems.

Second, the quality of the final result, whether it is an overhead transparency, status report, or customer flier, is the single most important factor to the user. Hewlett-Packard addresses this with high-quality output devices such as the LaserJet printer and Hewlett-Packard plotters, and by providing software such as Graphics Gallery that produces the most professional-looking results possible on these devices. For example, Gallery provides scalable, professional-quality fonts and picture libraries.

Third, Hewlett-Packard is making these tools available to the office worker, not just the graphics artist, by focusing on ease-of-use and ease-of-learning. Hewlett-Packard is providing pull-down menus and mouse-driven interfaces in

products such as Graphics Gallery and Microsoft Windows.

Fourth, Hewlett-Packard will let you take advantage of the power of your PC's. While minicomputers and mainframes are great for storing corporate databases and linking remote sites, PC's are most appropriate for the productivity applications such as DTP, word processing, and graphics. Compared with terminal-based applications, productivity software on PC's: is more interactive than a terminal connected to a minicomputer; provides more reliable response time; is more easily expandable (it's easier to add PC's than to incrementally expand minicomputer resources); and provides better integration with other PC productivity tools such as Lotus 1-2-3.



While productivity applications should be based on the PC, it is still critical to provide host/workstation compatibility for Electronic Mail and remote printing. This allows users to easily send and receive text, spreadsheet data, graphics, and even complete DTP documents. For example, Hewlett-Packard's Graphics Curator/3000 will automatically translate between PC and minicomputer graphics formats so that PC graphics can be sent through electronic mail, displayed on a terminal, and printed on a minicomputer laser printer.

FUTURE DIRECTIONS

As desktop publishing technology spreads throughout the office, Hewlett-Packard sees it evolving towards what we call "integrated desktop publishing", or IDTP. Whereas many DTP users today are using an isolated system consisting of a laser printer connected to a workstation, an office worker in a workgroup needs to be connected into the rest of his organization's office automation. IDTP will need to offer solutions for:

Access to data: Office workers creating a report would like easy access to corporate databases, local spreadsheets, and standard document formats for quick creation.

Shared editing: Often more than one person will contribute to the creation of a document. For example, a manager may create the content using a word processor, a graphics professional may design the layout, and a clerical worker may combine and publish the final document.

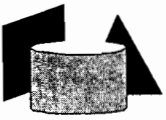
Shared peripherals: The more expensive peripherals used for final printing tend to be shared devices. Thus it should be possible to print any DTP document on a variety of remote peripherals from a local workstation.

Distribution: For review copies and sometimes even final results, users would like to send documents electronically so that they can be read or printed by other users.

Hewlett-Packard will add value to our current industry-leading DTP components by linking them into an overall IDTP solution.

SUMMARY

In sum, both business graphics and desktop publishing are popular with office workers because they improve the efficiency and power of communication. Hewlett-Packard is committed to providing industry-leading solutions that integrate effectively with the range of computer tools used by the office worker.



Integrating Graphics and Desktop Publishing

Ames Cornish
Hewlett-Packard
3410 Central Expressway
Santa Clara, CA 95051

This paper will describe the role of graphics in desktop publishing, will discuss the different types of graphics commonly used in desktop publishing, and will discuss what to look for when you are selecting graphics for desktop publishing.

WHY GRAPHICS?

Desktop publishing (DTP) is used when the presentation and appearance of your document is critical to grab the attention of your audience and effectively communicate your message. When communication is so important, of course you want to use graphic illustrations to improve the clarity, readability, and impact of what you're doing. Graphics is an essential part of DTP.

However, most DTP page-layout software does not include the capability to create illustrations! Just as you need a separate word processor to create text before it can be brought into the page-layout software, you also need a separate graphics package to create the illustrations that you wish to put on a page.

If you have seen the sample documents shown by vendors to show off desktop publishing equipment or software, chances are it was a fancy newsletter that contained some sort of stunning illustration. However, you are seldom told how that graphic was created. Typically, it was done by a professional graphics artist using a paint program, and it took anywhere from a few days to a several weeks to create that one picture! The good news is that there is an easier way. Let's discuss the different methods for creating DTP graphics one by one.

SCANNED IMAGES

Often you will wish to include an image currently on paper form into an electronically published document. These

images can be just about anything -- a photograph of a person, a technical drawing, a picture from a magazine, a photograph of your product, or even a logo. To do this, you should use a scanner. Typical scanners available today for DTP users will scan images at 300 dots per inch (dpi) (to give the maximum quality on a 300 dpi laser printer).

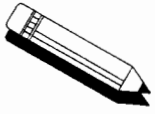
Some scanners or scanning software will also give you the ability to convert a scanned image of text into an editable format. For example, you could scan in a letter, edit it in a word processor, and and then reprint it. Usually this capability (called OCR for Optical Character Recognition) is an extra cost, and may not fully meet your needs. For example, OCR will often work only on a limited number of fonts, and the particular page you wish to input may be in a different, unsupported font. If you're looking for OCR, be sure to pick and choose carefully.

Most DTP users today, however, are using scanners to add images to their documents. Hewlett-Packard offers the ScanJet scanner and Scanning Gallery software to bring this capability to DTP users who need high-quality pictures that are easy to create and scan.

ScanJet Features

* **Advanced document handling**
  - **Flatbed design**
  - **Optional 20-page document feeder**
* **Three image scanning modes**
  - **Binary**
  - **Dithering**
  - **Gray scale**

When shopping for a scanner, you should look for these features:

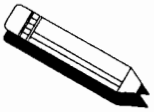


Document handling: Can the scanner take images from books and small sizes of paper as well as standard 8 1/2 by 11 sheets? A flatbed scanner will give you more flexibility in the type of document you can scan. A sheet-feeder will let you scan in a number of standard pages automatically.

Gray-scale support: To properly scan in photographs and half-tone images, your scanner needs to be sensitive to various shades of gray. A dithered image will simulate

shades of gray on a black and white device such as a LaserJet. A true gray scale image will allow more flexibility in editing and scaling, and can usually be converted to a dithered image when printing.

You will also need software to use the scanner. Sometimes this software will be included with the scanner, other times you may need to purchase it separately. Here are some things to look for in scanning software:



Resolution: Can you select the resolution of your image before it is scanned? This will give you more flexibility and higher quality if you are printing to a device that is not 300 dpi.

Scaling: Can your image be enlarged and reduced both before and after you scan it? Also consider the quality of the image after scaling -- do you lose your shades of gray when making an image smaller?

Preview: Ideally, your software should allow a quick "draft" scan, so that you can see the layout of the page. Then you should be able to select the area you wish to scan in detail by simply drawing a box around it.

DTP compatibility: Does the software create image files in a format which can be read in by your page-layout software?

Scanning Gallery Features

* User selectable resolution for maximum output quality (75 - 300 dpi)
* High-quality scaling / image enlargement and reduction (50% - 200%)
* Preview scan for precise electronic clipping
* A true "Windows" application
* Images compatible with popular DTP software

Scanners are an excellent way to include images that are currently on paper. However, if you wish to create new illustrations on your computer, you will need additional graphics software, such as a painting program or business graphics package.

PAINTING PROGRAMS

Painting programs, or raster editing programs, are the most simple and basic form of graphics application. Painting programs are typically used to create the extremely detailed and creative graphics in sample documents used in DTP advertising. You will need to consider whether you need a painting program or a business graphics (vector-oriented) drawing package.

Some of the benefits of a painting program are:

Full control over image: A painting application will let you individually specify each dot, or "pixel" in the image. This is good for graphics artists who wish to edit down to a fine level of detail.

Raster effects: There are some interesting raster effects, such a "spray-can" effects, that can be done easily in a painting package, but probably not at all in a vector-oriented drawing package.

Low cost: Because it is relatively easy to design a painting application, painting programs are usually available at a low cost. Many are even bundled with other products such as mice, computers, or window environments.

However, there are also some severe limitations in a painting program versus a vector-oriented package:

Requires artistic skill: To create a good-looking image in a painting program typically takes a great deal of time and skill.

Restricted editing: Objects cannot be edited independently of each other. For example, if you have added text that overlaps a box, you can erase the box and the text, but you cannot just delete the text or just delete the box! Also, once text has been entered, it cannot be edited or changed!

Limited scaling: Once an object, such as a circle, has been created it cannot be easily scaled. For example if you wish to make a circle bigger, you will also make the "jaggies" in the circle bigger, resulting in a poor-quality image.

Poor resolution: Many paint packages are limited to the resolution of the computer display you are using. When you wish to output this image to a laser printer, it is still in the low-quality resolution of the screen, not the high resolution you normally expect from a printer. Some more expensive paint packages will let you create and edit images in a higher resolution than your display.

In general, a painting package is inadequate for most DTP users. The exceptions are professional graphics artists who are willing and capable of the planning and detail work required to create an image in a painting package. For most users, a business graphics package is more appropriate.

BUSINESS GRAPHICS

A business graphics package can consist of many components. The most basic elements are pie, bar, and line charts. More flexible packages will also let you create text slides (for overhead transparencies), org charts, and arbitrary diagrams and illustrations. Hewlett-Packard offers the Graphics Gallery Collection, which includes all of these capabilities.

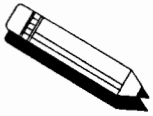
The Graphics Gallery Family

- **Charting Gallery**
  - Creates Pie, Bar, and Line charts
- **Drawing Gallery**
  - Creates Text slides, Organization charts, and Illustrations
- **Gallery Portfolios**
  - Over 1800 pre-defined pictures and illustrations
- **Executive MemoMaker**
  - Managerial word processor with merged text and graphics
- **Gallery Collection**
  - Includes Charting, Drawing, and additional Portfolio

Of course, if you wish to include these graphics in a DTP document, make sure that the business graphics package can create files compatible with your page-layout software!

Using a business graphics package for desktop publishing will let you use the same charts in different places. For example, if you have created a new org chart, this can be put on an overhead transparency for a presentation, and then later included in a datasheet you have created with DTP!

When looking for a business graphics package for DTP you should look for:



Picture Libraries: Picture libraries ("clip-art") are essential to creating high-quality illustrations quickly. Often, an appropriate illustration can be taken directly from a picture library, rather than created by hand. You should look for a wide variety of pictures, pictures that illustrate themes relevant to your needs, and professional-looking illustrations that will enhance the quality of your DTP documents.

Spreadsheet Integration: Most business graphics software will let you read in DIF data files that have been created from spreadsheet software. In addition, if your organization uses Lotus 1-2-3, you should expect your business graphics software to read Lotus 1-2-3 graphs directly from the worksheet file. This will let you quickly turn a rough looking analytical chart into a professional-quality chart appropriate to your DTP document.

Both Charts and Diagrams: Your business graphics should be able to create both data-driven charts such as pie, bar, and line charts, as well as diagrams, org-charts, and text. Ideally, you should be able to mix data-driven charts with customized text and pictures all in one illustration.

Output Quality: High quality output is essential for use with DTP. Your graphics package should have a number of features that will create a better-looking, more professional result such as a variety of high-quality filled fonts, automatic simulation of colors with shades of gray, and high-quality picture portfolios. While you're at it, also consider the quality of output on plotters for overhead transparencies. Remember, you'd like to use the same graphics for both presentations and desktop publishing!

DTP Integration: It goes without saying that you need your graphics package to be compatible with your page-layout software. However, many graphics packages are not compatible with the leading page-layout software! There are also different ways in which this compatibility can be achieved: a) Screen capture is a quick-and-dirty way to get pictures. Unfortunately, it tends to give poor results because the picture is limited to the low resolution of the screen, rather than the high resolution of a laser printer. b) Raster file transfer can result in the highest-quality results because the graphics software will specify exactly what each pixel in the final output should look like. On the other hand, it is relatively difficult to change the size of a raster picture after it has been brought in to your page-layout package. Common raster file formats

include TIFF and PC-Paintbrush. c) Vector file formats tend to result in slightly poorer-quality illustrations, and not quite "what-you-see-is-what-you-get". This is mostly because the current de-facto standards are inadequate for high-quality pictures. The major benefit of a vector format is that the picture can be easily scaled even after it has been brought into the page-layout software. Some vector file formats currently used are Autocad's "DFX" files, Lotus 1-2-3 "PIC" files, and HPGL files.

WYSIWYG: Your graphics software should provide "what-you-see-is-what-you-get". For example, you should expect to see the same fonts and text sizes on-screen as you get when printing on a Laser printer or plotter.

Ease of use: Of course you also wish to minimize the amount of training on graphics software and the time spent creating graphics. You should look for mouse input, pull-down menus, and other features to make the software easier to use.

Graphics Gallery Advantages

- ☞ Quality of output
- ☞ Complete range of output
  - Full color plots and slides
  - Excellent printed graphics
- ☞ Integration
  - Charting and drawing
  - Desktop publishing software
  - Lotus 1-2-3 spreadsheets
- ☞ Ease of use

SUMMARY

In sum, graphics is an essential part of desktop publishing that is often overlooked until after you begin to use your system. For most people, a combination of a scanner and a business graphics package is ideal. Be sure that both will create graphics files that can be used by your DTP software.

Systems Development in an Imperfect and Changing World

Marcia Shonnard Clarkson
The University of the South
Sewanee, Tennessee 37375
USA

This paper first will identify recent changes that affect the way we develop information systems and the nature of the data and functions we include in them. Next it will describe some problems our information systems departments face in meeting the needs of our users. And finally, it will suggest how we in the Hewlett-Packard community can meet the changes we face, provide high quality service to our users, and develop more effective information systems.

We must cope with changes affecting both the systems being developed and the systems development process. These changes are coming from Hewlett-Packard, from the software industry at large, and from within our own organizations. Four important changes which we must all address are the following:

1. Hewlett-Packard Precision Architecture. The expanded offerings from Hewlett-Packard will complicate as well as offer new opportunities for system development. MPE and Image as we know them today are just a subset of the software environment on the Spectrum series. The new precision architecture systems feature two operating systems, HP-UX and MPE, and a database management system called ALLBASE which is much more comprehensive than Image 3000.

Because of its new features, ALLBASE will especially have an effect on the systems development process. ALLBASE is made up of three components: DBCore, HPIMAGE, and HPSQL. DBCore, the internal component used by both HPIMAGE and HPSQL, performs the basic DBMS functions of data definition, data access, transaction management, concurrency control, logging and recovery, and accounting. In DBCore data are stored in the form of relations, and DBCore allows three kinds of indexes to be built on these relations: b-tree indexes, parent-child relationships, and hash indexes. DBCore also supports linked lists or threads that allow the higher levels of software to define their own access order. HPIMAGE supplies the user with a network model interface to the data managed by DBCore. This network interface is similar but not identical to the interface we are currently using in Image. HPSQL gives the database designer a relational interface to the data managed by DBCore.

According to the Hewlett Packard Journal, HP Precision Architecture, offers HP customers "the choice of either a network or a relational interface from one DBMS."[1] With these new software options the Spectrum should appeal to a much larger segment of the data processing community. For many of us, however, the additional options, especially the options available with ALLBASE, will make systems design more complex. We are used to thinking of data in terms of Image's masters and details

or in terms of the network model for structuring data. Although other models such as the relational model have been available on the 3000 for some time, we have found it cost effective as well as easy to limit our concept of data relationships to the network model because Image was included in the basic system provided by HP. But now with ALLBASE we will all have two different models as well as multiple indexing schemes from which to choose and we will have a much more sophisticated DBMS package with which to work.

2. End-user Computing. End-user computing will not only change the type of support which our information systems departments must provide but will also affect the systems development process. Many of the people at the terminal end of the cables connected to our HP3000's are, as they would proudly say, "doing their own thing." They are no longer limited to using programs supplied by our professional data processing staff. They may be using personal computers with office automation and spreadsheet software. They may be using fourth generation languages to access databases resident on our 3000's. They may be building their own databases. These users are becoming more knowledgeable about using computers, more dependent on computers in their day to day work, more interested in the software development process, and they have higher expectations for the information provided by our information systems departments. For instance, they may expect us to integrate data stored in databases we develop for our 3000's with the word processing and spreadsheet software available on their PC's.

3. Use of computers by top management. Executives have very different information requirements from our traditional users and they also expect a different level of service. Until the late 1970's most of the direct users of the software systems we developed were clerks and operational level managers. Executives may have looked at our financial summaries, but our systems were not designed to aid them directly in their work. Advances in both hardware and software have greatly enhanced our ability to build information systems for use by those above us in the organization chart as well as those below us or at the same level. Not only must we be more responsive to the president of the organization than to other employees, but we must recognize the need for a different kind of response. He or she performs different activities and requires different types of information and different means of accessing the information than we are accustomed to providing.

Executives are concerned with the organization as a whole so are interested in data that cross the boundaries of many traditionally separate systems. Data from marketing, manufacturing, and accounting systems must be combined to provide the information that executives require. Not only do we have to integrate many existing systems to provide information for top management, the questions executives ask cannot be predetermined because their activities are not routine. We do not have two months or probably even two days to write a program for them. The output generated by our programs must also change. Reading pages and pages of computer printouts is too time consuming; executives often prefer a graphic presentation of data.

Decision making and planning are the key activities of executives. Our traditional sort, select, calculate, summarize, and print programs which take weeks to develop are no longer sufficient. Executives need systems designed for decision making, systems that provide for modelling and simulation of the events in an organization based on the data in the database, and systems that provide an adhoc reporting capability. If these executives are also end users, the software they use must be very user friendly, without cryptic little error messages like "illegal access." This executive user, then, needs a different type of software and we must use different procedures for developing that software if we are to serve him or her effectively.

4. Advances in software development methodology. To meet the increasing demand for our services we must take advantage of new tools for developing systems. The traditional software development life cycle and the COBOL programming language are no longer the only means for developing information systems. New procedural languages such as Pascal and non-procedural fourth generation languages can reduce the time required to develop programs, and new methodologies and products can aid in designing information systems.

Not only has the environment in which we are developing systems changed, but our information systems departments are sometimes failing to meet the needs of our users. Criticism is aimed at our data processing personnel, at the services they provide, and at the products they produce.

Some problems with users involve attitude more than accomplishment but these problems are nonetheless real. Data processing personnel are perceived as over payed and inarticulate. Many of our best people are technological prima donnas or introverts or both. Their image is that of a narrow technician not in sync with the goals of the organization.

More justly users complain that the applications we develop are late and over budget. When the systems are finally complete, they do not perform as anticipated by the users and they are unreliable. We are fortunate in the reliability of Hewlett-Packard hardware and systems software. But many of us have unreliable communications equipment and imperfect applications programs that frustrate our users.

Our problems with users may be further complicated because data processing departments have traditionally been aligned with finance both in terms of reporting structure and in terms of systems produced. The type of systems required and the services expected by accountants are often different from other users. Accountants too have been criticized as narrow technicians and the systems they require are not as dynamic as those required by marketing or even personnel. We must adapt to our new users' requirements for products and services and we must see data processing departments as service departments which are central to the information revolution that is overtaking our institutions.

How then can we adapt to our changing world and better support the organizations in which we work? I suggest that we must attempt the following:

1. We must change our way of approaching data and information. Poorly organized, incompatible collections of redundant data fill our tape and disk libraries. We must begin to think of data as a corporate resource, not as the property of any particular division of the institution. There must be a strong, centralized, high level control of the data. We must also expand our ideas of the data required. To the data necessary for the day to day operation of the organization we must add demographic data about our customers and location, data about our competitors, and other data necessary to aid strategic planning and decision making, the activities of executives.

Data are such a significant resource that many systems designers are beginning to think of information systems as being data driven. That approach succeeds because the basic data entities in an organization remain the same unless the enterprise itself changes radically. By contrast, the information processing requirements, especially those of executives, are unstable, sometimes changing from day to day. If the center of our systems is a comprehensive, stable, easy to use database, then processes that turn that raw data into information can be changed without disturbing the system's foundation.

2. We must change our approach to developing information systems. First, database design must concentrate on the logical description of data, possibly using the entity-relationship model. Second, we must reevaluate traditional system development methodologies and investigate information engineering, prototyping, fourth generation languages, and tools to aid the systems analyst. Finally, we must improve our communications with users to enable them to assume a leading role in developing the systems they will use.

Database design is traditionally broken down into two phases--logical and physical. In the logical phase of design we think of data in terms of entities, attributes of entities, relationships among entities, and in terms of the constraints on those attributes and relationships. We also determine how various users view the data. During this phase of design we must consider what data mean and how data are used. The physical phase of design aims to implement the logical design in the form of a particular DBMS product such as ALLBASE.

Too often the logical phase of design is neglected. We concern ourselves with the data storage and access methods of the physical model rather than concentrating on the entities, the relationships among entities, and the ways users view and access the data. Until recently our options for data storage and retrieval on the 3000 were so limited that early in our systems study we naturally began thinking of data as Image masters and Image details, and we even started planning access methods. An early commitment to physical implementation is not only unnecessary but improper with a product like ALLBASE. Neither users or

applications programmers should be concerned with the physical implementation of the logical model. The stable, corporate data model on which all systems are built should be easy to understand and independent from its physical implementation.

The entity-relationship model for describing the database in logical terms was first proposed by Chen in a landmark article of 1976[2]; I believe it can be very useful for us. Chen proposed the entity-relationship diagram as a tool for logical database design and described how the relational and network models can be derived from the entity-relationship model. He also described the steps necessary for logical database design. Appendix A uses a subset of a student information system to describe the logical design process proposed by Chen and also provides an entity-relationship diagram for a subset of a student information system. The advantage of Chen's entity-relationship model is not only that it can be mapped to either the network or the relational structures supported by ALLBASE, but also that its method for logically describing data and their uses can be easily understood by both systems developers and users and can therefore make it easier for the users to participate in the design process.

James Martin in An Information Systems Manifesto[3] describes information engineering as "the set of interrelated disciplines which are needed to build a computerized enterprise based on data systems." Although information engineering, like many other data processing terms, means different things to different people, I believe the common ideas of many of its proponents can be very useful to us as we develop information systems. The emphasis on information suggests that systems should be data driven, and the emphasis on engineering suggests that the design process used by systems analysts should resemble the design process used by engineers. Seeing systems as data driven and putting more effort into developing a comprehensive, logical, data based model of an institution may help us to build more stable systems that meet the volatile information processing requirements of top executives. Information engineering proposes such a corporate, strategic data model as the foundation for all information systems and proposes an engineering approach for developing software that effectively uses this model.

Information engineering may help us build more stable systems and the entity-relationship model may help both data processing professionals and users better understand the nature and function of corporate data. But, to satisfy the concerns of users, we must speed up the development and cut the costs of information systems. Programmers and analysts are expensive, and the traditional processes of systems design and programming in third generation languages are very slow. Many tools are available today to aid in systems design and fourth generation languages can help to automate the job of programmers.

Fourth generation languages are not only the first step in automating the work of programmers, they also allow designers to build a prototype system in the early stages of system development. For years we in data processing have complained that users do not know what they

want from an information system. This is both true and understandable. Engineers build prototypes or small scale models for cars and airplanes. Architects build small scale models of buildings. Why should information system users be more imaginative than automobile manufacturers or developers in anticipating the finished product? And why should users commit funds to an abstract system without some idea of the product they will receive? In some ways users must experience the system before they know what they really want. If in the early phases of system development we can build a prototype using the logical database and let both the professional designers and the users experience the system, then both may see and avoid problems that would have been very expensive to correct later on.

If we are to develop systems in a more timely manner, not only must we use fourth generation languages to automate the job of the programmer and to aid in system design by allowing us to build system prototypes, we must also automate other aspects of designing systems. Data models as described above are too complex to design, draw, and update by hand. Design tools are being developed to help the designer create the model, to check the model's accuracy, and then automatically to develop a data dictionary.[4] These tools combined with fourth generation languages will be a big step in automating the jobs of analysts and programmers and so speeding system development.

3. We must organize our data processing departments to better satisfy our users. Information centers where users can obtain advice are becoming popular in many organizations. Exactly how we educate and advise our users is not important. But it is important to have a central control of end-user computing, and important that end-users have consultants available who can help them quickly. If we do not maintain control of all hardware and software being used in our organizations, we will be faced with incompatible equipment and data. And, if we do not educate our users of powerful tools such as fourth generation languages on our 3000 systems, we will have HP hardware that is bogged down with such tasks as unnecessary serial reads of databases. Providing this kind of service to users is in many ways incompatible with the work of the professionals who are developing new systems or working on complicated projects where interruptions by end-users are distracting. Although many of our shops are too small to support an information center, we must not lose site of the need to provide an efficient and effective way to support end-users while at the same time providing a good working atmosphere for the professional data processing staff working on more traditional programming and design projects.

We must also invest more of our data processing funds in educating our professional staff. Today's computing environment is complex. The new techniques for developing systems discussed above make education an even more important investment than it has been.

Traditionally information systems departments have been the proponents of change in organizations. We introduced the systems approach and automation and we have been at the forefront of the information

revolution. But now it is we who are resisting new methodologies and automation of our work, and it is we who are not keeping up with the changes in our industry and in our organizations. Computer produced information is now a key resource to even the top executives in organizations which makes it even more important for us to meet the changes we face, provide high quality service to our users, and develop more effective information systems.

References:

[1]Brown, Alan S. "Database Management for HP Precision Architecture Computers," Hewlett-Packard Journal December 1986.

[2]Chen, Peter Pin-Shan. "The Entity-Relationship Model--Toward a Unified View of Data," ACM Transactions on Database Systems Vol 1, No. 1, March 1976.

[3]Martin, James. An Information Systems Manifesto. New Jersey: Prentice-Hall, Inc., 1984.

[4]Freedman, David H. "In Search of Productivity," Infosystems 11/86.

Appendix A: Designing a small part of a student information system using the entity-relationship model and an entity-relationship diagram.

Steps in Designing a Database Using the E-R Model

1. Identify the entity sets and relationship sets of interest:

Entities: students, courses, faculty, student organizations, academic departments, students' parents, dormitories.

Relationships: students in courses, faculty lecturing in courses, faculty serving as course coordinators, students in organizations, faculty as advisors of organizations, students in dormitories, parents of students, students who are married to other students.

2. Identify semantic information in the relationship sets:

Students may take more than one course and a course may have more than one student. Faculty may lecture in more than one course and a course may have more than one lecturer. A course has one faculty coordinator, but a faculty member may be the coordinator for more than one course. A student lives in only one dormitory, but a dormitory houses many students. Students may major in more than one area and a department may have more than one major. Faculty members may teach in more than one department and a department may have many faculty. Faculty

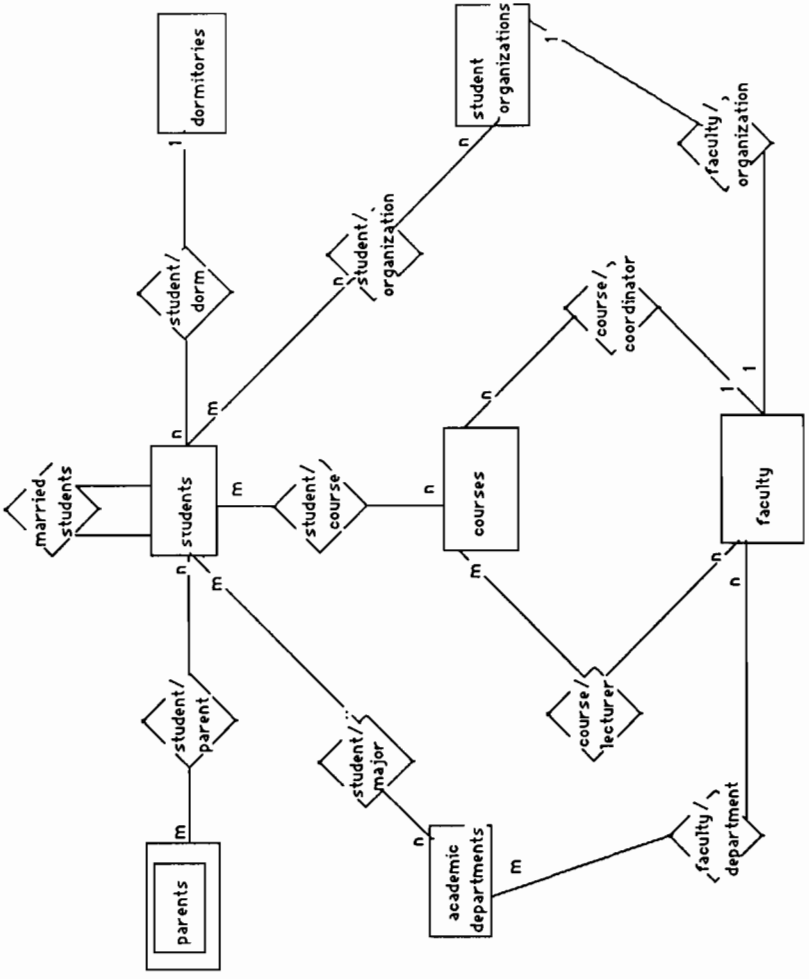
members may advise only one student organization and an organization may have at most one faculty advisor. We are only concerned with the parents of current students. When a student leaves, we want to eliminate his or her parents from our database (parents are therefore called a weak entity by Chen because their existence depends on the existence of a student.)

3. Define value sets, representation, and allowable values:

```
first-name:  character (12)
last-name:  character (12)
no-of-years:  integer (2) [15-85 allowed]
class:  character (2) [fr, so, jr, sr,ir allowed]
course-no:  character(7) [3 letter department code, 3 digit course
              number, one character section code]
course-name:  character(30)
grade:  character(1) [A, B, C, D, F, P, W allowed]
```

4. Organize data into entity relations and relationship relations and decide on primary key values. For each relation describe the attributes of the relation along with the value-set from which each attribute is drawn:

```
regular entity relation student
attribute/value-set:
  name/(last-name, first-name)
  age/(no-of-years)
  class/(class)
primary key: name

regular entity relation course
attribute/value-set:
  course-no/(course-no)
  description/(course-name)
primary key: course-no

regular relationship relation: student-course
  student/name (primary key)
  course/course-no (primary key)
attribute/value-set:
  grade/grade
```



Entity-relationship Diagram for Student Information

CAN YOUR HP3000 MAKE YOU MONEY ?

James Cohen
Mecca Leisure Ltd
76 Southwark Street
London
SE1 0PP

Why has your firm got an HP3000 ?

Most companies primarily bought their original computers to reduce the company overheads. The overhead usually reduced was the one generally regarded as the most expensive - namely Manpower.

These systems are usually of the type:-

Bought Ledger
Sales Ledger
Fixed Asset Recording
Payroll
Stock Control

All these systems are useful in themselves, but their task's are generally restricted to controlling existing overheads, rather than generating additional new income.

The purpose of this paper is to look at a particular system installed at Mecca Leisure. The system's sole purpose is to try and generate income, rather than controlling existing expenditure. The application we use is called "Clubman", for Club Management System, but is purely used for example purposes.

Introduction to Mecca Leisure

Before looking further at the concept of income generation I would like to look a little at Mecca Leisure and its history, as no equivalent conglomerate exists in the USA.

Mecca Leisure has been in existence for just over 100 years, having been founded as a coffee shop in 1884. By 1898 it became a publicly quoted company.

In the early part of this century Mecca Leisure expanded its catering operations both in the City of London and by taking catering contracts in such places as the Crystal Palace.

Through catering Mecca became involved in Dance Halls and by the mid 1950's Mecca had become established as the leading operator of dance halls, in most cities across the United Kingdom.

In 1961, with the introduction of a new Gaming Act, Mecca Leisure became one of the first commercial operators of Bingo in Britain and was soon established as a market leader. Towards of the end of the 1960's Mecca Leisure had started to expand by acquisition notably into bookmaking and other gambling interests.

In 1970 Mecca Leisure merged with Grand Metropolitan (who own Intercontinental Hotels and Childrens' World, in the USA).

Early in 1985 Grand Metropolitan decided to concentrate on its international Brewing, Drinks and Hotels businesses and as a result of this decision Mecca Leisure management bought the company from Grand Metropolitan in a leverage buyout.

The company was requoted on the London Stock exchange, as a result of a public flotation, at the end of 1986.

Lets look a little at the history of Bingo...

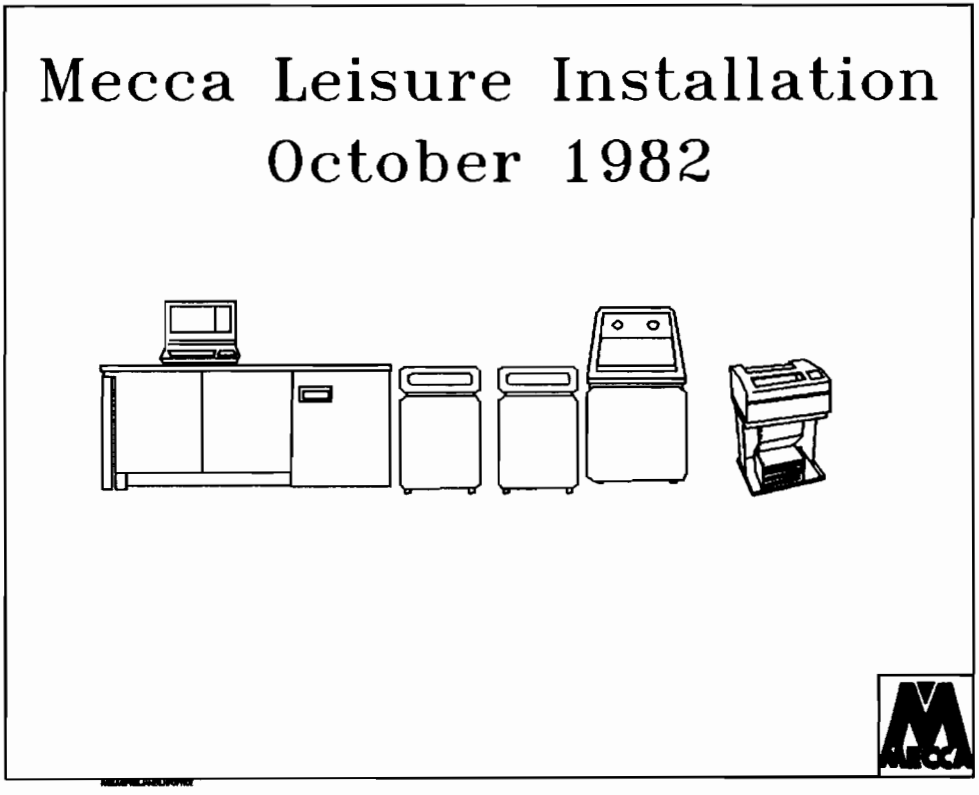
Bingo proved to be a very popular activity, mainly for the over 60's, but as the Gaming Act forbids any direct advertising, this restricts the business from being able to promote itself through the normal advertising media such as newspaper advertising or television advertisements.

Ever since 1961 there has been a legal requirement that all bingo players are members. To become a member, you fill in a form and the next day you could start playing.

In 1979 it was decided, as a marketing decision, to hold this name and address information centrally on a computer bureau, rather than at branch level on paper. To this end a share (stock) registration bureau was appointed to complete this task. Problems relating to the bureau operation are discussed later in this paper.

In 1980 Mecca Leisure was re-organised and a new Finance Director appointed. One of his first tasks was to review the archaic and manpower intensive accounting and administration operations carried out at Head Office.

The package selected happen to run on an HP3000 and this, along with a batch payroll system, went live in October 1981. The hardware selected was a 1 MByte HP3000/44 with seventeen terminals were installed for data preparation and on-line enquiries.



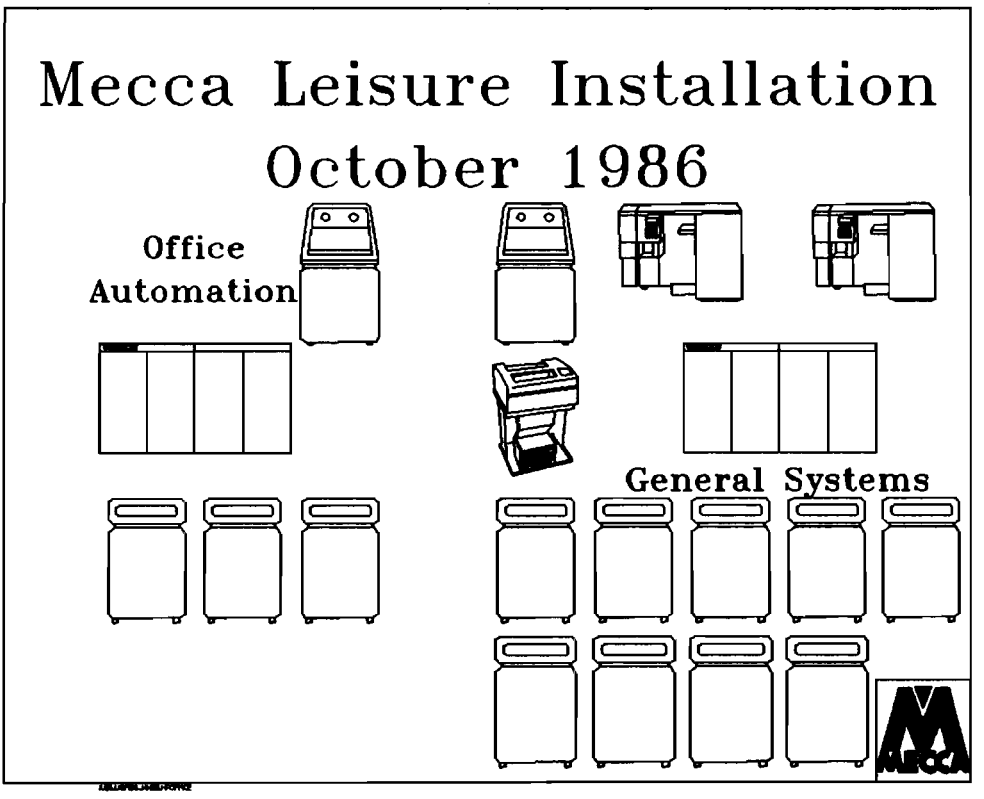
Over the next 18 months these foundations were slowly built on with the addition of one MByte of additional memory and two HP7933 disc drive, as well as an HP2680 Laser Printer.

All developments during this period were user demand driven, but the time was basically a period of consolidation and stabilisation. The only real developments were a small Personnel Management system developed in-house and the introduction of early versions of Word Processing (HPSlate and HPWord using three HP2626W).

During early 1984 the Systems Department had grown to four staff and was ready to start a period of rapid growth. This happened in two directions both which resulted in benefit to the company. One direction was full implementation of Office Automation, the other which has been paying quantifiable returns ever since, is the Clubman System.

Obviously this growth required additional capital expenditure for hardware and this was completed in October 1984. The HP3000/44 was upgraded to an HP3000/48 (for the OA trial) and an HP3000/68 with necessary peripherals installed. The two machines were linked using INP's and DS/3000, this was later replaced by LAN and NS/3000.

With the upgrade of the HP3000/68 to a series 70, the upgrade of the HP3000/48 to a further series 70, the addition of a second HP2680 and HP7937's this is the configuration used currently along with about 150 terminals.



Of the 150 about 120 are HP150's used for Office Automation, Lotus etc, as well as HP3000 terminals.

How can an HP3000 Generate Wealth ?

Clearly the answer to this must vary from company to company. In the case of a leisure company, this means getting more people through the doors, spending more money per visit (ie take per head).

The key question is how ?

In October 1984 the bureau had over 1,500,000 names and addresses, what was not known was :-

- The duplication rate (proved to be 3:1)
- The attendance patterns
- The response rate to mailings
- Even if the member was alive (remembering the high average age !!)

There were a number of problems, caused by the bureau, such as :-

- Limited control of the list
- Since the cost was a factor of the number of records held, there was no reason for the bureau to assist in the reduction of the list size.
- Cost
- Virtually no analysis
- Limited selection criteria for mailing requests

After a detailed analysis of the the bureau system a decision was made to centralise the service onto our own computers, the pay back period was about 18 months. This pay back period was calculated on the current value of the system and before taking added value into account.

Over the following year a specification was prepared, the programs developed and the system went live on 1st October 1985.

The only major problem was the transfer of the data from the bureau and its subsequent deduplication.

The data which was held on a very old ICL 1904 system, the was eventually converted using the program ICLTOHP found in the contributed library. The cleaning of the duplicate and non active members was completed within two months, by removing any one who had more than one membership card and/or had been a member for more than a year and had not played for thirteen months.

Just in reducing the membership with deduplication and archiving membership to the core 400,000 (ie playing membership) saved $800,000 in the first year. This saving came about by reducing on-line storage costs and with savings on unnecessary mail outs.

The database was extended beyond basic name and address, so that additional relevant information is maintained.

I will go into detail later as to the relevance (or otherwise) of the data collected.

However, substantial savings were available by insisting that all new memberships were post coded. (The post code is an 8 byte alphanumeric string, similar in use to the zip code, the post code usually identifies an address to within about 10 dwellings). By post coding the entire lists discounts of up to 35% were available on each mail out from the Post Office.

Other good reasons for post coding are :-

- Speedier delivery
- Use of demographic analysis

Attendance was collected by issuing all members with 'credit card' type membership cards, with their membership number recorded on the magnetic strip. Discs with the attendance recorded on them were returned monthly by the branch, for transfer to the HP3000 where individual members records are updated to reflect which sessions were attended and where (if visiting another branch).

Lets look at the data that is collected.....

Membership Number -	This is a concatenated key of branch number plus incremental number (from one to 99999). This number uniquely identifies the individual.
Base Name Details -	Title, first name, initials, surname.
House Name/Number	
Street Name	
District	
Post Code	There is no need to store town, county, country. This saves about 30 key depressions per address and hence time and over 22 MBytes within the database (remember the post code is unique).
Date of Birth	
Date of Joining	
Male/Female Indicator	
Attendance Pattern Indicator	Show indication of playing patterns, if known.
Star Sign -	Bingo being the type of operation it is. It is popular to mail on sign of the Zodiac!!
Telephone Number -	For follow-up - telephone questionnaire.
Occupation Status -	Full time/Part-time /Unemployed/Old Age Pension.
Marital Status	
Attendance Record -	All the sessions attended.

Profile Code - This code indicates two things, the house type (ie council estate, flat, terraced house etc) and a indication of the occupant likely disposable income. This is allocated by post code and is fully described later.

Lets look deeper into the data collected ...

The base information of membership number, name and address needs no further explanation.

Lets start by looking at the "Attendance Pattern Indicator", each period end we look at every member's attendance record and allocate a flag from 1 to 9 with 0 for no attendance. 1 is a regular attender, 9 has played once.

Why do we spend time updating every individual record ?

Because analysis has shown that it is unlikely that the player, playing two or more times a week can afford to play more often. Therefore the marketing effort is best made in the direction of the member who may play regularly once a month and encourage them to play once a week.

Narrowing down the mailing is best considered as using a sniper rifle as against buck shot.

Make your marketing effort pay.

The star sign while not a very scientific way of marketing has always been successful, send someone a personalized birthday card, with an invitation is highly efficient !

The Attendance Record is perhaps the most critical area. Knowing when members play gives a clear indication of the success (or otherwise) of a promotion and your general marketing effort. By tracking members, watching how often they play is essential in establishing the success of the company. The system's major purpose is to reactivate lapsed or irregular players into playing more often.

To give some indication of the success of the system in this area lets look at what happened:-

In January 1986 the system archived (that is players not playing for 13 months) 20,000 members or 4% of the

database. After a number of promotions (not always to all members) the archive rate by January 1987 rate is down to 8,500 or 1.48% of the database.

Within one year of installation of the system 75,000 more members were playing bingo regularly.

So what ?

Amongst our competitors admissions are down 6% at Mecca they are up 3%. This increase amounts to 25,000 more admissions per month.

Profile Code

This is a code assessed on Post Codes (there are 1,500,000 post codes within the United Kingdom). The code was produced by a research unit at Liverpool University by analysing the last National census and it is made up of two parts. A one byte "housing indicator" and a three byte "Disposable Income Indicator". Together they provide a very accurate indication of the type of person playing bingo.

This field is extremely useful in showing what type of person is joining and playing in our clubs.

How can we further improve System Administration ?

Data is collected from a number of sources, the primary source being members themselves.

On joining they are required to supply the key information on a membership form, these forms are currently dispatched, by the branch, daily by post.

Problems :-

- Unreadable forms
- Incorrect post codes leading to either data preparation delays, or wrong addressing
- Postal Delays/Industrial Action

Solution :-

- Get branches online and make them responsible for their own standards of data input. This will be achieved using micro's and a private X.25 network.

As I mentioned earlier, attendance is collected by swipe reader discs being dispatched monthly.

Problems :-

- Discs get damaged in the post
- Wrong discs returned
- Unreadable discs, usually caused be hardware malfunction
- Excessive elapsed times from swiping attendance, to update of central records

Solution :-

- Replace 5 $^{1}/_{4}$ disc drives with memory boards and modems
- Poll round daily and collect data, then immediately update membership records.

While this solution does involve considerable investment (in excess of $500,000), combining the current swipe reader and current box office system, in a single new system, the new system will additionally provide a superb opportunity to reconcile the admissions to the box office take, thus reducing cash shrinkage.

Now we have collected all this data, how is it going to be used to make money?

To grow a business, and particularly a leisure business, you need to know your customer. The more you know the customer, the better you can service his needs and hence develop the business strategy.

This involves analysis of the membership, lots of it, but beware the pitfalls. Encourage and cajole your users to use sampling techniques, and random sampling. Trying to analyse 750,000 membership records in 'n' directions, can lead to considerable performance problems !

Do not run analysis too often, or the trends may be over looked. For example analysis on a daily basis is less likely to show trends, rather than on a monthly basis.

What sort of analysis is carried out at Mecca ?

Since we have a number of pre-defined goals:-

- Reduce Average Playing Age
- Increase Playing Frequency
- Reduce Ineffective Mailings

Analysis is carried out usually quarterly to analyse any trends in these directions. Since the installation of the system and the change in direction in the marketing effort the average playing age has dropped by two years and the playing frequency has increased by 3%. Mailing costs are down, by target mailing rather than the shotgun approach.

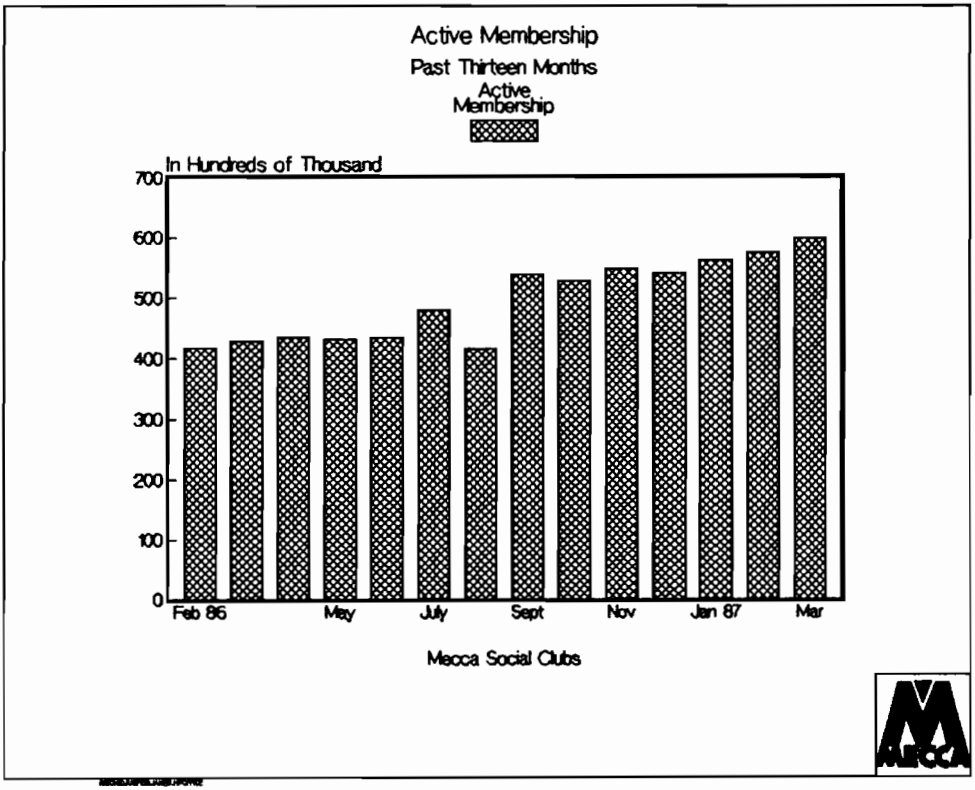
Clearly are large number of factors not recorded on the computer effecting, some seriously, the above aims.

These factors must be taken into account when judging the success or otherwise of the attempt to increase profit.

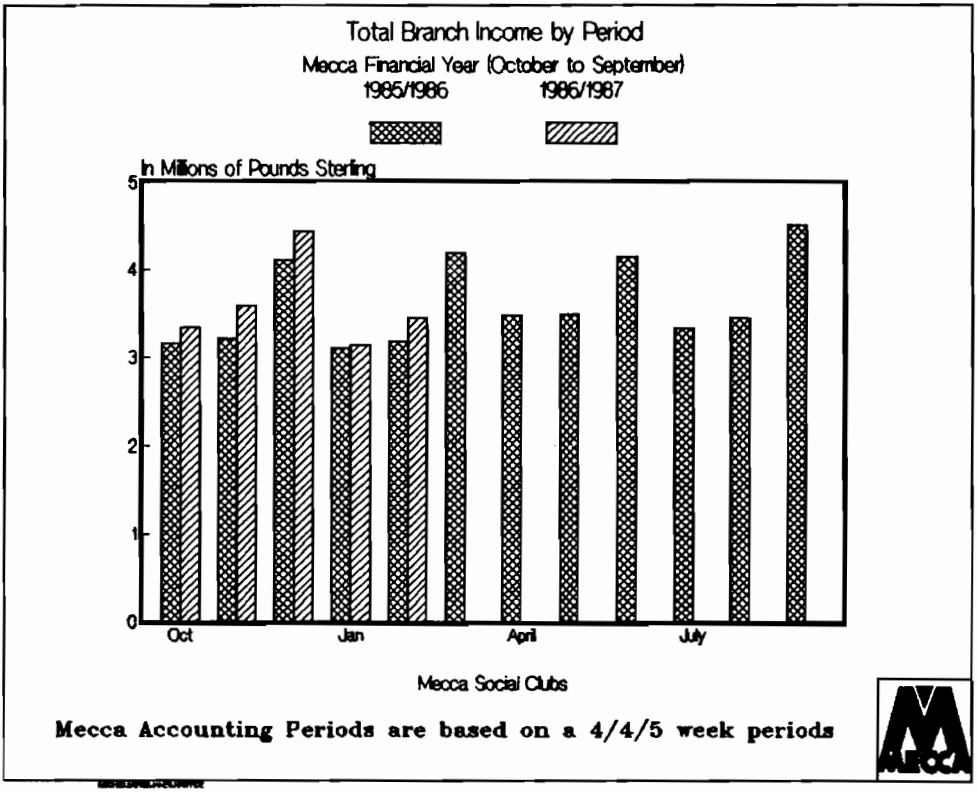
Again using leisure as an example, some of the factors are :-

- Bad weather, people don't go out in the snow
- What the competition are doing
- What promotions are being run
- What is on television, such as World Cup Football

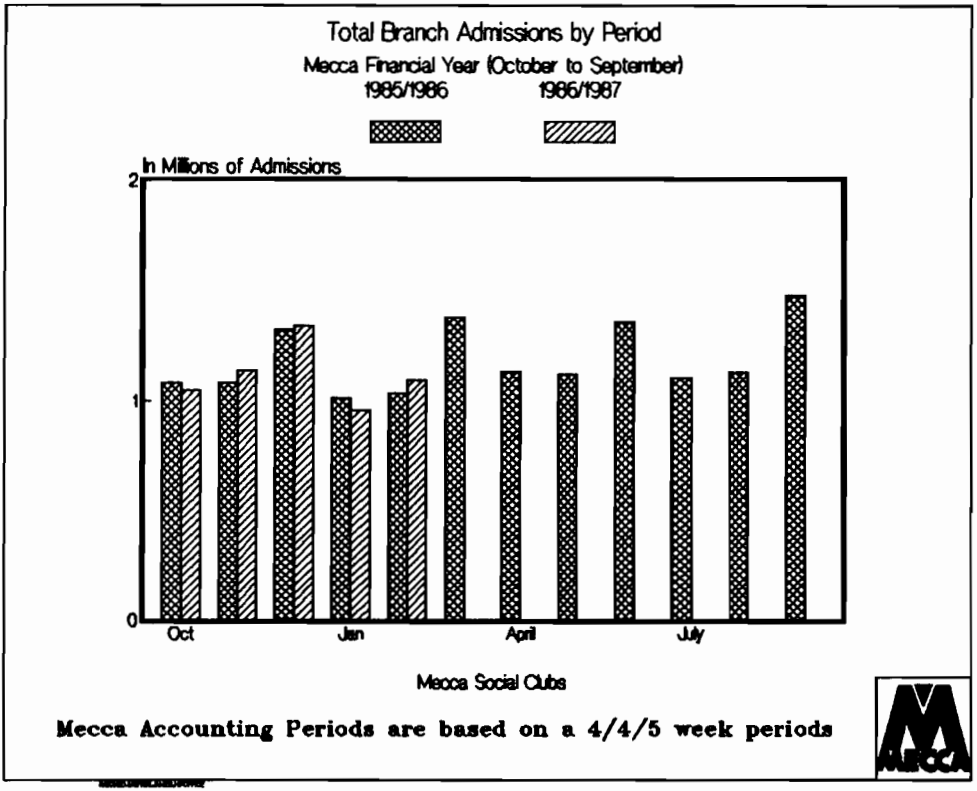
Looking at the graph below we can see the movement in the number of active members and how it has increased during the past year (to March 1987).



Income has increased this year, as against last year. This increase can be largely attributed to the use of the Clubman marketing system :-



Admissions for the past 12 months have improved, compared with last year :-



OUTPUTS, OTHER THAN ANALYSIS

Up to now we have only looked at the inputs and the internal outputs (ie the analysis).

Where the HP3000 generates income is in how it is used to encourage members to visit branches more often, spending more when they come.

While the actual criteria for mailing remains a Marketing Department issue, and beyond the scope of this paper, the mailing request has to be able to select members on as many user defined criteria as possible. This can lead to some highly complex selections such as, members who :-

- Are members of a single branch
- But do not live in post codes YO1 to YO2 and not in post codes YO17 to YO22
- Who are female and over 60 or who are male and over 65 but under 80
- Who played during November, not during December, but did play during January
- Who only play Wednesday afternoons or Saturday evenings only

Even with OMNIDEX installed on the database, many of these requests can only be resolved by serial reads, and then with envelopes in rebate sequence time can be excessive. So excessive that often the ten hours processing available, on an HP3000/70, overnight is not enough.

Therefore, increasingly standard mailing requests are identified and specially coded with resulting much faster run times.

Currently output is produced in one of two formats. Either labels that are machine transferred onto envelopes or two special German made envelope printers that are spooled directly onto the HP3000. These printers feed envelopes through at a rate of over 4,000 per hour each and print up to eight lines of 40 characters.

Printing directly onto the envelope reduces the productions costs to a minimum since it removes the need for labels.

Other output produced for the users are:-

- Top ten attenders (weekly and monthly)

- Membership List

- Archived List

Other then reports and envelopes increasing quantities of personalised letters are being produced, utilising the high quality, high capacity HP2680 laser printers.

Names and addresses, along with another variables required for the letter, are extracted and used as user variable data. These are then printed, trimmed and separated before being folded into windowed envelopes.

A word of warning here, if you do this use TDP rather than HPWord. TDP is about 7 times faster in writing text to the spooler and uses the spooler more efficiently. Also avoid graphics in the text or you will soon run out of disc space as the spooler fills. If you are going to print a signature use IDS/Char rather than a raster image form HPDraw.

DATABASE INTERNALS

Now before looking at some more general topics. Just a few warnings for those of you who have not suffered the excitement frustrations and frights of handling large databases, or for those of you who have moved to TurboImage, but have not had a chance to look at its benefits.

It is not my intention to delve into the database design suffice to say that with a large database (it has handled up to 1,000,000 entries and is stable at about 700,000) traditional chaining is not practical or efficient. A number of fields are concatenated to produce short chains, and the use of OMNIDEX reduces the need for frequent serial reads when selecting members for mail outs.

Just before going into further detail it is essential when dealing with large database (this one is now about 700MBytes) that you do consider some of the management problems you are likely to incur :-

- You must use ILR, with Rollback Recovery
- You should consider dedicating a tape drive or disc drive (in our case a 7914) to database logging since broken chains can be painful !
- When you do need to increase capacities you must have a tool such as Adager, DBManager etc. An unload/load is not an attractive proposition !!
- If you are intending to carry out any batch updating do it over night with auto defer enable (but remember to store before and after and switch it off before the start of the on line day)

ANALYSIS

Analysis, particularly statistical analysis is only as good as the raw data it is given. Initially branch managers did not understand the necessity for clean data. If a new member did not fill in their post code, there was a tendency to create one, rather than look it up.

A large training effort was undertaken to explain the system to branch staff, since the system is constantly evolving and there are staff changes. The training effort is ongoing.

Avoid large samples, but at the same time make sure the sample size is statistically sound. While absolute analysis might be helpful in the ideal world, sampling techniques have to be applied. Otherwise the run times can be somewhat excessive.

We were advised that no sample need be bigger then 23,000 (ask a statistician why!). In the case of a trial we undertook on birth dates, a sample of 10,000 produced an accuracy of 99.3%, compared with a analysis of the entire database.

Avoid over analysis - do not produce reports where samples are over subdivided - this risks reducing the validity of the analysis to almost nil.

For example, if you total on the post code (outer and inner elements) you are unlikely to get a total of more than ten members per post code (since a post code is unique to no more than twelve dwellings). If, however, the outer post code only is used, then since there are only 2,500 in the entire country, the analysis is more meaningful.

ARTIFICIAL INTELLIGENCE??

Currently we depend on the Marketing Department to establish whether or not a promotion is a success and what type of person we are aiming at.

We are currently investigating whether AI products (such as Beagle) can produce "rules" that can be applied to establish the playing characteristics of a typical bingo player.

If we are successful in this, not only does it offer the opportunity to refine further are marketing accuracy, but further reduce mailing costs.

Furthermore it would provide the facility to cull through bought-in mailing lists and only mail to those who match the pre-defined rules.

Currently if a mailing list is bought-in, the entire list is mailed to, if we are able to define some rules the mailing costs could be brought down and the accuracy raised.

These rules are likely to be based on profile code and age, but at the time of writing no firm conclusions have be made. It is hoped to be able to present these 'rules' at the conference.

FUTURES

With the membership growing for Social Clubs and other divisions within the group improved performance is required from the system to ensure rapid response to analysis and mailing requests. The current target of 14 days is too slow in a fast moving business.

Once attendance is collected and updated daily, analysis and further mailing must be able to respond equally fast.

While improved database design could provide some improvements, the major improvement will only become feasible with faster CPU and in particular faster disc I/O.

Hewlett Packard permitting, we will be installing an HP3000/950 during December 1987. Even though the current HP3000/70 is rarely utilised with more than 40% CPU (for marketing applications), the major bottleneck is disc I/O. The additional CPU power along the new disc controllers (rumoured) providing considerably faster I/O, should provided the power needed for the predicted growth in the system.

The improved performance should enable the Clubman system to provide a rapid response to the success (or otherwise) of a promotion.

With the installation of micro's in the branches there will be a speeding up in the data preparation time and enable branches to check online, valid memberships correct addresses etc. More importantly it will provide the branches with a sense of identity with their membership lists.

CONCLUSIONS

Throughout this paper I have been referring to a system developed over the years in house, however this is just an example.

Many companies should be now using their computers beyond the normal "cost control" activities.

Most established users have completed this exercise and you should now be looking to new horizons.

However, since Marketing is not such a clearly defined activity as payroll, you will have to design your system round a few core common features such as post code.

For specific target areas geographically - Post Code

For specific customer type - Demographic Grouping

For specific age groups - Date of Birth

Given that the marketing of your product is the lifeline for your company's future - now is the time to invest in marketing system.

This paper was prepared using the following products :-

HPWord
HPDraw
HPEzchart
IDS/Char
IDS/Form
IFS/3000

ABSTRACT

Extending LANS Over a Wide Area via the 802 WAN

Mike Coker, Vitalink Communications Corp.

LANs have been successful on a local basis due to their low cost, their ease of use and the ability for a wide range of devices to share the network. Now companies want to connect LANs together across the enterprise. The 802 WAN is the vehicle to accomplish this because it is transparent to the user and it is designed for all protocols which conform to the 802 standards to use the network. It does this by listening, learning, filtering, and forwarding LAN traffic across a wide area network using the Vitalink transparent data link-layer bridge, TransLAN. TransLAN logically extends the LAN cable across the wide area network. TransLAN bridges interconnect LANs regardless of differences in station capacity, network architecture, or higher-level protocols. TransLAN provides data transmission capacities of 9.6 kbps to 2.048 Mbps across virtually any terrestrial or satellite transmission link. Two companion products, TransLINK and TransSDLC provide 802 link-layer interfaces to bit-synchronous circuits and IBM 3270 devices to enable SNA, X.25, and HDLC LANs to share the 802 WAN.

PROGRAM TESTING - THE FORGOTTEN ART?

by
Effy Crawford,
VRS Consulting Inc.,
4676 Admiralty Way,
Suite 206,
Marina del Rey,
California 90292

1. Introduction

Ask a programmer to list the tasks he hates most and he is likely to respond with cleaning the car, doing the dishes, testing programs - not necessarily in that order.

The task of program testing is undeniably the most tedious, error prone and unpopular activity in a software project. On medium to large scale projects approximately 50% of time expended on coding a program should be spent on program testing.

However, as more powerful source languages enter the market, the amount of object code generated per line of source code is increased. The relative amount of time spent on actual programming is dramatically reduced while the time spent on software testing will probably increase beyond the 50% rule-of-thumb figure.

This paper will discuss briefly some testing methodologies, although it will not go into any in detail. It does however strive to remind those involved in a software project of the testing tools available and the standards which can be reasonably applied.

In most MIS departments, a very tight schedule for program development is set but with relatively no imposition of standards or discipline. The checking out of a program is left to the programmer's imagination, provided the program is "ready" at the end of the allocated development period.
It is exceptionally rare then that at this stage all the bugs are flushed out - especially in program segments dealing with complicated or unusual procedures.
Consequently a major portion of the programmer's time is devoted to program maintenance. As more of his time is spent on subsequent program maintenance over the course of the project the programmer is actually less productive.

Mistakes discovered earlier in the life cycle of a project are less costly to correct, therefore it is important that a programmer test his software systematically and thoroughly. This not only cuts down on the total development time, but also makes project planning and control more effective, and releases manpower for more productive tasks.

2. AUTOMATED TESTING TOOLS

The progress made in the field of software engineering research that aims to automate all the activities on a software project has not sufficiently embraced the field of software testing, although efforts are being made to automate this task, and some automated tools are available.

A couple of techniques recently being researched are outlined below:-

RANDOM DATA TESTING:

This process involves running a program and throwing large quantities of randomly generated test data at it (using a random number generator for example).

This method is cheap in terms of software support and another advantage is that you can predict the operational reliability of a program from its response to random data.

The arguments against this form of testing is whether random data can exercise a program thoroughly and if it is to be thoroughly tested, the large amount of data and the number of program executions needed. This also means that the human tester may have to examine physically the results of a large number of test cases.

Recent research and the advent of powerful computers has however managed to overide some of these arguments. Results have shown that 100% of statements and 93% of coverage of branches can be achieved using this technique.

ADAPTIVE TESTING:

This is probably the most likely way ahead into the automation of software testing.

The program to be tested is placed inside a test harness and the effectiveness of the data run through it is tested using another program called the "Optimiser". The optimiser examines the test database and predicts further data to increase the test effectiveness.

The predicted data is then run through the program ,its effectiveness measured, the database updated and so on until a specified level of test effectiveness is achieved.

Although adaptive testing is limited in its application area

it is highly regarded as a possible technique for more powerful testing systems.

Various other techniques are also being researched but the slow progress made in automating an activity so essential to a project enviroment should be a glaring indication of the immaturity of the software engineering field.

3. A DEFINITION OF 'TESTING'

Let us now look at some of the possible responses we might get if we asked for a definition of testing or a test case.

1. - Testing is the process of demonstrating that errors are not present.
   - To show that a program performs its intended functions correctly.
   - Testing is the process of establishing confidence that a program does what it is supposed to do.

2. - Testing is an exercise performed with the intention of finding the maximum number of errors.
   - To show that a program performs no function it was not designed to.
   - To confirm that a program gives only the results expected and no others.

In which of the above two groups would your answer lie?
If it lies in the first group I would suggest you rapidly rethink the philosophy behind testing.

If your aim is to demonstrate that a program has no errors then you will subconsciously steer towards this aim and tend to select test data that has a low probability of causing program errors.

In reality the aim of testing is the exact opposite. Since it is not feasible to produce a completely error free program, the aim should be to find as many errors as possible and therefore raise the reliability of the program.

Before testing one should start with the assumption that the program contains errors.

Some of the best testing definitions encountered were by a well known author on software testing.

> -TESTING IS THE PROCESS OF EXECUTING A PROGRAM WITH THE INTENTION OF FINDING ERRORS.
>
> -A GOOD TEST CASE IS ONE THAT HAS A HIGH POSSIBILITY OF DETECTING AN AS-YET UNDISCOVERED ERROR.
>
> -A SUCCESSFUL TEST CASE IS ONE THAT DETECTS AN AS YET UNDISCOVERED ERROR.

The same author also suggests that -

> -TESTING IS AN EXTREMELY CREATIVE AND INTELLECTUALLY CHALLENGING TASK!

When investigating the variety of testing techniques and their applications across the entire software development industry it proves surprisingly easy to agree with the above statement.

4. AREAS MOST PRONE TO ERRORS

The most common areas in which errors are prone to occur are:-

1. **CLERICAL ERRORS** - When the program has been miscoded or mispunched.
   e.g. A=B+10 instead of A=B+100

2. **LOGIC ERRORS** - The most common cause of software errors. May occur due to ambiguity or misunderstanding the original program specification for a function, or in the attempt to code a particularly complex function or procedure.

3. **DATA ERRORS** - A program may work perfectly if all data fed into the program is correct, however a robust program must have very stringent checks for the acceptance of all input data.

4. **EXCEPTION CONDITIONS**- In addition to performing the normal operations of processing data, programs must cope with exceptional and infrequent circumstances.

5. **PROGRAM EXTENSIONS** - When a program is extended to deal with more application areas or the intention is to optimise the programs, errors may be introduced when dabbling with existing code.

6. **PROGRAM CONVERSIONS** - Idiocyncrasis of individual compilers and the accommodation of non-standard features are just some of the conditions which are apt to hamper conversions and introduce errors.

7. **TIMING ERRORS** - These are particularly prevalent on real-time systems and are sometimes difficult to introduce for testing or to reproduce when found.

8. **LANGUAGE ERRORS** - These occur usually when a programmer is using a new language and may misuse or misunderstand the language facilities.

9. **OVERLOAD ERRORS** - Storage areas stressed upto and beyond their capabilities.

10. **THROUGHPUT AND CAPACITY ERRORS** - These involve computational resources such as CPU utilization, storage allocations etc.

11. **FALLBACK AND RECOVERY ERRORS** - The programs response to systems hardware and software failures.

12. **DOCUMENTATION ERRORS** - Technical and user documentation must reflect the programs function and use accurately.

13. **STANDARDS ERRORS** - The program must meet the overall project standards in terms of functionability, design, documentation etc.

5. SUGGESTED STANDARDS AND ACTIONS TO EASE TESTING

Most programmers will admit to using the "seat-of-the-pants" methodology for testing programs.

In the case of experienced programmers this is in itself a stratergy for testing, albeit unintentional. Their methods have been honed and finely tuned using years of experience based mainly on past foulups. The best of these programmers will readily admit that their methods work because of their experience not just in terms of time but also of the application they are working on. There are a few programmers who have the natural talent for a method of testing called "error-guessing". They are adept at identifying problem areas and more importantly at devising suitable test cases for those situations.

This is all well and good for the few lucky enough to have vast experience within a particular application , but regardless of what the arrogance of most programmers will have them believe this method cannot conceivably work for every member of a project team.

The economics of most project environments dictate the planning and scheduling of the project. The costs in terms of time and money on the essential exercise of testing can be minimised by some forward thinking at the project planning and control level.

5.1 At The Overall Project Level

Formulate the testing stratergy early in the life of the project. Ensure that this stratergy includes testing guidelines and standards at an overall level and at the level of individual programmers.

Bear in mind at this stage the impossibility of completely testing a program and then plan on who should test the programs, how much time should be spent on the testing exercise, how much can be done to ease the exercise by structuring at the overall project level etc.

When drawing up schedules allow sufficient time for testing. Remember that even minor modifications to a program need to be thoroughly tested.

Maintain simple but comprehensive documentation on tests undertaken when the program is written and also at each stage of modification.

Identify the need for a test harness and invest time in planning its layout, how it will be used and its upkeep during the life of the project.

Think of all possible ways of reducing testing time at individual program level.

Depending on the enviroment for example:-

- put as much information as possible on the dictionary
- identify copy code wherever possible
- write and pre-test sub routines which will be used throughout the system.

5.2 At Program Level

Some of the points individual programmer should bear in mind.

Avoid testing your own program. If you must then do not test assuming you will find no errors. Be masochistic, aim to find as many errors as you can.
Start formulating your test plan the moment you receive the specification (whether it is written formally, on the back of a cigarette packet or verbal).

Use your knowledge of the system where possible to identify relating programs and the effect of your program on them.

Ask questions at all stages and make notes for future reference.

Plan to test that the program does what it is supposed to AND what it is not supposed to.

Desk check your code all the time, not just when the machine is down.

Write test cases for valid/invalid, expected/unexpected conditions depending on the path of the input data.

To save time maximise the tests being performed on each test item of data.

A test case must have a predefined input and therefore a predefined output or result.

Thoroughly inspect results of each test.

Avoid throwaway test cases. Keep a record of any undertaken.

Remember! The probability of additional errors existing in a piece of code is directly proportional to the number of errors already found in that piece of code.

Warning! Sometimes correcting one error may introduce more errors in the program or uncover previously hidden errors.

Where the program terminates may not be in the neighborhood of the real source of the error.

If a test case fails for a test run repeat all test cases not just modified ones.

6. HUMAN TESTING

Usually undertaken after coding and before testing, but can in practical applications be used in higher levels ie design

6.1. Code Inspection/Walkthroughs

Both of these methods require a group of 3 to 5 people one of whom is the author of the program.

Each member is provided with a copy of the program listing and program/design specification. A "Chairman" who is not the programmer should control the session. The program is analyzed against a list of historically common programming errors. New errors found are added to this checklist. (see APPENDIX A).

Code inspection uses the technique of the programmer narrating the logic of the program statement by statement.

Questions are raised and errors discovered and recorded throughout this procedure.

The difference with walkthroughs is that the participants "PLAY COMPUTER". A series of paper test cases of input data and expected results are executed through the code. Although the test cases may be simple they provide a starting point for discussion.

In both methods the programmer is provided with a list of errors detected and provision is made to meet again after the errors have been corrected.

There are many advantages to code inspections and walkthroughs.

- Provides early detection of errors.
- Finds groups of relating errors and therefore the reason behind the errors, (although debugging is not the objective of the exercise).
- Since it brings together people of varied experience it benefits all members of the group in terms of being exposed to different programming techniques and styles.
- The net is also cast wider by finding errors not just in program code but also finding analysis and design errors. The overall position of the program in the system and its interaction with other programs is clear. These errors cannot be detected with machine testing techniques.
- The process will detect weaknesses in program structure with a view to maintenance of the program in the future.

- This technique is ideal for the error prone exercise of modifying programs. Most modifications to programs are hurriedly and sometimes thoughtlessly made. Inspections and walkthroughs remove the blinkers and look at the overall effect of a modification.

Surprisingly it is the programmers themselves who find most of their own errors. The process of reading their code out loud and explaining it to an audience is very effective in error detecting.

After being involved in a few of these sessions the programmer can learn to be objective and critical of his own code.

It can lead to programmers desk-checking portions of their code quite voluntarily. The standard of programming improves overall and less time is spent finding minor errors.

The disadvantages usually quoted for this technique are:-

- Code inspections and walk throughs can be time consuming and not always practical on projects with tight schedules. Most sessions can last 90 to 120 minutes.

- Sessions can disintegrate if people with strong egos start trying to score points against each other. Alternatively members of the group start playing the roles of "attack" and "defense". It is up to the Chairman of the group that the process proceeds with an egoless attitude and concentrates primarily on its purpose of error detection.

- External influences and interruptions.

6.2 Desk Checking

This technique can be looked on as a one person inspection/walk through.

It is not considered the best method of human testing but is preferable to none at all. It is relatively unproductive, even more so if programmers have to check their own programs.

If this form of human testing is to be applied improve the chances of error detection by letting programmers swap programs or test each others programs in pairs.

Whatever method of human testing you are involved in, ask yourself these questions:-

- Is the program easy to understand?

- Is the design obvious and reasonable?

- Is the program easily modifiable?
- Are you/would you be proud to have written this program?

7. TESTING METHODS

Testing cannot guarantee the absence of all errors in a program. This implies that "complete" testing is impossible and testing of any program may be incomplete.

Therefore given the constraints of time and cost it is necessary to identify the set of possible test cases which have the highest probability of detecting the most errors.

In practise a testing strategy must involve a combination of BLACK BOX (Data Driven, Input/Output Driven) and WHITE BOX (logic driven, path driven) methods for maximum testing effectiveness.

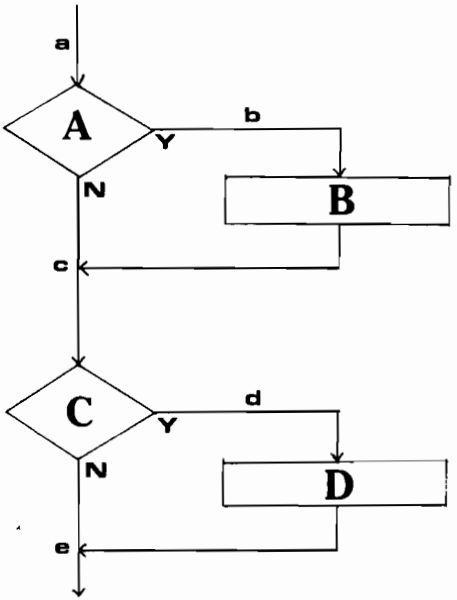
The methods used within each of the above are:-

BLACK BOX
Equivalence Partitioning
Boundary-value analysis
Cause-effect graphing
Error guessing

WHITE BOX
Statement coverage
Decision coverage
Condition coverage
Decision/condition coverage
Multiple-condition coverage

7.1 White Box Testing

Also known as LOGIC DRIVEN or PATH DRIVEN. What follows is a definition of each method available under the heading of "White Box" testing, its usage, advantages and disadvantages, using as an example the program code defined by the flow chart below.



7.1a Statement Coverage

Test cases should be written so that each statement is executed at least once.

In our example the statements could be shown to read

```
IF A THEN DO B
IF C THEN DO D
```

The test cases could be written to result in A & C being true and this would meet the definition. However paths 'c' and 'e' would not have been tested.

Though Statement coverage is basic to the definition of White Box testing, it is by itself a weak and insufficient method of testing since it is not required that the logic of the decisions be tested or that all false paths are taken.

7.1b Decision Coverage

Test cases must ensure that each decision has both true and false outcomes exercised and each entry point must be invoked at least once.

An entry point could be the result of

ON - Units
CASE statements
GO-TO-DEPENDING-ON etc

This method would cover the execution of each statement and decision but in the case of a complex decision it need not cover each condition within the decision.

e.g. If decision A consisted of the conditions

IF (X > 1 AND Y=0)

both true and false outcomes could be exercised with the test cases where Y EQ 0 and Y NE 0, but the condition for X would never be tested.

7.1c Condition Coverage

Test cases should ensure that each condition in a decision takes on all outcomes at least once and on each entry point is invoked at least once.

This definition may appear to cover all criteria but can with our simple example be insufficient in its literal definition.

e.g. For decision A above, condition coverage alone would be satisfied by using the test cases

i) X = 2 ; Y = 1
11) X = 1 ; Y = 0

but condition A is never true.

7.1d Decision/Condition Coverage

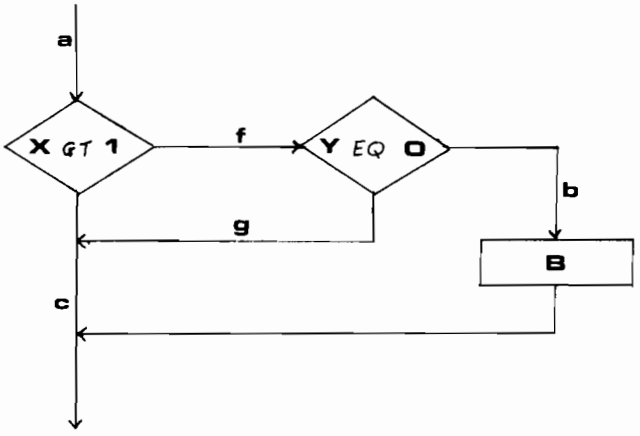
Test cases should ensure that each condition in a decision takes on all possible outcomes at least once, each decision takes on all possible outcomes at least once and each point of entry is invoked at least once.

Theoretically this is an ideal definition, but it falls over in cases where certain conditions mask other conditions.

e.g. For decision A above, the test cases of

i) X GT 1 ; Y EQ 0
ii) X LE 1 ; Y NE 0

could be said to meet all criteria.



However, if we expand decision A and view it as if at execution time, neither of the cases would test path g.

7.1e **Multiple Condition Coverage**

This definition should cover all the above problem areas. It requires that sufficient test cases be written such that all possible combinations of condition outcomes in each decision and all points of entry are invoked at least once.

For condition A then the test cases would be

i) X GT 1 ; Y EQ 0
ii) X LE 1 ; Y EQ 0
iii) X GT 1 ; Y NE 0
iv) X LE 1 ; Y NE 0

General Rules-of-Thumb

The number of test cases required to ensure all branches are tested is at least of the order 2^n for n branches.

For 1 condition per decision write sufficient test cases to

1) evoke all outcomes of each decision at least once
2) invoke each point of entry at least once.

For multiple conditions for a decision write sufficient test cases to

1) evoke all outcomes of each condition at least once
2) evoke all outcomes of each decision with every possible combination of conditions at least once.
3) invoke all points of entry at least once.

7.2 Black Box Testing

Also known as DATA DRIVEN or INPUT/OUTPUT DRIVEN. As with White Box testing there follows a brief discussion of Black Box testing methods and their usage.

7.2a Equivalence Partitioning

This method deals with testing input conditions for a program. It is clear that attempting to test input conditions of a program by throwing vast quantities of random data at it is impossible and impractical.

It is therefore necessary to select a subset of data with the highest probability of finding the most errors.

Divide the input necessary for testing into a definite number of classes (equivalence classes).

The test cases chosen should have two important properties

- each test case should invoke as many different input conditions as possible in order to minimise the number of test cases required.

- a test case using a representative value of an equivalence class would give the same result as any other value in that class.

for each input condition, tests for both 'valid' and 'invalid' equivalence classes must be selected.

The procedure for Equivalence Partitioning should be that for each input condition

- identify a 'valid' and 'invalid' equivalence classes and assign a unique number to each class.

- write representative test cases to cover all 'valid' and all 'invalid' equivalence classes.

e.g.

INPUT CONDITION	VALID EQUIVALENCE CLASSES	INVALID EQUIVALENCE CLASSES
Value must lie within range 1 - 9999	(1)(In range) 90	(2)(<1) 0 (3)(>9999) 1000
1 to 4 names may be listed on policy	(4)(1-4 names) J. Bloggs A.N. Other	(5)(No Names) (6)(>4 names) Happy Dopey Sneezy Grumpy Doc
Insurance type valid for Bus, Truck, Taxi, M/Cycle	(7) Bus (8) Truck (9) Taxi (10) M/Cycle	(11)(Not valid) Passenger
Policy number must begin with a letter	(12) A12345	(13) #12345 (14) 123456

The example layout above allows each input class with its uniquely numbered valid and invalid equivalence classes. A column to mark test results in each case should also be included.

7.2b **Boundary Value Analysis**

This method is similar to Equivalence partitioning but takes the concept further by investigating situations on, above and below edges of each equivalence class.
Therefore rather than selecting any element as representative of an equivalence class, more than one element is selected to test each edge of an equivalence class.

Boundary value analysis also attacks output equivalence classes by selecting test cases which would give results on the boundaries of expected results.

e.g. 1) If the intput must be in the range -1.0 to 1.0 inclusive, the test cases would be -1.0,1.0, -1.001 and 1.001.

2) If the input specifies a number of values i.e. 1 to 4 items, test cases should be written for no items, 1 item, 4 items and 5 items.

3) If input items result in an output to be in the range 1-5 inclusive, invent test cases to result in 1 and 5 and attempt to create output values of <1 and >5.

4) If input items result in an output of upto 4 items i.e. a code entered results in the retrieval of 0 to 4 records to be displayed, write test cases for the retrieval of 0, 1 and 4 records and also attempt a test case which will retrieve 5 records

5) For input and output conditions on ordered sets (sequential files, tables etc) focus attention on the first and last elements.

7.2c **Cause-Effect Graphing**

This method explores combinations of input circumstances and devises a systematic approach to generating test cases for these circumstances.

Without a systematic approach an arbitrary set of test conditions is usually chosen leading to inefficient and incomplete testing.

What follows is a brief description of cause-effect graphing using a simple example which is in no way representative of the complexities of the function. Should the reader require detailed explanations of cause-effect graphing it is suggested that he read one of the excellent reference books available.

Cause-effect graphing is a technique for selecting high-yield

test cases for combinations of input circumstances with an added benefit of highlighting incompleteness and ambiguities in the specification.

The technique used transfers the specification into a 'formal'model or language using BOOLEAN logic onto a cause effect graph.

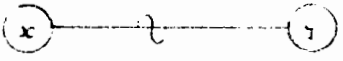
To derive test cases the following process is recommended

- divide the specification into small manageable pieces representing each function.
- identify each cause (input condition of equivalence) and effect (output condition or program function) and class assign each a unique number.
- analyse the content of the specification and using the causes and effects construct a cause-effect graph using BOOLEAN logic.
- annotate the graph with constraints that describe combinations of causes/effects that are impossible to generate.
- produce a 'limited-entry' decision table with each column representing a test case by tracing the state conditions in the graph.
- convert the columns in the decision table into test cases.

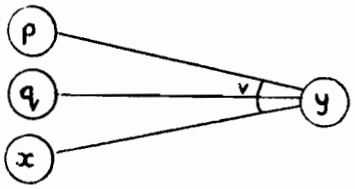
A variable's 'State' may by 0 (absent/false) or 1 (present/true)

BASIC GRAPH SYMBOLS

Identity symbol -
If x = 1 then y = 1
else y = 0



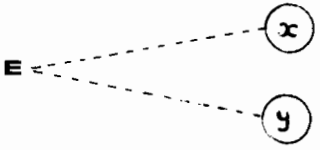
NOT symbol -
If x = 1 then y = 0
else y = 1



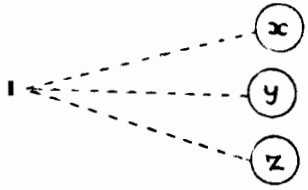
OR symbol -
If p = 1 or q = 1 or x = 1
then y = 1
else y = 0

AND symbol -
If q = 1 and x = 1
then y = 1
else y = 0

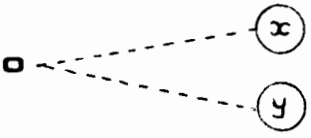
CONSTRAINT SYMBOLS



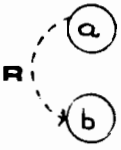
EXCLUSIVE symbol - at most one of x and y can be 1 (x and y cannot be 1 simultaneously)



INCLUSIVE SYMBOL - at least one of x, y & z must be 1 (x, y & z cannot be 0 simultaeously)



ONE AND ONLY ONE SYMBOL - one and only one of x and y must be 1.



REQUIRES symbol - for a to be 1, b must be 1.

"MASKS" SYMBOLS

MASK symbol - if a = 1; b is forced to 0.

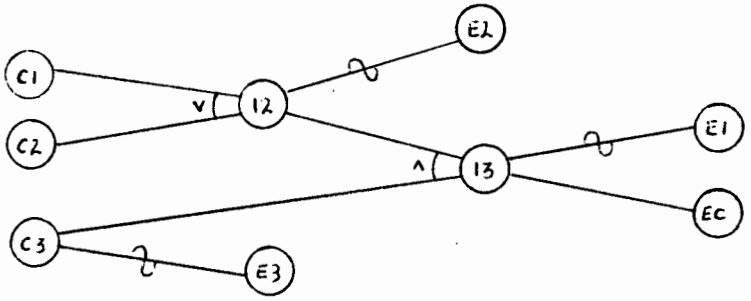
Example

Let us assume we have identified a function of a program which requires a supplier code to be entered and the record relating to that code displayed on screen, otherwise a relevant error message is displayed. The first two characters of the code must represent a valid Branch code or Country code and the next two characters represent a valid year from 80 - 87. (There is no intersection between the sets of Branch and Country codes).

Causes : C1 - first two characters represent Branch code
C2 - first two characters represent Country code
C3 - second two characters represent year.

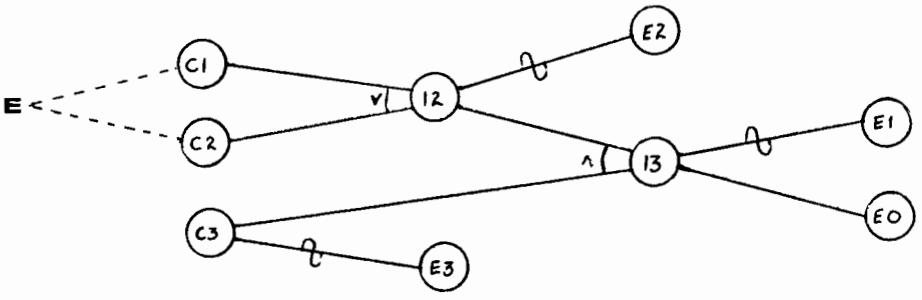
Effects: E0 - record exists and is retrieved and displayed.
E1 - record does not exist and error X1 is reported
E2 - first two characters incorrect and error X2 is reported
E3 - second two characters incorrect and error X3 is reported

CAUSE-EFFECT GRAPH



CAUSE-EFFECT GRAPH WITH CONSTRAINTS

The first two characters cannot represent Branch and Country code at the same time.



LIMITED-ENTRY DECISION TABLE

	1	2	3	4	5	6	7
C1	1	0	1	0	0	1	0
C2	0	1	0	1	0	0	1
C3	1	1	1	1		0	0
12	1	1	1	1	0	1	1
13	1	1	0	0		0	0
E0	1	1	0	0	0	0	0
E1	0	0	1	1	0	0	0
E2	0	0	0	0	1	0	0
E3	0	0	0	0	0	1	1

It is obvious from the diagrams that nodes 12 & 13 represent intermediate states.
Using the seven test cases derived on the limited-entry dicision tables appropriate test cases can be constructed.

(N.B. 1 represents 'present' or true
0 represents 'absent' or false
blank represents don't-care situations)

8. MODULE TESTING

The concept of splitting a program into a number of self-contained units is well established in data processing. The success of modular programming depends on the way the program is divided, the average size of a module , the interfaces relating the modules and how the modules are tested.

Some of the advantages of module testing are:-

- ease of readability of the program
- logical units of the program are obvious and easy to isolate
- the inputs and outputs to logical units are clear
- when modifications to a program are necessary, the relevant areas in the program are easily identifiable.
- testing (and debugging) modules is easier than for a monolithic program.
- module testing can introduce parallels by allowing multiple modules to be separated simultaneously.

When testing individual program modules apply the relevant white-box techniques to the logic of the program and then supplement these tests with applicable black-box testing methods.

Module testing can be performed using 'incremental' or 'non-incremental' methods.

8.1 Non-Incremental Module Testing_

Using this approach each module in a program is tested as a stand-alone entity.

The modules are then combined or integrated to give the final program, which should then be tested for module interfaces.

When testing individual modules, a 'driver' module and one or more 'stub' or 'receiving' modules are required.

A 'driver' module is a small module which has to be available or written to input the test cases through the module being tested.

One or more 'receiver' or 'stub' module have to be written to accept each type of output from the tested module and return control to the tested module.

The output may be a call to another program, files being updated, output to a printer etc.

It is not always enough for a stub module to be hardwired with a result. If a stub simply returns control to the calling module or writes an error message, the module being tested may fail since the stub module does not give the correct response.

It is therefore important that stub modules are carefully written.

8.2 Incremental Module Testing

The difference between Incremental and Non-incremental module testing is that incremental module testing does not test each module in isolation.

It follows the reasoning that the next module to be tested is first combined with the set of modules which have already been tested.

The two methods available are commonly known as 'TOP-DOWN' and 'BOTTOM-UP' testing. These terms should be familiar to the readers.

8.2a Top-Down Testing

This method requires that the initial module of the program be tested first.

The next module for testing must have its calling modules already tested.

To select the next module for testing the guidelines to be considered are:-

- If there are critical sections of the program, design the sequence of testing such that these sections are added as early as possible.

- Design the sequence such that the I/O modules are added as early as possible.

Carefully written stub modules are necessary to represent each module being called by the tested module. These stubs are then replaced by the module they represent when it is its turn to be tested.

If the I/O modules are added as early as possible, the input of test data and the inspection of results is greatly simplified.

8.2b Bottom-up Testing

Bottom-up Testing may be seen to be the opposite of Top-down testing. The advantages of one are the disadvantages of the other and vice-versa.

Using this method the terminal modules in a program i.e. those that do not call other modules are tested first.

Each terminal module needs a driver with wired in test inputs which calls the module being tested. It may also display the output or compare the output with expected results.

The next module to be tested must have all of its subordinate modules previously tested.

The guidelines for selecting modules are the same as for top-down testing.

Multiple versions of driver modules may not be necessary since a driver can iteratively call the module being tested. Sometimes a test tool if available, can be used. Drivers are usually easier to produce than stub modules.

There is no clear cut answer to the issue of whether top-down or bottom-up is the best method to use. Both have their advantages and disadvantages.

The best idea is to weigh up the factors outlined in APPENDIX B and relate them to the program being tested.

8.3 Incremental Vs Non-Incremental Testing

From the discussion below it will be observed that incremental testing is more efficient than non-incremental testing in most situations.

However it is never a good idea to discount any testing method if it suits a situation.

- Non-Incremental testing requires more work. Each module needs a driver and one or more receivers. Bottom-up testing will require an equivalent amount of drivers but no receivers while top-down testing will require equivalent receivers but no drivers (except perhaps for the initial module).

- Interface errors between modules will be detected earlier using the incremental method. With non-incremental testing the modules do not come together till the end.

- Debugging is easier using the incremental method especially where interface errors are concerned since the error is likely to be present in the most recently added module.

- Incremental testing usually results in a more thoroughly tested program. This is because each module in the chain is tested every time a new module is added and therefore receives more exposure to test data.

- Although non-incremental testing appears to use less machine time, more drivers and stubs are required which need more machine time and interface testing is also time consuming when the program is brought together.

- At the beginning of module testing, it is easier for parallel testing to take place using non-incremental testing. This can be quite significant on a large project.

Appendix A

DATA REFERENCE

1. Unset variables used?
2. Subscripts within bounds?
3. Noninteger subscripts?
4. Dangling references?
5. Correct attributes when aliasing?
6. Record and structure attributes match?
7. Computing addresses of bit strings?
8. Based storage attributes correct?
9. Structure definitions match across procedures?
10. String limits exceeded?
11. Off-by-one errors in indexing or subscripting operations?

COMPUTATION

1. Computations on nonarithmetic variables?
2. Mixed-mode computations?
3. Computations on variables of different lengths?
4. Target size less than size of assigned value?
5. Intermediate result overflow or underflow?
6. Division by zero?
7. Base-2 inaccuracies?
8. Variable's value outside of meaningful range?
9. Operator precedence understood?
10. Integer divisions correct?

DATA DECLARATION

1. All variables declared?
2. Default attributes understood?
3. Arrays and strings initialized properly?
4. Correct lengths, types, and storage classes assigned?
5. Initialization consistent with storage class?
6. Any variables with similar names?

CONTROL FLOW

1. Multiway branches exceeded?
2. Will each loop terminate?
3. Will program terminate?
4. Any loop bypasses because of entry conditions?
5. Are possible loop fallthroughs correct?
6. Off-by-one iteration errors?
7. DO/END statements match?
8. Any nonexhaustive decisions?

INPUT/OUTPUT

1. File attributes correct?
2. OPEN statements correct?
3. Format specification matches I/O statement?
4. Buffer size matches record size?
5. Files opened before use?
6. End-of-file conditions handled?
7. I/O errors handled?
8. Any textual errors in output information.

INTERFACES

1. Number of input parameters equal to number of arguments?
2. Parameter and argument attributes match?
3. Parameter and argument attributes units system match?
4. Number of arguments transmitted to called modules equal to number of parameters?
5. Attributes of arguments transmitted to called modules equal to attributes of parameters?
6. Units system of arguments transmitted to called modules equal to units system of parameters?
7. Number, attributes, and order of arguments to built-in functions correct?
8. Any references to parameters not associated with current point of entry?
9. Input-only arguments altered?
10. Global variable definitions consistent across modules?
11. Constants passed as arguments?

OTHER CHECKS

1. Any referenced variables in cross-reference listing?
2. Attribute list what was expected?
3. Any warning or informational messages?
4. Input checked for validity?
5. Missing function?

Appendix B

Top-Down Testing

ADVANTAGES

1. Advantageous if major flaws occur toward the top of the program.
2. Once the I/O functions are added, representation of test cases is easier.
3. Early skeletal program allows demonstrations and boosts morale.

DISADVANTAGES

1. Stub modules must be produced.
2. Stub modules are often more complicated than they appear to be.
3. Before the I/O functions are added , the representation of stubs can be difficult.
4. Test conditions may be impossible, or very difficult to create.
5. Observation of test output is more difficult.
6. Allows one to think that design and testing can be overlapped.
7. Induces one to infer completion of the testing of certain modules.

Bottom-Up Testing

ADVANTAGES

1. Advantageous if major flaws occur toward the bottom of the program.
2. Test conditions are easier to create.
3. Observation of test results is easier.

DISADVANTAGES

1. Driver modules must be produced.
2. The program as an entity does not exist until the last module is added.

Bibliography

Myers, Glenford J., — The Art of Software Testing John Wiley & Sons.

Deutsch, Michael S. — Software Verification and Validation Realistic Project Approaches Prentice-Hall Series in Software Engineering.

Wong, Kenneth K., — Computerguide 5:Concepts in Program Testing. The National Computing Centre Ltd.

Perry William E., — A Structured Approach To System Testing. Prentice-Hall,Inc., Englewood Cliffs, New Jersey.

Ince Darrel, — Its a Testing Time For Harassed Programmers. Computer Weekly September 12, 1985. -Senior Lecturer. Open University.

Up-front Planning - The Key to Success With Resource Sharing

Len Croley
Hewlett-Packard Co.
Office Systems Division
8010 Foothills Blvd.
Roseville, California 95678

Resource Sharing implements the shared disc concepts for sharing data files directly from PC applications (previously implemented only on PC servers) using the HP 3000 system discs. The product also allows the user to access printers and plotters connected to the HP 3000 directly from PC applications and features a PC Tape Backup facility which allows the user to backup files from a local hard disc to an HP 3000 system tape drive. Resource Sharing works over a ThinLAN, StarLAN, or SERIAL Network (modem connection suggested) to the HP 3000.

What's that you say? Up-front planning is the key to success with anything from cleaning windows to building a house to managing a department or even a company. So, what is different with Resource Sharing?

One of the main differences between Resource Sharing and many other products on the HP 3000 is that users do not have to acquire a lot of PC technical knowledge let alone knowledge of the HP 3000 system and data communications. Without careful planning and correctly setting the expectations of your PC users, they may think that they need HP 3000 system and data communications knowledge to do the same work they had previously been doing with their PCs. The reality is, using the capabilities of Resource Sharing can be almost entirely invisible to the PC user. The PC user's perception should be that you have magically connected all of the printers, plotters and additional disc space to their PC. They need not concern themselves if others are using the same physical resources. This can all be accomplished through careful up-front planning with the end result of not only happy, but enthusiastic users with increased productivity and quality of work.

To successfully implement Resource Sharing, the initial set-up of both the HP 3000 and PCs should be well thought out and planned ahead. This document will discuss the following items related to planning for and implementing the Resource Sharing product.

1. The planning process - When to do it, and what is involved in planning the environment for sharing data and printers and for loading PC applications from shared discs.

2. Performance - Things to do and not to do which directly affect the performance of the HP 3000 and PC application.

3. Data Security - What is available and how is it implemented.

4. Automation - Setting up automatic access to the shared resources on the HP 3000 for both novice users and experienced HP 3000 users.

The Planning Process

The planning process should begin when the needs are being analyzed to see if some kind of server for file, printer or plotter sharing is an answer to your problems. Then, when a decision has been made that implementing a server is the correct thing to do, much more planning should take place to implement the correct training for both the PC users and whom ever is going to manage the server. It also take a lot of planning for implementation of the server according to the requirements of the end users.

Who needs a server?

First, lets take a look at what needs to be considered when planning on getting a server. Besides costs, but what productivity and quality gains need to be considered. As a result, the choices could be any of the following:

1. A separate printer and/or plotter attached directly to each PC.

2. Multiple peripherals shared by users using data switches.

3. A PC server for sharing files, printer(s) and plotter(s).

4. An HP 3000 host system server.

To analyze your needs, the following worksheet might be helpful. After completing the worksheet, the higher the total number, the higher the number of the above options it is that you need.

Probability Of Needing A Server

Condition	# Points	Your Points Today/In 1 Year
1. Number of Users within the group being considered.		
A. 1 - 3 Users	1	
B. 4 - 10 Users	2	
C. 11 - 15 Users	3	
D. 16 - 25 Users	4	
E. > 25 Users	5	
		_____/_____
2. Amount of printing per day.		
A. 1 - 10 Documents	1 - 2	
B. 11 - 20 Documents	3 - 4	
C. > 20 Documents	5	
		_____/_____
3. Amount of plotting per day.		
A. 1 - 5 Slides	1	
B. 6 - 10 Slides	2	
C. 11 - 15 Slides	3	
D. 16 - 25 Slides	4	
E. > 25 Slides	5	
		_____/_____
4. PC Specifications		
A. Most or all PC users have a hard disc and floppy disc.	1 - 2	
B. Spreadsheet users have hard discs, word processor users do not.	3	
C. Most or all PC users have Floppy discs only.	4 - 5	
		_____/_____
5. Number of PC applications primarily used per day.		
A. 1	1	
B. 2	2	
C. 3	3 - 4	
D. > 3	5	
		_____/_____

6. File sharing requirements.

A. Infrequently.	1
B. Within small select groups (2-3 secretaries, etc.)	2
C. Frequently within a work-group.	3 - 4
D. Data base access or across a complete functional group (MIS or Mkting Dept.)	5

_____/_____

7. HP 3000 Application Usage.

A. Infrequently or None.	1
B. Electronic Mail	2 - 3
C. Frequent (Data base access, Mfg applic., etc.)	4 - 5

_____/_____

YOUR TOTAL _____/_____

If your total is between 7 and 12, especially with 1-4 users, you might well want to consider a separate printer and/or plotter attached directly to each PC. Before doing this, however, you should carefully consider both your present and future needs.

If your total is between 13 and 20, you will probably want to consider either option 2, data switches, or option 3, a PC server, based more on costs. Remember, however, that the cost of data switches or print spooler black boxes can quickly equal the cost of a PC server and will not provide file sharing.

If your total is over 20, then an HP 3000 server is your best solution. The HP 3000 server overcomes the limitations of a PC server. The HP 3000 server can both print and plot at the same time and accept files to be printed or plotted from a remote PC over a modem. It should be emphasized that going to an HP 3000 server involves a big step in cost and possibly training (of the system manager and possibly the users), however the work of implementing the HP 3000 server is not much more than implementing a PC server.

While the general values given above are close, there are other factors that should be considered as will be individually discussed below. The prime consideration is that if a point value of **5** was chosen as the answer for any of the questions, then an HP 3000 server is most probably required.

Plotting: Plotting is mostly a case of how much. If plotting is only an occasional thing, smaller plotters attached directly to the PC are probably a good solution. If you are consistently plotting several slides per day then consider an HP7550A, which comes with an automatic sheet feeder, either attached to one PC and move the files to that one or attached to a server so that the plotter can be spooled, thus freeing up the user's PC for more work while the plotting is being done.

Printing: For printing, both quality and quantity should be considered. In some cases, the best solution is to attach an HP ThinkJet or HP QuietJet printer to most PCs for preliminary printing of documents and attach an HP LaserJet+ to a server for printing the final version of the documents. If the printer is attached to an HP 3000 server, you also need to consider the type of printing the user will be doing to decide whether to configure the user to use Processed mode, Direct mode, or some of both.

The philosophy of Processed mode printing is that the PC application printer is configured as an HP2601 or Diablo 630 and the HP 3000 will translate the data to print using the font (typeface), media, and language for the printer specified when the connection to the printer was made. The user does not have to be concerned about any particulars of the printer. When using this mode of printing, the user can also specify the number of copies for the HP 3000 server to print whereas most PC applications printing directly to a printer requires the user to do a separate print operation for each copy. Another consideration for when using Processed mode printing is that many of the unique printer features cannot be used such as mixed text and graphics and, with some applications, bold or underline.

Direct mode of printing can only be used for printers which can also be supported if attached directly to a PC. This means that in order to use a printer attached to an HP 3000 in Direct mode, the PC application must have a printer driver that will support that printer. The name Direct mode comes from the fact that the HP 3000 does not modify the printer data in any way, but just passes it <u>directly</u> to the MPE spooler. This means that the printer must be supported as an MPE spooled printer and that escape sequences can be passed to the printer thus allowing mixed text and graphics and printer control escape sequences.

Some printers such as the HP LaserJet series can be supported as either Direct or Processed mode printers. In this case. one user can be using the printer in one mode and another user can use the printer in the other mode or, a user can be using the same printer in one mode with one PC application and then in the other mode with another PC application. What mode is used depends entirely upon information specified when the connection to the printer is made. Your HP Sales Representative has the latest information about what printers are supported and in what mode. These factors needs consideration, however when deciding what kind of printer to purchase and how to set up the user.

Printing and Plotting to a PC Server: There are some limitations when trying to print and plot to a PC server. One of these limitations is that the HP PC Server solution in most situations can either print or plot, but not both at the same time. Therefore, if someone has sent a lot of files to the PC server to be plotted, the printing files might not be printed for a long time without someone managing the queues within the PC Server. The files are printed or plotted on a FIFO (first in; first out) basis. Also, the PC server cannot accept files to be printed or plotted from a modem, only from an HP ThinLAN or StarLAN connection to the PC server.

There is also a connectivity issue with a PC server in that as of this writing, only 10 users can be connected to the PC server at one time. The connection limit may be increased, so check with your HP Sales Representative if this is an issue for you.

Number of Users: The more users you have or will be adding during the next year will mean the more printing, plotting and file sharing that will be required. Therefore, by the time you have greater than 20 users, the requirements for these types of capabilities should justify an HP 3000 server.

Amount of Printing per Day: The more documents that are printed per day means the more requirements there will be for quality printing and for mixing text and graphics. As the number of documents printed per day increase, you need to look carefully at the type of printers and their duty cycle in addition to the printer's capabilities. For example, some printers cannot do graphics, some printers cannot do enhance printing such as underline, bold, and different fonts, and some printers do not have sufficient duty cycle (are not designed to print very many pages per day). If there are more than 3 users and LaserJet printing quality is required, at least a PC server could be well worth the money.

Amount of Plotting per Day: The more slides that are required to be plotted per day is also an indication that there are more users trying to access the plotter. Since slides take a lot of time to plot, by the time the requirement of 6 - 10 slides per day is reached, you will probably find that it almost takes a dedicated PC for the plotter. Of course, there is the choice of purchasing more cheap plotters and attaching them to the individual PCs, especially if 2 or 3 users are doing most of the plotting, but you will probably find that as more users get familiar with PC applications such as Graphics Gallery, the quicker the plotting requirements increase. Once you have spent the money for a high quality high durability plotter which automatically feeds the paper or transparencies, such as the HP7550, with a PC to drive it you are well on your way to having a PC server. Also, if you are doing a lot of plotting, you might want two HP7550 plotters; one for paper and one for transparencies.

PC Specifications & Number of PC Applications: Considering these two factors together is a good indication of the amount of work being done by the users just to switch floppy diskettes. Eliminating this factor when several PC applications are being used each day could by itself give a surprising amount of productivity improvement. Also, there is the factor of floppies getting damage and causing the loss of data as well as the factor of archiving information. With PC servers, one of the best ways to archive information or backup files for disaster recovery is to use an HP9144 Cartridge tape drive (HP-IB interface required). With an HP 3000 server, shared disc files are backed up automatically during the normal system backup procedures. The files can then be recovered using the PC file names by using a utility called RESMGR that is provided with the Resource Sharing product. Also, with the Resource Sharing product, hard discs attached to the PC can be backed up on a regular basis for disaster recovery by the user activating a utility program on the PC before going home at night. The first time the disc is backed up, all files will be backed up. Thereafter, each night the utility is run, only files that have been created new or modified since the last backup will be backed up.

File Sharing Requirements: Sharing files and data infrequently or within small groups such as secretaries is many times just as convenient to accomplish by passing the information on floppy diskettes because the person needing the information does not have to worry about "where the data is". Generally only a few files will be on the floppy. When a server is implemented, the person trying to get a copy of a file usually has more concerns about where the file is at, especially if the users are using good file management techniques, i.e., using subdirectories and extent names to more readily define what type of data is in a given file.

However, with more users and the need to have access to data already in the form of a PC file being a requirement, servers can significantly improve productivity. This can be seen with a couple of examples. One example is when say 12 people need a copy of a file, maybe multiple times such as you would find with a standard letter head or memo head. Another example is when one person needs to assemble a report consisting of smaller reports from several people. If the people submitting the individual reports all save the files on a common shared disc in a common subdirectory, the person assembling the combined report can immediately have access to all the files needed.

HP 3000 Application Usage: You might not have answered this question because you did not already have an HP 3000. But try answering the question on the basis that you need the HP 3000 as a server because of your needs as determined by the other questions. The capabilities of the HP 3000 and the applications available for the HP 3000 could give you that extra plus you need to really take advantage of the office productivity gains available by using the full capabilities of the PC applications with the HP 3000 used as a server.

Conclusion:

If after all of your investigation you are still unsure whether to implement a PC server or HP 3000 server, a valid alternative is to implement a PC server first and then move to an HP 3000 server. As the number of users increases, the needs of the users increase, or HP 3000 applications are needed to solve other problems, the increase in cost will be lower by moving the PC functions to the HP 3000 server. Remember, that the PC server can be converted to another user PC simply by reworking some of the software and the peripherals that were attached to the PC server can be directly ported to the HP 3000 server and used. Even the HP9144 Cartridge Tape drive that is used to back up the PC server can be moved to the smaller HP 3000 systems to be used as a dedicated tape drive for the PC Tape Backup feature of Resource Sharing.

This migration is extremely easy because the software on the PC used to implement access to a PC server is the same for accessing an HP 3000 server. The only software that might be added to the user's PC is the PC Tape Backup utility. The data communications software is the same.

What training might be needed?

You need to consider and plan for two types of training; (1) training for someone to accomplish management of the server and network and be the main person for users to go to for help, and (2) training for the PC users.

Before what training is needed can be adequately determined, you need to understand the technical expertise level of the PC users. What ever the level of the PC users, however, it is highly recommended that a person be designated as the "expert" (in some organizations, an "expert" is someone to take all or part of the blame, but this should not be the case here) and be dedicated to implementing the server and helping the users for a period of time. The amount of time will depend upon the server being used, how many users there are and the ability of the users to use and understand DOS level commands. It would probably be a good idea to have this person be the instructor for the end users.

Training for the Expert: To adequately set everyone's expectations (both management's and co-workers) to be able to get the person the time to learn, prepare for any classes that might be taught to end users and to be bothered by the users when they have questions, the "expert" should be given a duty title (possibly "job" title in large organizations with a great many users) of something like "Network Manager" or "Office Products Coordinator".

To take a look at the training that might be needed for this person, lets assume we have the following situation: the server is an HP Micro 3000 system with LAN (Local Area Network) connections to the PCs (either ThinLAN or StarLAN - the only difference besides the physical hardware connections is the data communications software is on the PCs). The PCs are being used for word processing, spread sheets using data retrieved from IMAGE data bases on the HP 3000 using HP Information Access and electronic mail (HPMAIL) for communications to people throughout the company. There are 25 users consisting of secretaries, managers, marketing people who write sales literature, support sales representatives and technical people who write product documentation and develop product training, and support service representatives. One of the technical people is chosen as the "expert" to manage and implement the HP 3000 system and PCs so that the end users do not have to learn anything beyond how to use the PC applications.

The best philosophy to get this person trained is to spend the money to send the person to courses whenever possible even if the person has the ability to learn on their own because this gives the person a dedicated time to learn what is needed in an organized manner. Also, if you do not actually have the system yet, this gives the person a chance to actually try the things learned by using a system.

To manage the HP Micro 3000, the person must take a "HP System Administrator" course. This course teaches how to manage the HP 3000 system and gets the student started using the HP 3000 JCL to automate many items such as nightly back-ups of the HP 3000 system and user's PCs. HP requires someone at the site to have completed this course in order to purchase a support contract on the HP 3000 system.

To be able to set up, manage, and troubleshoot the network, the person should take the "Managing the OfficeShare Network" course. While this course primarily covers PCs connected to a PC server, the fact that the OfficeShare data communications product is required in order to use Resource Sharing means that much of the information required in the case we are studying is directly applicable. Because we have specified that the users would be using HPMAIL for communications to others within the company, the implication is that an understanding of intersystem communications is necessary and therefore, this person should take the Managing NS/3000 Data Communications course. If you do not have HPMAIL or if you are only going to only have this capability within the users on the single HP 3000 system, you will not need this course. The Resource Sharing product only requires the HP ThinLAN/3000 Link or HP StarLAN/3000 Link product and not the full NS/3000 product capabilities These products are to be initially set up by an HP SE. Once set up, these HP 3000 data communications link products should not require a great deal of detailed knowledge to maintain and assistance can be purchased from HP or the course can be taken at a later time if it is determined to be required.

Assistance for Resource Sharing is available as part of the data communications HPASSIST package. Specific training for Resource Sharing can also be obtained as a customer classroom course (contact your HP Sales Representative) and in a self-paced training manual called "Resource Sharing HP FastTrak" orderable through the HP Computer User's catalog.

There is also customer data base and HPMAIL training and HP ASSIST available for customers purchasing the HP Information Access or HPMAIL products.

If this person does not have a good background on DOS commands such as MKDIR, RMDIR, and CD, there are commercial courses available.

For PC application knowledge, HP teaches courses on HP proprietary PC software and many of the applications have self paced training available. This person especially needs time to learn all of the applications that need to be supported at least to the degree that they can configure the applications so that they print and/or plot on the devices connected to the network.

Training for the users: If the operations required to access the peripherals on the server is fully automated, very little training should be required for the users beyond training for new PC applications such as AdvanceMail which might be implemented at the same time. It is recommended, however, that a "training" session be put on for the users to properly set their expectations and fully explain what is happening. Actual usage training for fully automated users would probably only consist of what P.A.M. labels or commands (which actually call .BAT files to do the work) to switch between a local printer and a printer attached to the server or to switch between different printers on the server.

If the PC users are also going to start running HP 3000 applications, they might need some training on using the HP 3000 and some training on using AdvanceLink (the only way to run HP 3000 applications from an HP Vectra or IBM PC).

If the users are going to use HPMAIL directly, this is an HP 3000 application and therefore they may need the training mentioned above as well as HPMAIL training. If HPMAIL is going to be accessed by using AdvanceMail, the users should only need AdvanceMail training.

If the users are going to use HP Information Access, they may need training on this product. What they should not need is training on the HP IMAGE or Query products.

Planning For The Users:

Notice that in the previous paragraph, I did not propose a lot of training for the PC users, but to make this happen, the person setting everything up for the users must do a lot of up-front planning. The first thing that must be accomplished is to do a complete study of each user and generate a "User Profile" on each individual user. The end result for any particular user could be anywhere between "Give me the software and any configuration values I need and I will do it" to a fully automated PC where the user does not need to know anything except which P.A.M. labels or .BAT command files to use.

As a result of having a good user profile, the "expert" (whom I will hence forth call the Network Manager) will not only make the users happy (which translates into "successful"), but will in turn also be successful. In the case of the very technical user, you may not want to overdo the automation. (I know some technical people who think that anyone who automates things and uses P.A.M. labels must not be very technical.) In the case of the fully automated user, you can have the work done ahead and should not under automate them which will cause them to have to learn DOS and/or MPE commands.

Because you are using a network even to a PC server, however, there is some coordination that needs to be accomplished such as IP (Internet Protocol) addresses and node names.

The user profile: Following is a sample user profile form. This form can either be given to the individual users or filled out by the Network Manager. I would recommend the latter because the questionnaire by itself may not bring out other factors that would possibly come out while doing an interview to fill out the form. For example, a user might not think that the fact that they can use some DOS commands vrs. the fact that they must do so if the PC is not fully automated may turn out to be too much of an imposition when they really begin to use the new capabilities. This could result in a frustrated user.

This form makes some assumptions. If these assumptions are not correct for your situation, you might want to add or reword questions. The assumptions are that the server being installed is not an upgrade or replacement for a PC server, the PC user has not used a print redirector such as HP Print Central, and the user does not have experience using HP Information Access.

User Profile

Name _____

Position or Job Title _____

1. What level of experience using the HP 3000 do you have? (Check one.)

   _____ Experienced HP 3000 user.

   _____ Can log onto the HP 3000 and run programs.

   _____ No experience with the HP 3000.

2. What level of experience using PCs do you have? (Check one.)

   _____ Experienced PC user; can use DOS commands well and can create .BAT files.

   _____ Can use some DOS commands such as DIR, MKDIR and CD.

   _____ Can run some PC applications, but do not know any DOS commands.

   _____ Cannot use a PC.

3. What types of PCs have you used? (Check all that are appropriate.)

_____ HP 150

_____ HP Vectra

_____ IBM PC, PC/XT, or PC/AT

_____ Other (Specify.) _____

4. What type of PC do you expect to be using the most and does it have a hard disc?

_____ HP 150 Hard Disc? Y / N

_____ HP Vectra Hard Disc? Y / N

_____ IBM PC, PC/XT, or PC/AT Hard Disc? Y / N

5. Have you ever called into an HP 3000 system using a modem?

_____ Yes

_____ No

6. How many PC applications do you use per day?

_____ PC applications per day.

7. Do you have a printer or plotter attached to your PC?

_____ No

_____ Printer and Plotter

_____ Printer only

_____ Plotter only

8. How many documents on an average day do you print?

_____ Average # documents.

_____ Average # copies.

9. List all of the printers to which you need to print. (List in the order of most frequently used first.)

Printer Location	**Type of printing (Direct/Processed)**
_____	_____
_____	_____
_____	_____

10. How many drawings on an average day do you plot?

_____ Average # drawings.

11. List all of the plotters to which you need access. (List in the order of most frequently used first.)

Plotter Location

_____

_____

_____

12. List others with whom you expect to be sharing files.

Person to share files with.	How often?
_____	_____
_____	_____
_____	_____
_____	_____

What to do with the information from the questionnaire:

The questionnaire is designed to give you a lot of information with just a few questions. Following is what the answers to the questions mean in terms of being able to prepare the users ahead of actually implementing a server.

1. What level of experience using the HP 3000 do you have? This question will give you information on what training might be needed according to other parameters such as would they be needing a copy of the Resource Sharing Utilities manual so they can convert files to/from MPE file format and shared disc file format and possibly be setting up shared discs and assigning shortnames to these shared discs.

It could also be an indication of the types of P.A.M. labels or .BAT files that might need be created such as those required to automatically log onto the HP 3000 when they go into AdvanceLink.

A person that is an experienced HP 3000 user and PC user (question 2) is a good candidate to become the Network Manager discussed above. (Don't forget to find a backup person that may learn more on the job than by having all of the formal training of the main Network Manager.)

2. What level of experience using PCs do you have? This question is meant to get a good feel of the level of automation required for the user. The less experience the user has, the greater level of automation that probably will be required. The process of automation will be more fully discussed later in this paper.

If either of the first two answers are checked, the person is generally more technical and this may also be an indication that the person might need less training in order to be able to run HP 3000 applications if they haven't done so before and will now be required to do so.

3. What types of PCs have you used? This question will greatly aid in determining what training the user needs and what level that training needs to be. If the "Other" line is checked and the PC listed does not use DOS, the user could need a lot of training. If the user has only used HP PCs and expects to be using an IBM, having to interact with DOS and issue commands instead of selecting P.A.M. labels may be a big transition and may require training.

4. What type of PC do you expect to be using the most? Sometimes the user may be currently using the PC listed in question 3 or is using another PC and will soon be getting a new PC. Many times they will forget this when filling out the questionnaire and this question will provide the opportunity to find out future plans. This question has been found to be especially helpful when the person doing the study does not know the users or all of the future plans of the workgroup.

Asking the user if they have a hard disc on the PC is for the purpose of determining whether or not the PC applications must be loaded from the server. If the user does not have a hard disc, they will most likely want to load from the server. This would greatly impact the scheme used for setting up the shared disc structure. (See the discussion on loading applications from shared discs below and the section on performance for further information on this.)

5. Have you ever called into an HP 3000 system using a modem? This question is designed to get a feel for the user's comfort level with data communications in general. You could have users between the level of not even knowing that the terminal they have been using even has a data cable to the HP 3000, to data communications experts. For the more experienced users, this question does not supply much information, but for the less experienced users, if they have used a modem the transition to a LAN network will probably not take much training.

6. How many PC applications do you use per day? The answer to this question will help determine the setup of the user's PC in several ways. If loading the PC applications from a shared disc, it will help determine the structure of the shared discs (see discussion on loading PC applications below). Regardless of where the applications are loaded from, however it will help determine the placement of P.A.M. labels on the user's PC. Other things should be considered as a result of the answer to this question. For example, if the user frequently switches between several applications, you might consider making up fairly comprehensive batch files which disconnect shared discs, printers and plotters that are not going to be used and connect to those items that will be used for this particular application.

7. Do you have a printer or plotter attached to your PC? This question is the first question designed specifically to give you information needed to set up the user's PC. It will supply information needed to configure several applications and tells you if P.A.M. labels/.BAT files need be created to switch to the local printer/plotter. The question did not ask about printers/plotters attached through switch boxes because these devices will generally be disconnected and attached to the server to be shared.

8. How many documents on an average day do you print? This question will provide information both for determining what equipment to buy and how to configure the user's PC. In the case of what equipment to buy, the cumulative total of the number of documents printed per day by all users should help determine the duty cycle required of the printer. This information is needed by the sales representative to determine the correct printer for your needs.

In the case of configuring the user's PC, especially if the user has a printer attached directly to the PC, this value being high should be an indication that the user should be set up to automatically connect to the shared printer when the PC is booted up. A P.A.M. label/.BAT file should then be set up to disconnect from the shared printer so the printer attached directly to the PC can be used. If the number is low, do not automatically connect to the shared printer and provide a P.A.M. label/.BAT file to do the connecting.

The average number of copies should be an indication of whether to set up a P.A.M. label/.BAT file to set the number of copies.

9. List all of the printers to which you need to print. This question will help you set up access to all of the printers the user needs to access. The primary printer should be automatically connected when the user boots the PC. Access to the other printers should then be through P.A.M. labels/.BAT files. The printer location should tell you the shortname of the printer and the type of access should tell you whether to set it up for Direct Printing, Processed printing or so the PC user can have a choice according to what they are going to do.

10. How many drawings on an average day do you plot? Since all supported plotters have a higher duty cycle because they have an automatic feeder, this question is only to provide you with an indication of whether to automatically connect to the plotter when the user's PC is booted up.

11. List all of the plotters to which you need access. This question will help you determine the shortnames of the plotters and which one to set up for automatic connection vrs. connection through P.A.M. labels/.BAT files.

12. List others with whom you expect to be sharing files. This question will help you determine the structure of the shared discs on the server and what connections to provide for the user. Care should be taken when setting up connections to shared discs because many PC applications search all connected discs when doing some operations such as first starting or updating files. Instead, provide dynamic connections through P.A.M. labels/.BAT files.

Loading PC applications from shared discs:

There are a number of things to consider when planning on where the users will load applications. Not the least of these items is cost. While the disc storage space per megabyte for HP 3000 systems is becoming cheaper all the time, so is the cost of disc storage for the PCs. These costs should be carefully considered because most PC applications today are not "Networked", i.e., they are only single user applications, not multi-user. This means that even though the application program and other directly related files (such as message and configuration files) are on a shared disc, they still cannot be used by more than one person and therefore a separate copy of the application must reside on the shared disc for each user. According to how the shared disc shortname is set up, applications on an HP 3000 server could take up significant more bytes of storage than the application would take on the PC's hard disc.

An additional cost of disc drives connected to the HP 3000 system that is often not considered is the cost of the maintenance contract. The disc drives attached to the HP 3000 are bigger, more complex and faster drives designed to be repaired, but the costs of repair is not small whereas the cost of replacing a hard disc in a PC is small comparatively. It is really not a cost effective alternative to go without a service contract on the disc drives attached to the HP 3000.

Another factor to consider is performance. Although most servers will give performance between hard disc and floppy for loading the application, the overhead required to move the data over the LAN network will mean that the performance for a person that is frequently moving between applications will be better with a local hard disc.

Given all considerations, the most ideal situation for most users is to have a hard disc on the PC to load applications and use the HP 3000 for data and document files. This also gives a great deal of security, protection of the data, flexibility, and very few backup concerns for the user. The security comes from having the data on shared disc files on the HP 3000 which is generally much better protected than PCs sitting out on desks. Protection comes from the regular backups done on the HP 3000. The files can be recovered from HP 3000 backup tapes using the PC known filenames by using the RESMGR utility. The files needed to implement an application are generally available to be reloaded onto a hard disc if the disc needs replaced, but the data files on a hard disc could be difficult, if not, at times, impossible to replace.

The flexibility comes from the fact that with the PC applications on hard disc, if the HP 3000 server goes down for any reason, the PC applications can still be loaded to do some work such as creating new documents with word processor applications. The files can then be stored on the hard discs until the HP 3000 server is available again. If applications are loaded from the server and the server goes down, the PC basically cannot be used unless the applications have also been loaded onto floppy diskettes as a contingency.

For backing up files, if the data files are on the HP 3000 server, they are automatically backed up. If the data files are on the hard disc, the recommended backup procedure is to use the PC Tape Backup feature of Resource Sharing. This could,however, complicate the HP 3000 operational procedures and it requires the PC users to run the utility on their PC before going home at night. If the users do not stick around for the utility to finish, the PC will have to be left on all night. Be aware that this form of backing up files is not an archival mechanism; it is only a disaster recovery mechanism. This is because when a new tape set is started using the PC

Tape Backup facility, files from the previous tape set cannot be recovered. Files backed up using the normal HP 3000 back-up procedures, however, is an archival mechanism because these tapes can be used at any time to recover the files by using the RESMGR utility if the PC file name is known and the file is recovered back to the same shared disc.

Performance

There are a number of things to consider, when setting up the HP 3000 and PCs to use shared resources, which can affect performance of the HP 3000 and PC applications. These things should be considered as a part of your up-front planning because once the setup work is done, it is hard to make changes, especially in the structure of shared discs.

Shared device considerations:

Following are some considerations for setting up and using the shared devices on an HP 3000 server that will improve performance for the user.

Subdirectories: You should try to set shortnames to point directly to the subdirectories that contain the data files to which the user needs access as often as possible. This is because in a multi-user environment, someone could possibly remove a file or subdirectory that another user is trying to access between read operations. As a result, the path to the file must be re-verified for each access. If a file is several subdirectories deep, a lot more work need be done to verify the path for each access to the file. Remember that if you need to restrict the permission levels of different users, than more than one shortname can point to the same subdirectory on the shared disc. However, a shortname must be unique on the HP 3000 server regardless of which shared disc it points to.

Connections to shared discs: It is not recommended that too many people be connected to the same shared disc because as they access the shared disc some of the same files need to be accessed for each user. This is especially true for files that contain directory information needed to implement shared discs on the HP 3000. When multiple users are creating, deleting, or otherwise accessing a file on a shared disc, the Resource Sharing product locks the directory files to hold off other users until the first person's request is finished. Thus, with several users accessing the same shared disc, other requests will be held off giving "poor performance". If you are going to use the same account on the HP 3000 for all shared discs, it is much better to create a shared disc for global information (maybe on the PUB group of the account) and separate shared discs for each user's private

data. This will also give better security because with one primary shared disc on one group for all users, someone logged onto that group could run RESMGR and find out all of the shortnames and passwords for the shared disc and gain access to any data desired.

A separate shared disc for each person would also facilitate better management of the shared discs. If, for example, a user left the company, all of the shared disc files could easily be identified. The Automation section below explains an easy way to get an offline listing of all files on a shared disc.

Printer/plotter connections: How many connections there are to shared printers/plotters does not directly cause performance problems, however it might confuse the user if there are connections to redirected printers for the device files LPT1, LPT2,COM1, COM2, COM3, etc all at once. It is better to find out what printer will primarily be used for each application and provide the ability to specify a different printer if a different printer is desired for some reason. This is discussed further in the Automation section below.

While there is no performance consideration on how many printers are connected, there is some consideration about how certain printers are used. The performance consideration is that using Processed mode printing takes more CPU time and takes longer to get the printer output.

Creating Sessions: If possible, don't disconnect from all shared devices on the HP 3000 server during the work day because the Resource Sharing session is created when the user connects to the first shared device and is logged off when the user disconnects from the last shared device. If the user is connected to only one shared device, say his D: drive, and reconnects the D: drive to another shared disc, the HP 3000 session will be logged off when the disconnect occurs and then have to be logged on again to connect to the other shared disc. If need be, connect the user using a PC device file that is not likely to be used such as the Z: drive connected to a shared disc with no files (LAST DRIVE = Z command required in the CONFIG.SYS file on the user's PC).

Allocating Programs: Allocating certain program files that are a part of the Resource Sharing product will speed up performance to some degree. This allocation work can be done by using the MPE ALLOCATE command in the SYSSTART file or as a UDC or by enabling the Auto Allocate facility of MPE. The latter has the effect of allocating every program that is run and may require some MPE tables such as the XCST or parameters such as Maximum Number of Concurrent Running Programs to be increased before enabling this feature.

The Resource Sharing System Manager manual contains a list of suggested program files that, if allocated, could improve the user's performance when printing or plotting or when accessing data on shared discs.

P.A.M. Label Considerations:

How many connections? Generally, it is better to keep the number of connections to shared discs from one PC to a minimum at any one time. This is because many PC applications verify that they can access all connected discs when the PC application first starts up and when doing other operations such as creating a file. When the user is connected to several shared discs, this will cause the application to appear slow to load, even when loading from a hard disc. An example of this is P.A.M. for the HP 150. Each time the user exits a PC application and goes back to

P.A.M., the HP 150 looks at each connected disc to see if there are any installed applications. Especially if there are a lot of files on the directory pointed to by the short-name, i.e., what P.A.M. thinks is the Root directory is examined for installed applications. Therefore, if there are a lot of connections and the user frequently switches between applications, the user could become frustrated with this feature.

Another reason for not having too many connections is that unless an especially good job of automation is done, the user will be left to decide which drive letter to use to find the desired data. This could easily lead to loss of productivity because the user has to search for the desired file.

How many P.A.M. labels? This is a lot like the number of connections in that the fewer the better. Both the HP 150 and HP Vectra screen has 20 P.A.M. labels (4 deep X 5 wide for the HP 150 and 5 deep X 4 wide for the HP Vectra). Most users find the first 20 P.A.M. labels the easiest to use because they can be immediately seen without pressing the Page Down or any other key. All versions of P.A.M. have a feature which allows moving the labels so that the most frequently accessed labels are at the top of the screen where they are handy. It is suggested that the user be shown how to do this as it is a fairly simple operation and which P.A.M. labels are used the most may change from time to time.

For the HP 150, there is the additional consideration that P.A.M. will only look for 40 P.A.M. labels (two screens). When the user tries to access the third screen of labels or access either of the first two screens of labels from the third screen, P.A.M. must research for the labels. This makes moving between the boundary of the second and third screens slow because P.A.M. again searches all connected discs.

HP 3000 Server Considerations:

Most of the things discussed here are to be implemented before allowing users to use the Resource Sharing services. Many of these items come under the heading of performance because if not considered ahead, the users will have problems when first trying to use the services and start feeling that the performance might not be good. Its the old "first impression" theory and works for you as well as the PC users.

The first thing to do even before trying to start the data communications software is to make sure there are enough system resources configured on the HP 3000 system because you will probably have to plan on doing a RELOAD of the system to increase some of these resources.

VTERM Devices: Any session created over the LAN network will use one VTERM (Virtual Terminal) device as the $STDIN and $STDLIST device for the session. Therefore, you will need to calculate how many VTERM devices to configure in the system.

The general rule stated below is based on an average usage and tries to give a buffer so that if one or two more PCs are added to the network, you do not have to shut down the HP 3000 system to configure a few more VTERM devices. The rules are also based on how the sessions are created over the LAN network and how the sessions are terminated. Resource Sharing will create a virtual session when the first connection is made to a shared disc, printer, or plotter. Access to all other shared devices is then made through this one session. The session will stay logged onto the system until the user disconnects from all shared devices. This can be done with the USE * /D command or simply by turning off power in the PC.

Sessions created by using AdvanceLink will stay logged onto the system until the user issues the MPE BYE command or turns off the power on the PC. Therefore, if a user is connected to a shared device and logged onto the system using AdvanceLink, that one user will have two active sessions on the HP 3000 system. There is a limit in AdvanceLink that will only allow only one session that has been created by AdvanceLink over the LAN network at one time.

Sessions created by using Information Access will automatically be logged on and off as the need requires. When a session is created by this product, however, the user could have three sessions active over the LAN network at one time; the Resource Sharing session, the AdvanceLink session, and the Information Access session.

Sessions created by AdvanceMail work the same way as Information Access, i.e., they are only logged on when required. For AdvanceMail, the sessions are only logged on to upload or download HPMAIL messages.

As a result of the above, a maximum of three sessions will be active from any PC at any one time. But, this requires three VTERM devices to be configured in the MPE I/O configuration for each PC. The general rules for calculating the number of VTERM devices needed are:

1. For the first PC, configure 6 VTERM devices.

2. For each PC after the first that is NOT using AdvanceLink, configure 1 1/2 VTERM devices.

3. For each PC after the first that IS using AdvanceLink, configure 2 1/2 VTERM devices.

4. PC Tape Backup requires one VTERM device to be active on the HP 3000 server. However the user will generally not have any other sessions active over the LAN network. Therefore, additional VTERM devices will generally not be required.

Another configuration that needs to be considered is the number of buffers configured in the HP 3000 data communications software using the NMMGR.PUB.SYS utility. For the HP 3000 data communications, there are quite a few different kind of buffers that could need to be configured. It is advised that the SE installing this software on the HP 3000 be consulted. Calculating the number of VTERM devices that needs to be configured on the system should give the SE most of the data required to calculate the number of each type of buffer required.

MPE Tables: Recommendations for increasing MPE table sizes is contained in the Resource Sharing System Manager manual. These table recommendations must be considered along with the require- ments of the other HP 3000 based applications that will be used.

The fact that your HP 3000 system will probably start having many more sessions logged on will cause an increased usage of many MPE tables including the PCB, Extended CST, DST, ICS, Loader Segment Table (not measured by OPT/3000), UCOP Request Queue, Maximum Number Concurrent Running Sessions, and Maximum Number Concurrent Running Programs. Other parameters may also have to be increased such as the session limit which is often automatically set by the SYSSTART file or a logon UDC for OPERATOR.SYS.

Some MPE tables need special consideration because they take a RELOAD of the system to modify. These tables include the RIN table and the size of the system directory (your users will probably be creating a lot more files). The two configurable parameters for the RIN table are "# of RINs" which is the total and "# of Global RINs". The difference between the two values is how many RINs are left to be used as local RINs. Therefore, do not set the number of Global RINs equal to the number of RINS. The data communications software uses both Global and Local RINS. If the two values are equal, the system allocates all of the Global RINs and none are left to be used as Local RINS. The suggested values are 256 RINS and 128 Global RINs with both values increased by the number of RINs used by other HP 3000 applications you are already using. The SYSDUMP facility can be used to see how many RINs you are already using.

Virtual Memory: The LAN/3000 Link data communications software on the HP 3000 requires about 12K (Kilo) sectors of virtual memory in order to start running. In addition, you can generally count on needing about .5K sectors of virtual memory per VTERM device configured. This will allow for the process stacks, extra data segments, etc. Don't forget that the virtual memory on disc can be spread out across all discs on the system. A RELOAD is required in order to change the amount of virtual memory on the discs because it is a preallocated area on the discs.

Data Communications considerations:

The performance consideration of not disconnecting from all shared resources on the HP 3000 server except at the end of the day is discussed further in the Automation section below. But, the performance of creating these sessions can be increased using the following parameters in the NSCONTROL command:

```
NSCONTROL START;SERVER=PDSERVER.PPC.SYS,25,256
```

Where "PDSERVER.PPC.SYS" is the Resource Sharing server (identified as PDS with the NSCONTROL STATUS command), 25 is the minimum number of servers which will be in existence at all times created when the NSCONTROL command is executed (and requires virtual memory, PCBs, DSTs, Max # of concurrent running programs, etc), and 256 is the maximum number of servers.

The Network Manager:

Also, consider the performance of the Network Manager or System Operator. By setting up commands in the SYSSTART file to start up the data communications software and creating UDCs to quickly control the network, a lot of interactive work can be saved. For example, a UDC called STOPLANUSERS containing the command NSCONTROL STOP=PDS, no new users can create a Resource Sharing session, but all current users can finish what they are doing.

When setting up the commands in the SYSSTART file or in a UDC to start the network, always start all networks with the NETCONTROL command. The LOOP network is required in order to run diagnostics or use the PC Tape Backup utility. The LAN1 network (LAN1 is the default and may have been changed) is used for all other services from the PC including Resource Sharing sessions, Information Access, AdvanceLink, and AdvanceMail.

There is a list of suggested UDCs and job files which can be created to automate some of this work in the Resource Sharing System Manager and HP FastTrak manuals.

Data Security

The first rule for data security is to prevent physical access to the systems where the data resides, whether the system is a PC or an HP 3000 system. Toward this end, many PCs including the HP Vectra have optional key locks that provide various levels of security. In the case of the HP Vectra, the key lock has three positions, completely unlocked, a position where the PC can be used but the cover cannot be removed, and completely locked where the key board cannot be used and the cover cannot be removed. Being able to prevent the cover from being removed is important because it prevents removing the disc drive to steal all data on the drive. Physical security on the HP 3000 is provided by placing the system in a physically secure area. The HP 3000 can not be locked up with a key lock like the PCs because the HP 3000 is designed to be a multi-user system.

The second rule for data security is the need to know. This is generally implemented using passwords so those without the need to know cannot access the data. For servers, in order to get connected to a shared disc you need to know the server name, the shortname of the shared disc, and the password. For an HP 3000 server, the PC data files are really more secure than regular MPE files. This is because from the PC there is no way to get any "capabilities" that would allow a user to get a list of shortnames or passwords.

The Resource Sharing Session:

As with all HP 3000 systems, in order to do any work, you must be logged onto the system as either a job or session. Resource Sharing automatically logs on a session on behalf of the user when connecting to the first shared device (disc, printer, or plotter). Once this session is logged on, if the user connects to any other device, access to all shared devices is through that one session. What accounting structure the session is logged onto is the same for all users and is controlled by the Network Manager. The default logon is the user PCUSER, the account SYS, and the group PPC. Some people worry that the session being logged onto the SYS account means that they have the capabilities to possibly break or bypass the security of the system and find out passwords. This is impossible to do through this session because there is no way for the user to be able to issue MPE commands through this session. If the user wants the ability to issue MPE commands over the LAN network, it must be done by logging on a separate session using AdvanceLink. Both sessions can be logged on over the LAN network at the same time and used interchangeably.

If the Network Manager wants to change the default Resource Sharing logon, it can be done using the RESMGR utility. The logon can be changed to any account, group, and user names desired. When this accounting structure is created, no special capabilities beyond the MPE default need be given to the new accounting structure even though the session will be used to access shared discs on any account on the system. The secret to accomplishing this is the fact that a program takes on the capabilities of the accounting structure where the program file resides, not the accounting structure from which it is run. For a system where most users will be accessing shared devices and not be using the HP 3000 applications directly, it might be easier to manage the shared disc files if they are all on the same account with a different group for each user and set the automatic logon to this account. The passwords for the logon account cannot be seen using the RESMGR utility without the user having SM (System Manager) or NM (Network Manager) capability.

The RESMGR Utility:

The RESMGR utility is used to create shared discs, create shortnames and put passwords on the shortnames. This utility works on a hierarchical security scheme. If a user with default MPE capabilities runs the utility, they can create or list the shortnames and passwords for the shared disc on the MPE group where they are logged on. If the user is logged on with a user name that has AM (Account Manager) capability, the shortnames and passwords can be created or listed for all shared discs on the account on which the user is logged on.

If the user has SM or NM capability, the shortnames and passwords can be created or listed for any shared disc on the system. As a result, the capabilities given users that log onto the system using AdvanceLink should be well managed.

The RESMGR utility will also list the shortnames of all shared printers and plotters on the system regardless of the user's capabilities, but not the passwords associated with these devices unless the user has SM or NM capability. Therefore,if you want to restrict access to a shared plotter or printer, put a password on the shortname for that device. You must have SM or NM capability to initially configure the printer or plotter as a shared device.

The RESMGR utility will also allow the user to back up files on the shared disc to a magnetic tape drive. Since the data written to the tape using this feature writes the data in MPE STORE format, the recover feature of the RESMGR utility can recover the data from a tape created by RESMGR, an MPE STORE tape, or an MPE SYSDUMP tape. RESMGR, however will only allow recovering the shared disc files to the fully specified path including the same account and group from where they were stored. This means that another user cannot create a subdirectory structure on another shared disc identical to the one from which the file was orginally backed up and use the RESMGR utility to get the file. Complete shared discs can, however, be to a different account and/or group by running the PDRCOVR.PPC.SYS program directly. This program is indirectly run when doing a RECOVER operation with the RESMGR utility. This is only documented in the Resource Sharing System Manager manual, not in the user's utility manual.

The FILECONV Utility:

The FILECONV utility allows the user to access the shared discs while logged onto the system to convert PC files to MPE files, convert MPE files to PC files, and to search for the shared discs files with a file manager operation. The screens are patterned after the normal DOS File Manager found on PCs (except for delete file operation) and AdvanceLink for the file transfer operations. A user familiar with these operations can easily run the utility from a terminal or PC (using AdvanceLink) and convert files to MPE format for access by HP 3000 applications or to PC format for access by PC applications.

Just like the user can access any shared disc on the HP 3000 server from the PC by knowing the shortname and password, the same is true when logged onto the system and using the FILECONV utility. The security feature here is that unless the user is logged onto the correct accounting structure so that the RESMGR utility can be used to find the shortname and password, the user cannot connect to the shared disc.

Security of the files:

The files used on the HP 3000 to implement the shared disc file structure are all "privileged mode" files, i.e., they have a negative file code like IMAGE data base files. They are not, however, IMAGE data base files. This makes the shared disc files even more secure than normal MPE files because there are certain things that cannot be done with these files even if you are logged onto the MPE group where the files reside with SM (System Manager) or PM (Privileged Mode) capability. Since the files have a negative file code, you cannot FCOPY the file, PURGE the file, RENAME the file or access the file through any normal HP 3000 applications. With the above things you cannot do, it is difficult for even someone logged onto the MPE group where the files reside to touch the data inside the files.

Security is also provided in the way that the files used to implement shared discs are named. Since the data is in files with negative file codes and therefore cannot be directly accessed, the files used to implement shared discs are not named using any scheme based upon the PC file name. Instead, the naming structure is based on a scheme required in order to better facilitate being able to recover a file from an HP 3000 backup or store tape. Files actually containing the data are named VDT##### where "#####" is a five digit number. Therefore, a part of the security is that no one can figure out what PC data is in any particular file because it is very difficult to find out what the VDT##### file name is for a particular PC data file. Also, the VDT file does not contain the PC file name, only the data for the PC file. Other negative file code files contain the information needed to cross reference a given PC file name to the VDT file that contains the data.

The only way someone on the HP 3000 can access the data is to use the FILECONV utility (which requires knowing the shortname and password) or by writing a privileged mode program. Very few accounts should have PM capability. The shortname and password can be found using the RESMGR utility, but this utility can be secured by putting a password on the program file.

Security through shortnames:

Exactly where shortnames connect a user on a shared disc is an important consideration. The rules discussed here apply to both PC servers and HP 3000 servers. The difference is that on a PC server, there is generally only one shared disc where as on an HP 3000 server, there can be a complete shared disc on each group of each account. A shared disc is defined as starting at the root directory. Each separate shared disc on the HP 3000 server starts with a root directory.

When a shortname is created on a server, the complete path must be specified starting at the root directory. Since there can be many shared discs on the HP 3000, the path must start with the account and group name where the shared disc resides. If it is desired to have the shortname point to some subdirectory below the root, each subdirectory in the path from the root directory must be specified. For example, a path might be \USRACCT\USRGROUP\SUBDIR1\SUBDIR2\USERDIR. With this path, it can be determined that the shared disc is on the group named USRGROUP of the account USRACCT pointing to a subdirectory named USERDIR which is a subdirectory of SUBDIR2 which is a subdirectory of SUBDIR1 which is a subdirectory off of the root directory. (In other words, it is a third level deep subdirectory; ROOT -> SUBDIR1 -> SUBDIR2 -> USERDIR.) When a connection is made to this shortname, the user can create and access other subdirectories, but cannot see or access any higher level subdirectories. In fact, the user will not even know that there are any higher level subdirectories because the DIR command or File Manager will not be able to see these higher subdirectories. Therefore, the user will think that they are at a root directory.

The use of permissions can be used to control the integrity of the files. The permissions are R (Read), W (Write), and C (Create). The only valid combinations that will allow useful work are R, R/W, and R/W/C. (An often misunderstood concept is that create permission means that you can create a new file. All the create permission does is allow you to create a directory entry for the file. It takes write permission to actually write the data in the file.) Since most PC applications do some reading from the file, the W/C permissions as a combination is usually not valid.

Permissions are set and kept with the shortnames. Therefore, if you want to restrict someone to read access only, you can create another shortname that points to the files with read only permission and only tell the other person the shortname and password that has the restricted permission set.

Managing Data Security:

The key to successfully managing data security, like everything else, is up-front planning. By planning the amount of security needed ahead and setting up the structure of the shared discs accordingly, maintaining good data security will happen almost automatically with the users practicing good security habits while just doing their normal work. For example, if there are several classes of data such as manufacturing procedures, performance reviews, and future project information all on the same system used by a workgroup in the manufacturing department, not everyone should have access to all of this information. In this case, the best setup scheme might be to set up a different account for each of the three classes of information. This way, if someone needs access to some of the data while logged onto the HP 3000, they are further restricted from being able to access another class of information through the normal system secsecurity methods including being able to find out shortnames, even with AM (Account Manager) capability.

To follow this through, suppose the class of information you are setting up is performance evaluations. The people who need access to this information are all managers and at least one person from the personnel department (assume the LAN network extends to the personnel department for this purpose). However, not every manager should have access to all performance evaluations because one manager should not be able to see the evaluations of other managers, or other people working for other managers. The setup might be to create a shared disc on one group of the account. The second level manager of this group would be set up with a shortname either pointing to the root directory of the shared disc or to a subdirectory. From where ever the second level manager's shortname is pointing, subdirectories with shortnames and passwords would then be set up for each manager. Access to their own subdirectory would be set up on each manager's PC using P.A.M. labels/.BAT files (use the * as the password parameter so the manager has to type in the password) because this shared disc is not one which is used by the manager in most daily operations. The second level manager can now get access to each first level manager's subdirectories to have access to all of the performance evaluations by doing a DOS CD command to switch to that subdirectory.

If another workgroup also shares the same HP 3000 server and a second such setup is required, the same account can be used for a similar setup with a shared disc on a different group.

The personnel representative would have access to the root directory of the shared disc so that all performance evaluations can be accessed. The performance evaluation form can be placed on a single subdirectory off of the root directory so the personnel representative could modify the form if required or since the form is generally not proprietary, it can be placed in a public shared disc where forms are kept.

Automation

Having analyzed the needs of each of the users, the Network Manager can prepare ahead for automating each user thus increasing his own as well as the PC user's productivity. This section will discuss how to go about actually setting up the users so they are automated.

Standardization: Standardizing certain things will greatly help the Network Manager with the setup process, with makeing changes as the PC user's reqirements change, and with troubleshooting. This includes making up standard diskettes for installing the PC users.

For Floppy only systems, consider creating a standard diskette which will contain everything needed for the user to boot the PC and connect to a shared disc where the .BAT files reside for other work required. This diskette would contain DOS and the OfficeShare for PC software. Copies of this diskette could then be made for each user and customized with the configuration values required such as the IP (Internet Protocol) address and workstation name. The boot diskette could be set up so that once connected to the shared disc, the A: disc would be redirected to the shared disc where the .BAT files reside. Once created, a copy of the user's diskette could be archived in case the work copy gets damaged.

A standard diskette can also be created for PCs with hard discs which would make the installation work the Network Manager has to do while sitting at the user's PC easier. This diskette would probably just have the OfficeShare for PC data communications software, but the Network Manager would be able to accomplish the configuration ahead. A .BAT file could then be created which would create the appropriate subdirectories and copy all the files including the configuration files. With hard discs, it is recommended that the PC applications be loaded from the hard discs because anytime a server is used, data communications time will increase the application load time. The .BAT files discussed below, used to automate the connections to the appropriate data files, should also be on the hard disc.

Workstation Names: The workstation name is used to identify the individual users for several situations. Following is a short discussion on what a workstation name is, what it is used for, and therefore, why it should be chosen carefully to best identify the PC user in only eight characters.

The data communications software for both the HP 3000 and PCs requires a Node Name be configured. The node name consists of three fields. When used on a server, the first field is called a server name. This is the name configured on the PC workstation in the SERVERS menu of the USRCONFG utility. When used on a PC workstation, this first field is called the workstation name. This first field must be a unique name for each node on the network. The other two fields of the Node Name are called the Domain and Organization. These last two fields must be the same on all nodes of the network that will be communicating with each other.

If using a PC server, it is suggested that the workstation name be configured so that the PC server STATUS command will show who is connected. If using an HP 3000 server, there is no place to configure the workstation name. Instead, the workstation name is picked up automatically by the HP 3000 server from the PC the first time the PC connects to a shared device. The first time a new PC connects to the HP 3000 server, the name UNKNOWN is used as the workstation name until the PC has been connected for at least fifteen minutes, disconnected from all shared devices to cause the Resource Sharing session to be logged off and then reconnected to a shared device. The fifteen minutes is needed to give the HP 3000 server time to get the workstation name and "register" it by saving the workstation name and IP address in a file.

Once the workstation name is registered, it is used for several key items and thus, should be carefully chosen. The only way to change this name on an individual workstation so that the registration gets changed is to change the last three digits of the IP address at the same time. The file used to register the workstations can be deleted using the RESMGR utility, but then all workstations will have to be registered again and until they are registered, all workstations will be called UNKNOWN.

Once registered, every time the user connects to the first shared device, the HP 3000 will use the workstation name as the MPE session name for the Resource Sharing session logon. This is helpful to the Network Manager or system operator because it allows the individual PC using a particular session to be identified since all users are logged onto the system using the same MPE user and account names. The user cannot be identified using the LDEV number like terminals because when the session is created, the data communications software uses the next available VTERM device and you therefore cannot predict which LDEV number will be for which user.

This workstation name will also be used as the file name in the MPE spooler printout header and trailer. This will allow the creator of the document to be identified.

Using .BAT files: In many cases, it is a good idea to use a .BAT file to start a PC application running or to create one if none exists. This is especially true for PCs that do not use P.A.M. labels to implement single word commands that will connect to the shared discs, change to the correct subdirectories, and accomplish other work to automatically establish the correct environment. If a .BAT file is provided by the PC application, consider modifying it. If only a run command is provided in a P.A.M. label, consider moving the run command to a .BAT file. For HP 150s, this will require mofifying the .IN$ file used to install the application before the HP 150 INSTALL program is run.

Resource Sharing - A Case Study: Following is a case study that shows a number of automation and security concerns and how they might be handled.

Our sample situation has 37 employees of which 18 employees have PCs with a single floppy disc drive. Two of the employees have PCs with 20 Mb hard disc drives which are being used to control inventory, capital assets, and some accounting information. The rest of the employees are manufacturing employees who will probably be automated at a later time for entering product flow, shipping and other manufacturing related information.

We have purchased an HP Micro/3000 XE system with ThinLAN/3000 Link and Resource Sharing to be able to run an HP 3000 based manufacturing application. This is in preparation for when our product is fully developed and goes into production. The manufacturing application will be accessed from the offices using the PCs with hard discs running AdvanceLink and from the manufacturing floor over RS232 links from terminals to the HP 3000 system.

Following are issues that need to be addressed:

1. One of the requirements is that, many times, the individuals at the company move from workstation to workstation for various reasons. We want to be able to have the workstations set up so that anyone could use any workstation and access either the PC application files or the HP 3000 for the manufacturing application that runs on the HP 3000. At the same time, we want security so others can not access the files of the main person that uses the workstation or the files of someone else who has been temporarily using that workstation.

The main method of setting up the floppy diskettes we might use is to set up an automatic connection to a common shared disc. This connection would stay active until the PC is turned off. On the common shared disc is a generic .BAT file that could be used by everyone to connect to their own shared disc for the work they want to do. This would also help with performance because once the Resource Sharing session is created in the morning, it would not get logged off. The contents of the LOGON.BAT file would be:

```
USE F: \\SERVER\%1 *
F:
```

The user (either the normal user or someone else) would then issue the command "LOGON name" where "name" is the user's name. This would insert the user's name and connect the user to a command subdirectory on their own shared disc. The "*" would cause the user to be prompted for the password. The DOS drive letter would then be set to point to the command subdirectory as the drive "F:". On the command subdirectory would be several .BAT files which would then connect the user to the subdirectory appropriate for the PC application being used and load the PC application. Upon exiting the PC application, the .BAT file would then disconnect the user from the subdirectory and return to the root directory. With passwords implemented for all connects, even if a connection to a .BAT directory was left activated, no one else could access any information.

The contents of the .BAT file might be as follows:

USE G: \\SERVER\WP021 * (WP021 is the shortname for the word processor subdirectory shortname and 021 is the last value of the IP address for the workstation that normally uses this shortname. This makes setting up the shared discs for each user a fairly standard procedure. The "*" then prompts the user for the unique password for that one shared disc.)

G: (Switch to the new shared disc for loading the word processor and accessing the data files for that user that is needed for the word processor.)

WP (This is the command to activate the WP.BAT file to load the word processing application.)

F:

USE G: /D (Disconnect the user from the word processor shared disc so no one else can access the files if the person leaves the PC.)

There are a couple of things to note about this setup. One is that the connects to the different shared discs are fast because there is still a session active on the HP 3000 for accessing the common shared disc and system printer. The other thing is that this scheme might not work for those people who are going to combine files from different applications such as Graphics Gallery and Executive MemoMaker or who do access files at the DOS commands level for example to purge old files. In this case, the Network Manager will periodically list off everyone's files and ask each person to mark the ones no longer needed so he can delete them. This does, however provide the security required and provides for the fastest possible access by having the user connected directly to the subdirectory being used.

For management purposes the Network Manager is the only one that has the shortnames and passwords to connect to the root directory of each user's shared discs. To get the listing of each user's PC files, the Network Manager would stream a file on each user's group of the shared disc account which would do a RESMGR Backup function to a $NULL tape with a SHOW listing. The stream job would be as follows:

```
:JOB JUDY,MGR.RESSHARE,JUDY
:FILE PDTAPE=$NULL
:FILE PDLIST;DEV=LP
:RUN RESMGR.PPC.SYS;LIB=G
BACKUP \;SHORTNAME=JUDYROOT/PASSWORD;SHOW
EXIT
:SET STDLIST=DELETE
:EOJ
```

The reason for the file equation for the PDLIST file is so that a separate MPE spoolfile is created for the output listing so that the password in the BACKUP command line can not be seen. The ":SET STDLIST=DELETE" command deletes the job listing so it won't be printed for the same reason.

2. Administration issues - How do we make it easy for the System Administrator to maintain and administer the network? The specific concerns are:

A. Configuring Resource Sharing - DOS Directory/File emulation on the HP requires the establishment of groups within the RESSHARE account (we want to put all shared files into one account) representing the DOS Root directories. The main reason that this is a concern is that there is only one person that has completed any training on the HP 3000. This person has completed the "System Administrator" course which is an abbreviated version of the System Manager course. This means that this one person is the only person that will be able to create the required setup on the HP 3000 for new people.

But, as previously discussed, each user has a separate group on the account to better facilitate management of their shared disc. There is another consideration for making a separate shared for each user. This is the fact that there is an MPE limit of 1480 files maximum per each MPE group with the practical limit being closer to 1300 depending upon the sequence in which the files are created and deleted and the file names used.

As a result of implementing one group for each user within the RESSHARE account, each user will then initially be connected to the subdirectory containing the .BAT files. There will then be one subdirectory for the files for each PC application. .BAT files will be created with names such as "WP" (for Word Processor) which would change the data disc drive pointing to the subdirectory containing the .BAT files to the correct subdirectory and load the word processor from the application subdirectory. The required steps to implement a new user are: create a new group, the shared disc, required subdirectories, and then the required shares. We could also set up UDCs to create the group and put on the passwords and another UDC to change the passwords. Then, to implement another PC application, all the Network Manager will have to do is to go to the person's PC and, while connected to the root directory, create a new subdirectory for the new application, create a corresponding .BAT file, and using RESMGR, a corresponding share.

B. Should it ever become necessary to change any of the Resource Sharing directory relationships, the Network Manager would be required to modify each of the workstation diskettes that may be affected. But, by putting the .BAT files on the server this is not required. Simply create the new .BAT file on the server and the user is set up.

C. Backing up and restoring the DOS files presents another complication in that although the DOS files get backed up along with the MPE (HP 3000) files, they can only be restored by using a Resource Sharing utility which reconnects them to the DOS directories. This suggests that in the interests of efficiency, the Resource Sharing files should probably be backed up separately from the MPE files.

Another alternative would be to set up a job file that would do the SYSDUMP and store the files from the RESSHARE account first (file set @.@.RESSHARE,@.@.@-@.@.RESSHARE). This would allow almost as fast a recovery as backing up the files on a separate tape, especially since all of the shared files are in one account.

3. Space Utilization - How effectively are the DOS files stored on the network hard disc? Specifically, under DOS emulation, it is not known beforehand how big a file will be. Therefore, the HP server uses a pre-configured maximum file size which it divides by 32 for the size of each extent. Thus, the smallest amount of space a file takes up is 1/32 of the file size configured for that shortname. If the largest file expected is 200,192 bytes, then a single extent (the smallest file) will take 6,256 bytes. This represents a lot of unused or wasted space.

The way to control this (in context of the other concerns expressed above and the resulting implementation) is to have different shortnames for each user with the associated file size set appropriately. For example, for the user Tom, there would be the following commands and corresponding .BAT files - "TOMWP" for Tom to access the word processing application with the file size specification set smaller; "TOMS" for Tom to access the spreadsheet application with the file size set to a medium value; "TOMDB" for Tom to access the data base application with the file size set for larger files. Each command would connect to a different shortname for the reasons of faster access by being directly connected to the appropriate subdirectory and for the more appropriate file size specification.

4. Data Security - Password protection must be embedded within the NETPARM file on the individual workstation diskettes. This allows anyone to get access to other's files if they get a hold of the diskette.

This feature is implemented into the "OfficeShare for PCs" software. However, it may be decided not to implement this feature since it would inhibit others from using any workstation, especially if the person that normally uses that particular workstation did not happen to be available to start up the workstation that day. The reason that the normal person not being available would be a problem is that one of the main desires is that people did not have to carry around diskettes in order to use another workstation. Instead, it is suggested that the only automatic connection that be done each time the workstation is booted be to the main DOS drive that contains the generic .BAT file discussed above.

Final Configuration:

As a result of all of the above concerns, the following configuration would be used:

1. For the Server - Create a shared disc on the PUB group of the RESSHARE account with only a root directory for holding the LOGON.BAT file required to implement the ability of any user being able to use any workstation. Create a separate group on the RESSHARE account for each user with a shared disc. Within each shared disc, have a separate subdirectory for the files related to that application and a subdirectory for the .BAT files to connect to the application subdirectory.

In the subdirectory containing the .BAT files, include both the files to connect to the application subdirectories and the files to control access to the printers. This includes .BAT files for connecting to the network printer, disconnecting from this printer (some users also have local printers), changing the number of copies to be printed, etc.

2. For the PC, have a common disc that can be copied for each PC. Each PC would thereby be configured so that each time the PC is powered on or re-booted, it would load DOS and run the USRLOAD utility to connect to the system for the global disc access and, if the user did not have a local printer, connect to the network printer.

To implement this, after copying each diskette, the Network Manager will have to use the USRCONFG utility to modify the following values in the NETPARM file that is put on from the master diskette:

A. IP address.
B. Workstation name.
C. If appropriate, delete the network printer automatic connection.

VOICE AUTOMATION - THE NEW SEGMENT OF OFFICE AUTOMATION

Don Crosbie
InterVoice, Inc.
1850 N. Greenville Avenue
Richardson, Texas 75081

SUMMARY

With productivity the driving force, office automation has spawned telephone call or Voice Automation. In the sections that follow, this new segment will be defined and placed within the office function. Perspective on the evolution of voice systems will lead to the question "Why will voice systems sell now?" Answers will be suggested such as "What new factors cause the predicted growth in the market?" "What are the productivity benefits that drive it?" And finally "What are the unique functions of the RobotOperator™ System by InterVoice, Inc."

VOICE AUTOMATION

Voice automation is what we address and to position it we speak of the "four faces" of Voice Automation. The first face is call directing or automated attendant. The second face is voice messaging - the most common use being voice mail. The third is dictation and transcription. Most of us think of this as using a dictation machine and then handing off the tapes for someone to transcribe. You may do just that in a digitized, random access method; allowing priorities in transcription to be determined. For every bank that takes wire transactions there is a device currently which records every voice communication. It is an analog device; i.e. is it is on tape. However if it is digitized and stored it can be randomly accessed and transferable to a digitally keyed index database.

The transcribing is itself a major task. For efficiency, silence must be detected, and comands for speed control, backup and repeat allowing transcriber convenience.

The fourth area is the most traditional use of voice automation; interactive data base information access.

When you look around the office you see people with phones and CRTs talking to people who call. Anytime you see just 2 people doing this, they become a candidate for automation of voice processing. I say candidate because if companies had their preference and cost was no object, they would use human operators. The facts are, however, that RobotOperators have gained acceptance through enhanced operator functions or features.

The PBX operator is a candidate for replacement on the routine calls through provision of direct dialed access to

the party or department being called .

This new combination is really the merging of computer and telephone. This merger is not easily accomplished in fact I would like to suggest that it is the area least attended to among the office automation functions. People with PCs perform multifaceted tasks but little is done about automation to get the folks off the phone allowing human agents more time to handle those items that sell more and provide more customer service. So we propose the merger of computers and telephones with the RobotOperator as catalyst as shown in Figure 3.

The traditional way that of callers interact with voice response systems is signalling the computer with a TouchTone telephone. What's the functionality of this device? Very simply, just a few . Data entry, using the touchtone pad. Information inquiries are the prime use - information inquiries such as; what is my balance? has a check been paid? Then voice messaging for callers to leave their name and phone numbers, for later call back.

Financial institutions are the major users today. There are about 1,000 banks presently utilizing voice systems. However, that may not be the largest future market. Retailing represents a large opportunity. These are still primarily financial transactions - credit authorization; amount open to buy, date of the last payment credited and amount and pay-off amount on a loan? However, catalog order entry or television orders to "800" phone numbers could be the largest voice automation function of the future.

Credit bureaus for check cashing authorization is yet another large financial tansaction area for automation.

Education - Class registration as you may remember is a tiresome experience. Now, do it by telephone. Save a lot of time; effort and money. One doesn't have to go to campus. You can call-up and register and even pay by Mastercard, Visa/etc.

Government - The largest systems in voice automation are in the IRS. "When do I get my refund?"

First Alabama Bank's slogan says it all "Want to talk to your Money?" "Anytime, Anywhere."

VOICE AUTOMATION IN PERSPECTIVE

A history of voice automation will establish background in voice response. In 1963 IBM released the first commercial unit with analog voice stored on a drum. It spoke somewhat clearly but not very many words. It did access the database of the IBM 360 computer. Tellers were the primary users as there were no on-line teller machines. They found out

balances and placed stops and holds. The price for a 4 line unit was about $200,000. Systems were so costly that only big banks bought them, - some 800. In 1972 Periphonics came out with a lower cost and much more human sounding voice. The patented method of analog recording on a disk provided a larger amount of voice, with better quality. In 1978, IBM made the second attempt at voice with the Series 1 computer. The price came down but the major innovation was that the script application was on the Series 1 front-end instead of in the IBM host. The front-end was a lot easier to program than the IBM 370 computer.

Perception Technology voice response came out in 1980 with all of the above functions plus they reduced prices so that voice response was available to smaller users in more places.

In 1985 InterVoice began incorporating all of these things in the RobotOperator with even lower price, higher quality and quantity of voice and script application created through a simple "authoring" system that eliminated host programming by creating the interface within the PC itself.

Why should RobotOperators grow now after these 25 years of being a limited business opportunity?

ROBOTOPERATORS - WHY NOW?

I would like to suggest several factors.

1. The application can be created easily.

2. Consumers now accept signalling the system through touchtone and other input as the technique has proliferated.

3. Consumer convenience - user friendliness - helps gain acceptance.

4. Database access has become much simpler without programming. Thus, it is easy to install.

5. Now systems network with other communication systems so instead of having 800 numbers - WATS lines - systems can be dispersed into the major cities such that many calls are local. Systems pay off through offset of phone costs.

6. Lower cost is the final item.

Let's discuss each of these areas:

What are some applications?

A. Ordering products. As soon as Sears prints its catalogue - before it is mailed - the prices are out of date. So people ordering over the phone get the right price and order seven days using the RobotOperator.

B. Time and Temperature. Most banks offer time and temperature by phone or signs.

C. Locator service? You've got a tape recorder, a Sony, and you don't know where to get it fixed. If you could just dial a Sony '800' number and hear "The repair center is at 7th and Main." Or where is the nearest ticket agent for American Airlines? "It is located at the Hilton Hotel, North Central Expressway and Campbell Road in Richardson."

   How does the customer locate himself so he can find out where to go? By keying the telephone area code and the number listed on the phone in front of him.

D. Product information description is already available through published '800' numbers. Why not automate the description - any time a customer wants to call.

Consumer Acceptance of the RobotOperator has grown immensely. For instance "banking from the bedroom" by telephone allows all the same transactions as an Automated Teller Machine except cash dispensing. What enhances customer acceptance? Well, any voice can be digitized and it sounds like a real human.

Calling in from anywhere, anytime to find out information is available from RobotOperators. If you make a credit card telephone call today you're asked to key-in your credit card number. Thereby, the phone company is helping us; they are teaching consumers how to use TouchTone® better and making consumers more aware of the technology.

Consumer Convenience

A new facility is the capability to handle a rotary dial/pulse telephone. The caller speaks menu number selections. VoiceDial provides speaker independent voice recognition in lieu of using the keypad on the TouchTone telephone to signal the computer. This allows all consumers to access the systems.

Another method of input is data - communication through a dial-up terminal. From the office, a company treasurer calls in to get detail information regarding his company account. The RobotOperator allows its same resources to detect PCs calling. Thereby the RobotOperator switches to the data

mode and sends ASCII code back and forth instead of voice.

Another factor that makes voice systems more friendly to callers is that it talks a lot - not only quality speech but a lot of words - measured in hours.

The RobotOperator doesn't change consumer habits. The normal caller access equation is not changed; that is, the caller calls into an intermediary who brings up a screen from a computer database and then speaks back the information they want. The InterVoice equation substitutes the RobotOperator for a human agent. Figure 3 shows the RobotOperator caller equation. If the RobotOperator's menu does not include answers to some caller questions then they are "switched" automatically to human agents.

In merging into the caller equation the InterVoice PC "engine" is in the mainstream of office automation and networking using Local Area Networks (LANS).

Simple Installation

Another factor is the "authoring" capability. Script writing is a tough task. How will the caller accept prompting and once you do a script you know its going to change. With InterGen the system provides simple tutoring tools that create and change scripts without programming.

A further facility is database access without programming on the host. Just as a human agent reads off the screen, the RobotOperator reads the same screen data and translates the data to words. Whatever screen the human gets currently the RobotOperator can read the same data with ScreenScan so no programing is required.

Recording must be equally simple. Recording right over the phone is convenient, quick and easy.

Networking

Companies are networking and it is based in PCs. LANs combine office files and tasks. Networking is where office automation is moving. Being consistent with PC networks is the right direction, whether Novell, Ethernet, Token Ring etc.

Another type of networking distributes the RobotOperator Systems to major local calling areas using the existing CRT network. Just drop another screen display next to existing ones. Long distance, FX and WATS costs offset pays for the RobotOperator Systems.

One other factor is necessary. Control the network systems and know that all lines are talking; know what the statistics are at each location; know central information through

the remote console with on-line, real-time diagnostics and control.

Cost Benefit

The final point is low price. The historical price scale of voice systems allowed only big companies like American Airlines and bigger banks to use the technology. Today there are many small banks and credit unions using voice response. Why? Because they cost $20,000 or less. Four RobotOperators for the cost of one human. So when you come down that cost pyramid many businesses can buy. Lower prices for PCs also helps lower the costs. Selling through a third party actually makes systems less costly. Value-Added-Resellers have expertise and credibility with their customers and "value add" to their own applicational systems.

The cost benefits can be summarized as follows:

- Agent productivity enchanced.
- Extended customer service hours.
- Reduced long distance charges.
- Reduced number of agents.
- Increased customer service.
- Consistent service level.

Vectra PC

The benefits of voice automation using the RobotOperator system by InterVoice may reside in the Vectra PC and interface to Hewlett Packard mainframes emulating a single ASCII, asynchonous terminal or emulating common IBM 3270 or Burroughs Poll Select terminals.

The unique combination of benefits from InterVoice

1. Based in the PC engine which is the heart of automation.
2. Expandable to accommodate as many as 32 RobotOperators and access up to two hosts on a single PC module.
3. Network expandable to handle any number of RobotOperators; i.e., hundreds, even thousands.
4. Listens to TouchTone, data, VoiceDial - voice recognition signals - as well as recording the voice directly for transcription or play back.

5. Interfaces easily to computer databases without programming.

6. VirtualVoice™ provides lots of voice storage accessible by all callers.

7. Easily created scripts makes it responsive to users needs and changes made with InterGen™.

8. Modularly expandable to provide all "faces of voice automation".

9. Low cost and integrated in the Vectra® PC.

FIGURE 1 InterVoice RobotOperator Concept

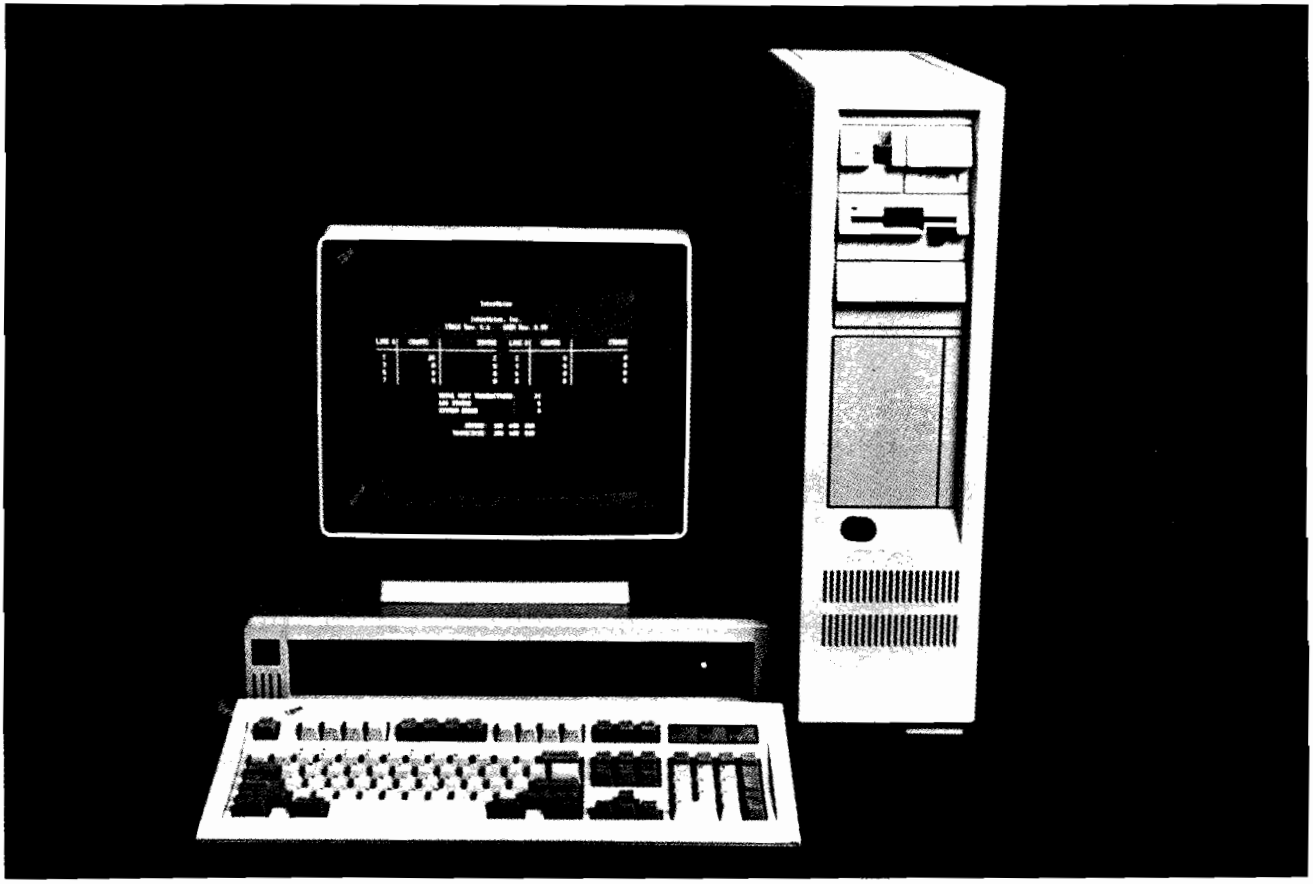


FIGURE 2 InterVoice RobotOperator System

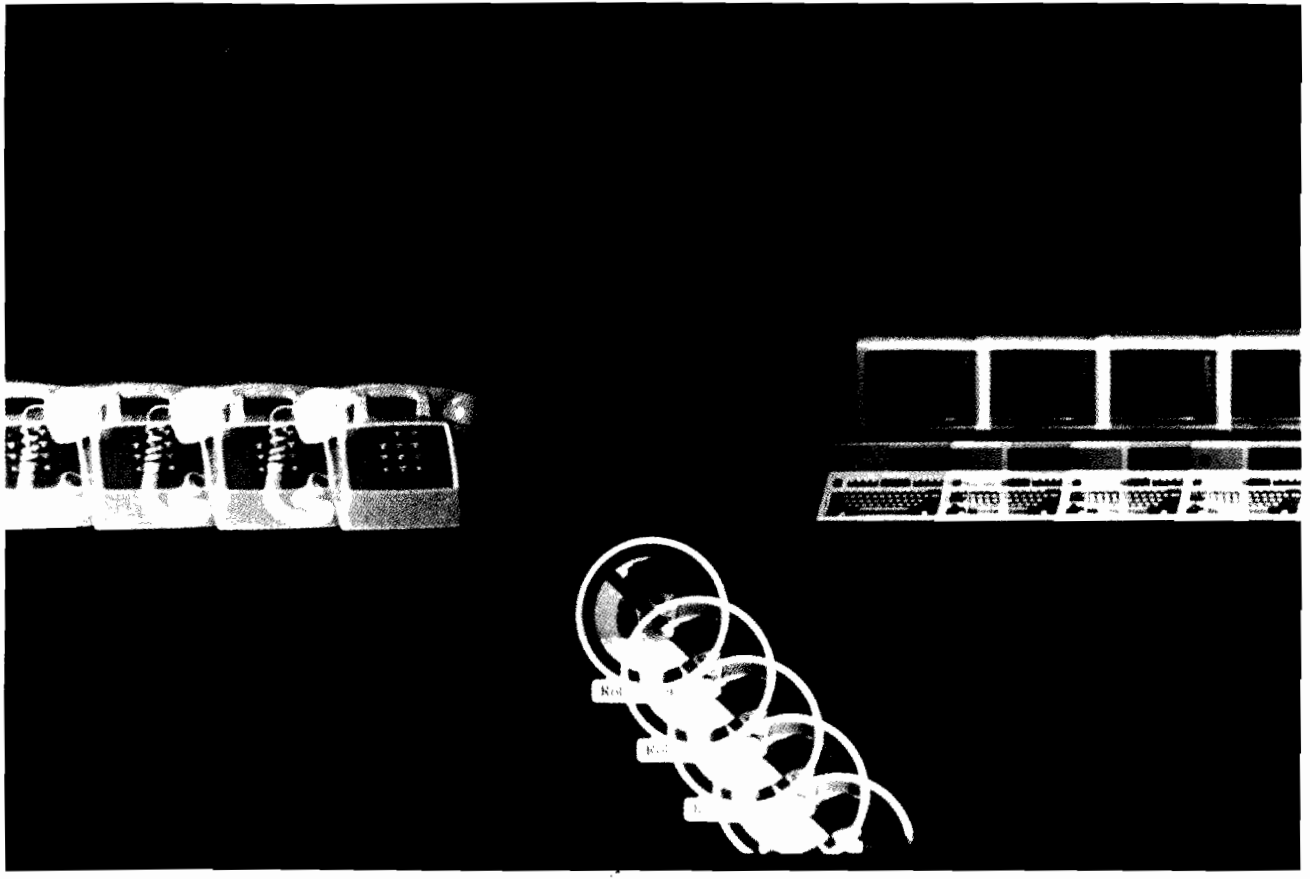


FIGURE 3 Metamorphosis - Telephony combined with computers with RobotOperator Catalyst.

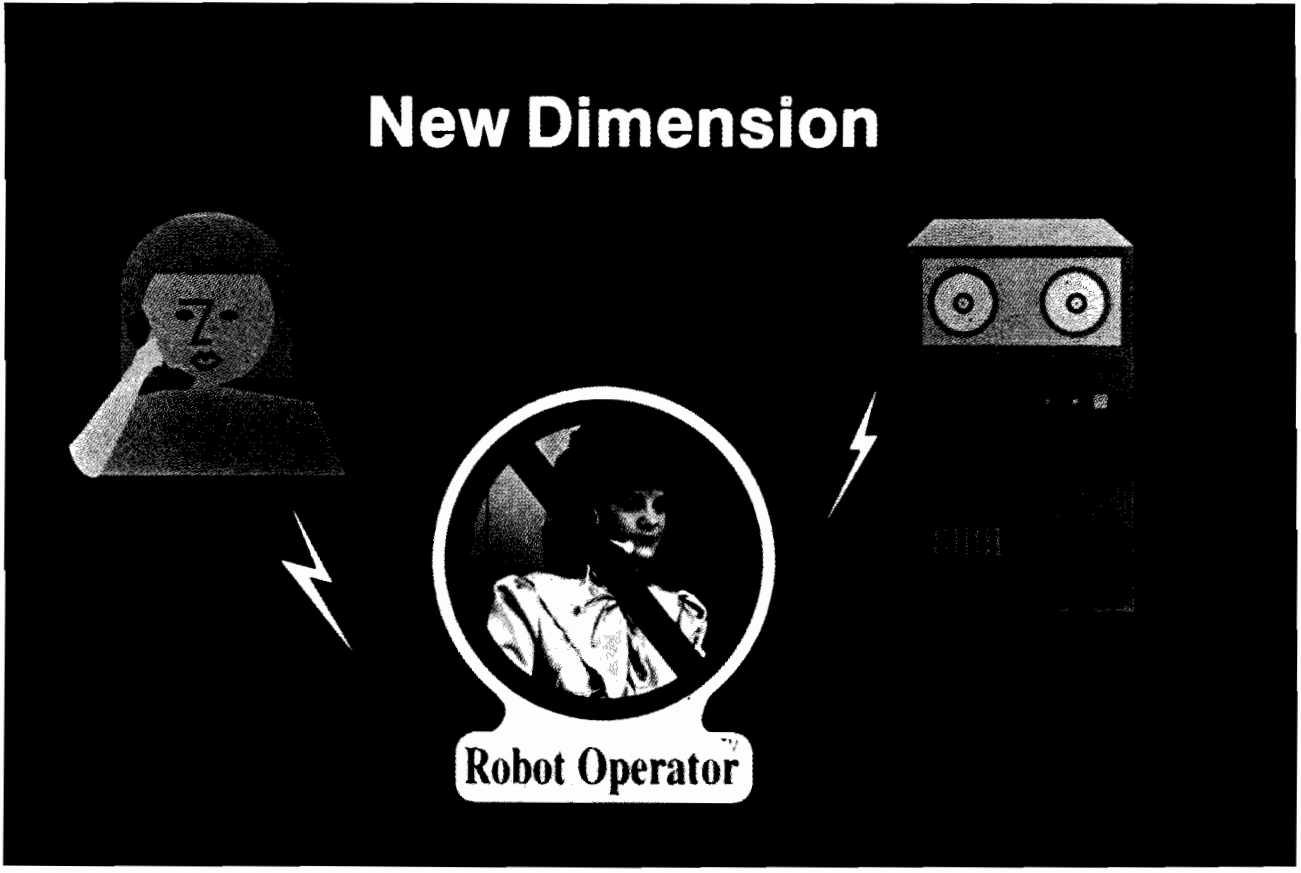


FIGURE 4 Caller Equation

BUILDING DISTRIBUTED APPLICATIONS WITH HP ADVANCENET

JIM CUNNINGHAM
HEWLETT-PACKARD
CUPERTINO, CALIFORNIA

INTRODUCTION

A basic set of network links and services is now available for the development and management of applications in distributed computer environments. NS3000/V provides a set of Network Services to the builder of distributed applications, including Network File Transfer (NFT), Remote Process Management (RPM), Network InterProcess Communication (NetIPC), Virtual Terminal (VT), and Remote Data Base Access (RDBA). The Network Services can be used to build distributed applications and are particularly suited for situations where real-time synchronous transfer of information is crucial.

Once these network links and services are in place, the builder of distributed applications faces the challenge of how to use the network services to solve his business problems. These basic services provide the application developer with a starting point to build distributed applications; however, potential programming pitfalls await him. What does the developer do if a remote system is unavailable temporarily, or if a network link is down, or if the program on the remote system is not running? How does the developer ensure that information his application needs is delivered reliably? When multiple applications are using the same link, who determines which data transfer has higher priority? NetDelivery, a new networking software product from Hewlett-Packard, provides these services and more.

NetDelivery is a distributed application service for the HP3000 family of business computers. The NetDelivery software provides reliable, transparent delivery of information between user-developed applications, regardless of the network configuration or state of the network links between HP systems. This paper discusses some common scenarios in today's distributed computer environment and how NetDelivery provides a solution to the developer of distributed applications. NetDelivery allows the application programmer to concentrate

on the application and not be concerned with the details of network topology or the state of the network.

NETDELIVERY, A DISTRIBUTED APPLICATION SERVICE

NetDelivery is a distributed application development service that acts as a message and file delivery system for HP AdvanceNet networks. NetDelivery is designed to simplify the development and management of applications for users in distributed computer environments. Through a programmatic interface, the NetDelivery software allows users to write customized applications that exchange information transparently and reliably between applications in different locations.

NetDelivery has two components -- a set of intrinsics, i.e., a programmatic interface from which customers can build distributed applications, and a utility program that allows a network administrator to manage NetDelivery applications (i.e., control, diagnose, and maintain).

NetDelivery uses HP Network Services' Network File Transfer (NFT), Network InterProcess Communication (NetIPC), and Remote Process Management (RPM).

NetDelivery provides a **reliable delivery** system by queueing each message to a disc file. (A "message" refers to a stream of data bits that can be text, graphics, or formatted data.) If the remote node is unavailable or the link is down, the delivery software periodically retries to transmit, thus guaranteeing the successful delivery of messages according to assigned priorities. Reliable delivery is critical to many distributed applications. For example, when the message is an order for a boxcar of goods, it is unacceptable for the message not to be delivered or to be delivered twice.

NetDelivery provides an **asynchronous** client-server model of communication. The client (sending or requesting) application can initiate a data transfer and then continue processing without waiting for the data transfer to be completed. Meanwhile, NetDelivery queues the

message to a disc file on the remote node and (optionally) activates the server (receiving) process. In contrast to the synchronous nature of NetIPC, the sender, receiver, and network do not have to be active at the same time.

NetDelivery's features combine to provide **network transparency** to the distributed application developer. The sending application need only know the name of the server's queue, and not the node on which the server resides or the topology of the network. The server application can be moved to another node on the network with no changes to the client application. This feature allows applications to be built on a single development machine and then moved onto the network without change. This feature also allows for modular growth of distributed applications. This, in turn, provides for ease of integration of new applications into existing distributed applications. The application designer can thereby move the processing to the data, keeping the data where it is managed and distributing the processing across the network.

NetDelivery provides **transfer scheduling and control** for distributed applications. Messages are scheduled for transfer according to priority. Transfer time specifications that determine when message transfer will take place between two nodes can be configured. The transfer is based on message priority, so a common transfer policy can be worked out among various applications on a network.

NetDelivery also provides **automatic server invocation.** NetDelivery can be configured to initiate a server program or job stream whenever a message is placed in the server's queue. This feature can be used to optimize computer resource utilization through the ability to execute an application only when required.

In a wide area network, NetDelivery provides a **store-and-forward** capability. NetDelivery does not require all links between the source and destination nodes to be available simultaneously. NetDelivery allows messages to be stored on intermediate nodes between

source and destinations so that the message can progress from source to destination whether or not all links are ever available at the same time.

NetDelivery messages can also include **attached files**. This simplifies porting existing applications that already communicate through shared files. NetDelivery provides the capability to **reply** to a message. Replies, which can also include attached files, can be used to return information to the requesting client. If an application server needs to route a request to a more appropriate server, NetDelivery can transparently forward the message. Additionally, NetDelivery supports multiple destinations in the form of **lists**, which are simply lists of message queues.

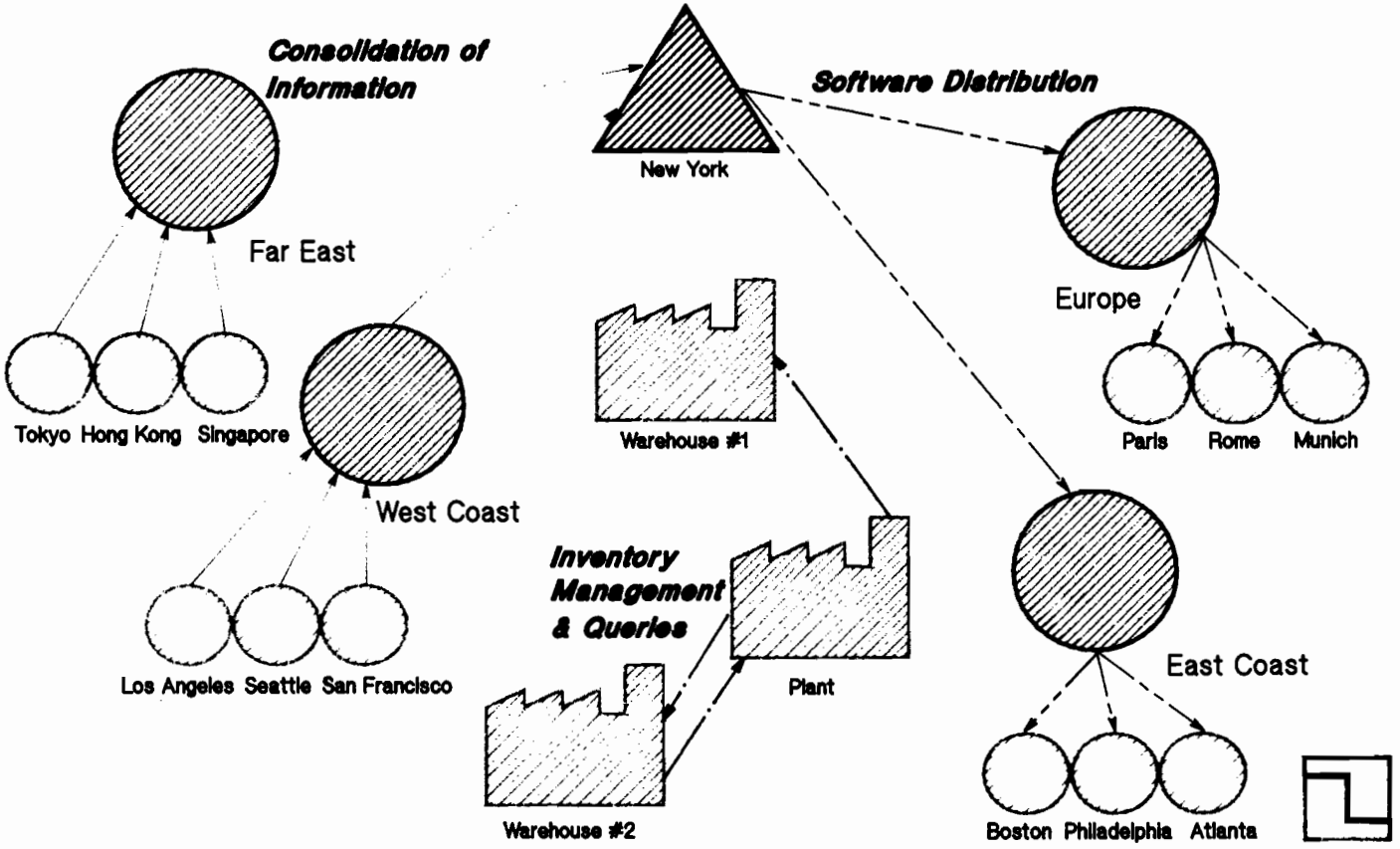
THREE DISTRIBUTED APPLICATION SCENARIOS

Consider the example of a multinational corporation, a manufacturer of consumer goods with corporate headquarters in New York, a West Coast regional sales office, with branch sales offices in Los Angeles, San Francisco, and Seattle, an East Coast regional sales office, with branch sales offices in Atlanta, Boston, and Philadelphia, and distribution centers, manufacturing plants, and warehouses in the Midwest.

As software applications are developed and revised by the Corporate MIS group, the new program files and job streams need to be distributed to the regional and branch sales offices and to the warehouses, distribution centers, and manufacturing plants. This currently may be done by magnetic tape sent via overnight courier. This scenario is referred to as **Software Distribution**.

As salesmen at the branch sales offices fill out their weekly expense reports and sales orders, the branch information is consolidated and shipped to the regional sales office. At the regional sales office, a consolidation application compares the week's data to quotas and past results, updates its database, and sends the summary to corporate headquarters. At headquarters, a program consolidates the data from each region, and produces a report for

The Problems



the Chief Financial Officer, as well as reports for each regional and branch sales office. This scenario is referred to as **Consolidation and Uploading Information.**

As shipping orders are received at the distribution center, a database inventory application monitors inventory level and requests new shipments from the manufacturing plant as needed. The manufacturing plant uses a just-in-time delivery system to obtain raw materials in its manufacturing process from the warehouses. The manufacturing facility needs to locate raw materials in the warehouse nearest to the manufacturing plant. It is important that the raw materials inventory databases at the warehouse and at the manufacturing plant be kept in synch. The application running at the manufacturing plant must be able to query the warehouse database, determine whether it currently has the needed materials and automatically update the database. This scenario is referred to as **Inventory Management and Queries**.

NETDELIVERY IS THE SOLUTION

It is likely that the needs described in the three scenarios are currently being addressed by three different teams of application programmers. But each needs to solve similar problems, and each can benefit from NetDelivery's distributed application services. NetDelivery provides a common framework and allows a consistent information transfer policy within the network. Each group can solve its business problem separately with NetDelivery; however, some coordination between the application developers and the network administrators is necessary to ensure a consistent transfer policy.

Let's assume HP3000s are in place at the various locations, that they are connected by wide area links and that NS3000/V has been installed on each node. The NetDelivery software can then be installed on each node. The network administrator must work with the application developers to devise a consistent transfer and message priority policy. They decide that database queries have a relatively high priority, financial uploads have a medium priority, and software updates have a low priority. At each NetDelivery node, the network

administrator, using the NetDelivery utility program, sets the transfer policy of sending medium messages after 5 pm, low messages after 11 pm, and high priority messages on demand. The network administrator also uses the utility program to set up store-and-forward relay nodes at strategic places, e.g., the regional sales offices. The administrator then configures the message queues and distribution lists as required by each application. Note that some configuration (local queues, routes, and lists) needs to be done at each NetDelivery node, while other configuration (e.g., more "global" queues, routes, and lists) can be done at a single node. The NetDelivery software then propagates the configuration information to other nodes.

NetDelivery can be used to solve the software distribution problem. At each NetDelivery node, a server queue is configured with the AddQueue command of the utility program. The queue is configured so that the server program will be automatically invoked by NetDelivery when a message is placed in the queue. This server makes a backup of the old software, updates a version number in its database, installs the new software, and replies with its status.

At the corporate headquarters, a distribution list is configured with the AddList command from the utility program. The list is constructed to contain each of the server queues as members. The client (sending program) sends to the list. The program files and job streams are sent as attached files to the message. The NetDelivery portion of the client process is essentially:

```
ncSignon();
ncSend ("distribution list","message",priority:=low,request_reply);
ncSignoff();
```

The server is identical on each node, and the NetDelivery portion is essentially:

```
ncSignon();
ncOpenQueue("queue");
ncReceive("message");
ncReply();
ncCloseQueue("queue");
ncSignoff();
```

In this example, NetDelivery offers these features:

- Upon successful return from the ncSend intrinsic, the sending application is ensured of reliable delivery.

- The priority scheme, in conjunction with the transfer policy, allows the software updates to happen at non-peak hours and not interfere with normal operation of the other applications.

- The server is initiated by NetDelivery when NetDelivery places a message in the queue, so the application server need not always be running.

- The software distribution server is identical on each node. This program can be tested and debugged on a single node and then installed throughout the network.

- Although the message is sent to multiple destinations, only one copy is sent across a network link or stored on disc by NetDelivery.

- If a link is down, let's say between the West Coast regional sales office and Seattle, NetDelivery stores the message at the regional sales office and periodically retries until the link is available. This is an example of the store-and-forward capability of NetDelivery.

- The sending applicaiton can use the reply feature to verify that the software has been successfully installed on each machine.

The NetDelivery solution for the Consolidation and Uploading Information scenario is similar to the above example. The financial data to be uploaded can be sent as an attached file, most likely in the same format as is currently used. This illustrates another important feature of NetDelivery. NetDelivery does not require data format changes, thus making it easy for the developer to convert an existing application to use NetDelivery.

The NetDelivery program segments for the third scenario, Inventory Management and Queries, are also similar to the previous ones. Here are examples of the use of NetDelivery's forwarding feature and location transparency.

For the forwarding example, let's assume that a central warehouse contains a raw materials inventory database. The central database keeps track of inventory levels at several other warehouses. The sending application sends all requests for raw materials to the central database server. This server determines which warehouse can fill the order and then uses the forwarding function of NetDelivery to forward the request to the appropriate warehouse. The database server at that warehouse acts on the request and replies to the sending application.

As an example of the benefits of location transparency, let's assume the central warehouse server is moved to another location. The configuration entry for this location needs to be modified with the UpdateNetQueue command of the utility. The NetDelivery configuration server then propagates the new location to other nodes. The client sends to the location independent name as before and the application continues to work with no changes.

CONCLUSION

NetDelivery provides a solution to the developer of distributed applications, as illustrated in the three scenarios of Software Distribution, Consolidation and Uploading Information, and Inventory Management and Queries. NetDelivery is suited for a large class of distributed applications including distributed database maintenance, bulk data transfers, document

distribution, and many others where reliable delivery is important. For those applications where immediate response time is more important than reliable delivery, the application writer can invoke NetIPC directly. An example of such an application might be a remote query where a user is expecting an immediate response. If the query is lost, the user can simply retry himself. This is a case where immediate response takes precedence over reliable delivery.

NetDelivery provides many features beyond the basic Network Services. The features of NetDelivery make it a valuable service for the developers of distributed applications. The designer can concentrate on solving problems at the application level, while NetDelivery provides a reliable, transparent, asynchronous message and file delivery system.

THE BENEFITS CAN BE ASTOUNDING!

KIMBERLEE S. DAVIS
MARTIN MARIETTA DATA SYSTEMS
DENVER, COLORADO

SUMMARY

I. INTRODUCTION

"YOU CAN JUDGE YOUR AGE BY THE AMOUNT OF PAIN YOU FEEL WHEN YOU COME IN CONTACT WITH A NEW IDEA."

Historically, attempts to implement report-writers for end-users generally proved unsuccessful. Products were too complex and inflexible to be properly used by non-technical personnel. Consequently, a very skeptical attitude toward report-writers resulted. This attitude has remained and is the primary reason why end-users and data processors are apprehensive to try implementing these tools today.

The purpose of this document is to present information that will encourage both technical and non-technical personnel to reassess the potential value of today's report-writers. The paper will begin by discussing why past attempts to use them failed and the lessons that were learned from these failures. It will then continue on to discuss the technical metamorphosis of report-writers that makes them practical in today's business environment. The paper will conclude by presenting proven guidelines and standards for successfully implementing a report-writer for end-user use.

(THE BENEFITS CAN BE ASTOUNDING!!)

II. PAST FAILURES/LESSONS LEARNED

"THE LEAST EXPENSIVE EDUCATION IS TO PROFIT FROM THE MISTAKES OF OTHERS."

In a progressive business like data processing, it is not uncommon for ideas to be born before their gestation is complete. Although this doesn't make the original concept perfect or imperfect, premature birth of a good idea can kill its chances for survival. This is what happened with the idea of report-writers being utilized by end-users. A notion was conceived for end-users to take the responsibility for simple reporting requirements. However, as the idea developed, competitive greed and impatience lead to premature marketing of this concept. Products were purchased by naive individuals listening to vendors "promising the world," but because the birth was premature, the original idea had to struggle to survive.

REPORT-WRITERS TOO COMPLEX AND TECHNICALLY DEMANDING

As a result of this untimely birth, early report-writers were nothing more than new, limited programming languages, sometimes coded on special forms. Problems with the products included the facts that were not user-friendly, required extensive training, required time consuming support from the data processing department, and many were specific to a single vendor or vendor application. In addition, still being logic (and not process) oriented, they offered no more simplicity than standard languages like Basic, Fortran and Cobol. No additional value was added by the products in this embryonic stage.

USERS STILL LESS SOPHISTICATED THAN TODAY'S 'DIRECT USERS'

In addition to report-writer products being immature in development, the users that they were being marketed for were also in the opening stages of their life cycle. When report-writers were first introduced, end-user's roles were different than those of today's user. Today's users have had technology put at their fingertips. They are more educated and demanding. Products like personal computers, spreadsheets, graphics, and office automation hardware and software were non existent to the users who attempted to utilize early report-writers. Most end-users were still skeptical about automation, insisting the computer threatened - more than strengthened - their jobs. The users of yesterday had no motivating factor to encourage their use of report-writers.

REPORT-WRITERS WERE NOT COST EFFECTIVE

Because early report writers were technically too complex for the non-sophisticated users that were expected to use them, there are many reasons for the dissatisfaction and failures that resulted when trying to unite them. However, they all point to the fact that report-writer use by end-users did not prove to be cost effective. The time spent to code a report, debug it, test it and produce it rarely justified the value added by it. Under these circumstances, failure was eminent.

LESSONS LEARNED

In summary, we have learned that because early report-writers were technically too complicated for the unsophisticated users that were attempting to utilize them, they were not cost effective. Therefore, the attempt to satisfactorily implement report-writers for end-user utilization generally proved to be a failure.

In order for the report-writer/end-user union to be a success, the report-writer products needed to become less technical and the end-users needed to become more technical. This was the only way that report-writers being used by end-users could become a satisfactory, cost-effective, successful concept.

III. TECHNICAL METAMORPHOSIS OF REPORT-WRITERS

"CONSTANT CHANGE IS HERE TO STAY."

Fortunately, "Constant change is here to stay." It is this constant change that has brought about the evolution of both report-writers and end-users that was necessary for a successful union of the two.

METAMORPHOSIS OF REPORT-WRITER PRODUCTS

In the past five years the emergence of the 'fourth generation' of data processing has occurred. Because of the increased sophistication fourth generation technologies have provided, many report-writers have metamorphosized into products that can now be productively utilized by non-technical personnel. The primary changes that make this possible include:

A. Report-writers are now user-friendly, on-line, menu-driven applications

B. Report-writers are now procedure driven instead of logic driven

What this primarily means is that an end-user can "fill in the blanks", "answer some on-line questions" and then the 4th generation technologies produce the code and the report. By eliminating logic oriented coding and debugging, the time necessary to produce reports is now significantly reduced.

METAMORPHOSIS OF USERS

In addition, today's businesses are heavily dependent on personal computers, word processors and other office automation tools. As a result, the role of today's users have drastically changed. They are more educated than the users of the past. Their fear of automation has been buried and replaced by an unyielding demand for computers in their jobs. Technology has been given directly to them and has created "direct-users" from yesterday's end-users.

SUMMARY

In summary, the fourth generation impact on both report-writer products and end or direct-users has complementarily metamorphosized each. These changes have created a business environment where implementing report-writers for users must be reevaluated as a cost-effective means of decreasing the data processing department's backlog, and a way for users to perform their functions more accurately.

IV. IMPLEMENTING REPORT-WRITERS

"THE MAN WHO REALLY WANTS TO DO SOMETHING FINDS A WAY, THE OTHERS FIND AN EXCUSE."

Even in today's progressive environment, successful implementation of a report-writer for end-user use does not just happen. Careful planning and preparation must precede the effort or the chances for failure still remain high.

This section of this paper will detail some of the planning and preparations that preceded successful attempts at implementing a report-writer for end-user use in the recent past.

A. Selection of Report Writer

The selection of what product will be used is a critical step. Both user's needs and technical abilities must be carefully analyzed and matched to the product. Products with a greater ratio of success are:

- On-line and menu driven
- Provide on-line documentation
- Can be learned progressively
- Procedure not logic oriented.

B. Training

Training may be THE single most important key to success.

- It is advised that the first training session be an overview session. Discuss data dictionaries, databases, and include a small realistic demo.
- Target your training materials to your audience; both their technical abilities and their needs.
- Train progressively. Schedule four, 1 or 2 hour sessions over a course of 2 to 4 weeks. This gives the user opportunities to "practice" with the product between classes. They can then ask "hands on questions".
- Provide the user with the least amount of technical knowledge necessary to utilize the product. Trying to incorporate too many database technicalities, etc. is not advisable and only leads to confusion.
- Train "like users" in "like-groups"

- Create in-house manuals to be given to users as they train. Provide them with only the section of this manual that is being taught. (Access to too much knowledge too soon will confuse an individual until it can be put in proper perspective).

C. Pilot Application

Choose a single, simplistic application as a pilot for end-user/report-writer integration. The pilot can then be reevaluated and adapted accordingly when the effort is repeated in the future.

D. Demonstrate New Report-Writers

Provide an "end-user" demo to acquaint users and management with the 4GT report-writer products. Discuss their metamorphosis from report-writers that might have been previously used (or attempted).

E. User/DP Liaison

Designate a user to be a user/data processing liaison for each application. This individual will be in charge of the data dictionary maintenance, additional training and support.

F. Standards

Setting standards and guidelines is highly recommended. They should be addressed for the following issues:

- File Retention
- Historical Reporting
- Statistical Reporting
- What should not be done by a user
- Security

V. CONCLUSION

"IT REQUIRES A WISER MAN TO TAKE ADVICE THAN TO GIVE IT."

The concept of end-users utilizing report-writers is an excellent one. However, problems resulted when this concept and products relevant to it were marketed prematurely. Fortunately, fourth generation technologies have metamorphosized early report-writers into products that can now be used productively by today's "direct-users." These products have been proven cost-effective and should be reevaluated for use in today's business environment. Careful planning and preparation, progressive training and set standards create a perfect foundation for successfully implementing a report-writer for end-user use.

When successfully done, even a blind man can see -

THE BENEFITS CAN BE ASTOUNDING!

77 or Bust!

Some Hints on Converting from FORTRAN 66 to FORTRAN 77

by Margie J. DeCastilhos

ASK Computer Systems, Inc.
730 Distel Drive
Los Altos, CA 94022

77 or Bust!
Some Hints on Converting from FORTRAN 66 to FORTRAN 77

Margie J. DeCastilhos
ASK Computer Systems, Inc.
730 Distel Drive
Los Altos, CA 94022

This paper is intended to help users convert their FORTRAN/3000 (FORTRAN 66) programs to FORTRAN 77/3000. It focuses on changes that need to be made manually after using the conversion utility provided by HP. The initial discussion will introduce FORTRAN 77/3000 and the conversion program.

A Little Background

FORTRAN 77/3000 is Hewlett-Packard's newest FORTRAN compiler. It meets FORTRAN standards proposed in 1977. The FORTRAN 77/3000 compiler is a superset of the ANSI standard. Therefore, it includes everything that ANSI standards require with additional features provided by Hewlett-Packard.

FORTRAN 77/3000 is similar to FORTRAN/3000 although the '77 version is more structured. Both compilers can be run on the same system but not mixed within one object file. Mixing code can result in either an invalid type, a bounds violation, an incorrect number of parameters or possibly other errors at run time.

A Few Features

- The new compiler will eliminate symbol table overflows.

- FORTRAN allows access to run-time parameters, as shown in the following example:

```
:RUN PROG;PARM=1;INFO="SY"
```

  - PARM and INFO (shown in this example) are optional parameters. which are part of the operating system. They are associated with the MPE :RUN command. The PARM field is a 16-bit signed integer. The INFO field is a character string of up to 255 characters (which includes double or single quotation delimiters).

- In FORTRAN 77/3000, programmers specify appropriate parameters in the PROGRAM statement to obtain the values of PARM or INFO. These parameters can specify a variable for PARM, a variable for INFO, or both.

- After using variable names in the PROGRAM statement for this purpose, programmers declare the variables to be of the correct types. Variables must be local variables in the main program unit.

  a) The variable for PARM must be of the INTEGER*2 type.

  b) The variable for INFO must be of the CHARACTER*(*) type. The variable assumes the length of the string included in the :RUN command.

- Example:

```
$CONTROL USLINIT, LIST OFF
      PROGRAM TEST (PCKG, TERM)

      CHARACTER PCKG*(*)
      INTEGER*2 TERM

      WRITE (6,*) "TERM = ", TERM
      WRITE (6,*) "PCKG = ", PCKG

      STOP
      END
```

- Running the code (shown above) with:

```
:RUN TEST;LIB=P;INFO="SY";PARM=3
```

Results in:

```
TERM =  3
PCKG =  SY
```

After these declarations, any further reference to the variable, TERM, will have the value of 3. This is found in the statement associated with the :RUN command. Similarly, any references to PCKG will have the value "SY."

- $INCLUDE files can be used to insert duplicate code in multiple files. Use of these files ensures that all source has the same code. An example of this new syntax is shown below:

```
$INCLUDE "COMCOM.INCLUDE"
```

- The new IMPLICIT NONE statement forces the declaration of all variables. It must be the only IMPLICIT statement in the program unit. Types of INTRINSIC functions are not affected by the IMPLICIT NONE statement. IMPLICIT statements must precede all other specification statements (except PARAMETER statements) of a program unit. This statement is recommended for general use in structured programming.

- Comments can be added to the middle of a line by typing an exclamation point (!) before the comments.

  Example:

```
DO I=1,10       ! Loop to compute sum
```

  Comment lines can also be denoted by "C", "*", or "!" in column 1, or by a blank line in a source file.

- FORTRAN 77/3000 supports OPEN and CLOSE statements, which give more control over access to files and file characteristics than older versions of FORTRAN did. This also makes it easier to port FORTRAN code to other computer systems. For example:

```
OPEN(UNIT=19)
WRITE (19,*) "This writes to _____."
CLOSE (UNIT=19)
```

- There are now four ways (instead of two ways) to implement a DO loop. You can use:

  - **Labeled DO Loops:** These are the DO loops with which you are probably the most familiar. They begin with a DO statement, followed by a label, and then a control loop expression. For example:

```
DO 10 I=1,100
    .
    .
    .
10 CONTINUE
```

- **Block DO Loops:** These differ from labeled DO loops because a label is not used in its DO statement. Each block DO loop must be terminated with an END DO statement. For example:

```
DO I=1,100
   .
   .
   .
END DO
```

- **Implied DO Loops:** These loops are found in the input/output statements READ, WRITE, and PRINT and in DATA statements. Implied DO loops contain a list of data elements to be read, written, or initialized, and a set of indexing parameters. For example:

```
WRITE (6,*) ((A(I,J),I=1,2),J=1,3)
```

This will print values for the array in this order:

```
A(1,1)  A(2,1)  A(1,2)  A(2,2)  A(1,3)  A(2,3)
```

- **DO WHILE Loops:** These loops cause a group of statements to be executed repeatedly while a logical expression is true. For example:

```
DO WHILE (I .NE. 999)
   READ (5,33) I
   .
   .
   .
END DO
```

This example will READ until it encounters 999 as the input value.

NOTE

The FORTRAN-77 compiler directive, $ONETRIP, forces at least one execution of a DO loop (but not DO WHILE loops) in compliance with the FORTRAN/3000 standard. This option is turned off as a default in FORTRAN-77.

o The IF statement has been enhanced in FORTRAN 77/3000 to include the IF-THEN-ELSE construct. This enhancement is a step toward achieving structured programming practices. For example:

```
IF (CC .EQ. -32767) THEN
   CALL DBGET(BASE....)
   IF(NGD(45)) GOTO 999
   X=-B+((B**2-4*A*C)**(1/2))/2*A
   WRITE (6,*)X
ELSE
   X=(A**2 + B**2)**(1/2)
   WRITE (6,*) X
ENDIF
```

Conversion Process in Brief

HP provides a conversion utility and a manual on the steps needed to go through this process. This utility will make the majority of obvious changes which need to be made. The manual will also explain the changes that it will make for you as well as what it cannot do. Then it will take the time to show you how to modify the commands file to add your own special conversions. Therefore the rest of this paper will be spent explaining all the little intricacies that were not mentioned in the conversion manual that were run into while converting approximately 2,000,000 lines of code.

Manual Changes to Look Out For

o The following compiler directives are obsolete in FORTRAN 77/3000:

WARN	EDIT
NOWARN	ERRORS
CROSSREF	TRACE LABEL
FIXED	STAT
NOLABEL	SOURCE
NOSTAT	NOSOURCE
INTEGER*4	

- To avoid compiler errors when you compile your code, you must delete these compiler directives from your code.

- Figure 1 is an example of code that includes obsolete compiler directives.

```
    1       C>>>>OL CROSSREF
    0       2         $CONTROL STANDARD_LEVEL SYSTEM, SHORT, CROSSREF

**** WARNING # 1   THIS COMPILER OPTION NOT AVAILABLE ON THIS
 OPERATING SYSTEM: OPTION IGNORED (711)
    1       3               PROGRAM CONVERT1
    1       4         C
    2       5               SYSTEM INTRINSIC CLOCK, FMTCLOCK
    3       6               INTEGER*4 TIME
    4       7               CHARACTER*8 TIMSTR
    4       8         C
    4       9         C>>>TIME = CLOCK
    5      10               TIME = CLOCK()
    6      11               CALL FMTCLOCK (TIME, TIMSTR)
    6      12         C
    6      13         C>>>DISPLAY 'TIME = '.TIMSTR
    7      14               PRINT *, 'TIME = '.TIMSTR
    7      15         C
    8      16               STOP
    9      17               END

   NUMBER OF ERRORS =     0    NUMBER OF WARNINGS =   1
   PROCESSOR TIME      0:0:0    ELAPSED TIME          0:0:8
   NUMBER OF LINES =      17

END OF PROGRAM
```

Figure 1
Example with Obsolete Compiler Directives
in Source Code and Corresponding Warning

- The default size of INTEGER's and LOGICAL's is now 32 bits or 2 words whereas in FORTRAN 66 it was only 1 word. To be compatible with all current code, the compiler directive $SHORT can be used to continue to default variables to the short form. This is important for constants, implicitly defined variables, and variables declared as just INTEGER or LOGICAL without the *2 or *4.

- Any programs that use HP specific routines (like FOPEN) will need at the top of the source file $STANDARD_LEVEL SYSTEM. This and the previous situation are good reasons for modifying the command file provided by HP.

- The FILE= compiler option will no longer open files for access. A FORTRAN I/O statement will need to be used to get that particular file open before reading from or writing to that file.

- In FORTRAN/3000, the type of a named constant (in the PARAMETER statement) is determined by the literal itself -- not by the initial letter of its name. However, in FORTRAN 77/3000, the type of a named constant in the PARAMETER statement is determined by the initial letter of the name. Consequently, a variable must be explicitly declared before used in a PARAMETER statement.

  Examples:

  The PARAMETER Statement in FORTRAN/3000

  ```
  PARAMETER PRODUCT = "SY"
  ```

  - Because the constant ("SY") is enclosed by quotation marks. FORTRAN/3000 made "PRODUCT" a character variable.

  The PARAMETER Statement in FORTRAN 77/3000

  ```
  CHARACTER*2 PRODUCT
  PARAMETER (PRODUCT = 'SY')
  ```

  - In FORTRAN 77/3000, "PRODUCT" must be declared as a CHARACTER before it is used in a PARAMETER statement. (Alternatively, the defaults of I-N can be used to declare INTEGER's implicitly. Everything else will be assumed to be REAL.)

- DATA statements that are used to initialize arrays must initialize all elements of the array in FORTRAN 77/3000. The following examples will show this:

  DATA Statements in FORTRAN/3000

```
INTEGER BUF(7)
DATA BUF /4,12,13,14,15/
```

  DATA Statements in FORTRAN 77/3000

```
INTEGER BUF(7)
DATA BUF /4,12,13,14,15,0,0/
```

- Composite numbers must be calculated and written as octal numbers as this feature cannot be used in FORTRAN 77/3000. For example:

  A DATA Statement in FORTRAN/3000

```
DATA KSAMDEF /%|4/1,12/16|/
```

  The Same Statement (Octal Number) in FORTRAN 77/3000

```
DATA KSAMDEF/10020B/
```

- EQUIVALENCE statements have become more stringent. A BOUNDS VIOLATION or other run-time errors may occur if there are EQUIVALENCE's for a single variable in more than one place. To avoid this situation, the EQUIVALENCE statements should be consolidated in the code. The following examples compare this:

  EQUIVALENCE Statement in FORTRAN/3000

```
EQUIVALENCE (A,B),(A,C)
```

  EQUIVALENCE Statement in FORTRAN 77/3000

```
EQUIVALENCE (A,B,C)
```

- All SYSTEM INTRINSICS should be declared. If one is not declared, it may cause BOUNDS VIOLATIONS or other problems when the code is run. This problem may not occur for all intrinsics. A classic example of an undeclared SYSTEM INTRINSIC (DATELINE) that always creates problems is shown in Figure 2. However, when DATELINE is declared (Figure 3), the problem is removed.

```
:FTNGO DATES
ftneqns DATES, $STDLIST,$OLDPASS
run ftn.pub.sys;parm=7;nocb

 PAGE   1  HEWLETT-PACKARD   HP32116A.00.07
   HP FORTRAN 77  (C) HEWLETT-PACKARD CO. 1986   MON, NOV 10, 1986,  2:00 PM


     0       1.000        $CONTROL STANDARD_LEVEL SYSTEM, SHORT, USLINIT
     1       2.000              PROGRAM DATES
     1       3.000
     2       4.000              CHARACTER CTODAY*28
     2       5.000
     3       6.000              CALL DATELINE(CTODAY)
     4       7.000              WRITE (6,*) 'CTODAY = '.CTODAY
     5       8.000              STOP
     6       9.000              END

    NUMBER OF ERRORS =     0    NUMBER OF WARNINGS =   0
    PROCESSOR TIME      0:0:0    ELAPSED TIME          0:0:5
    NUMBER OF LINES = 9


END OF PROGRAM

END OF PREPARE


ABORT :$OLDPASS.SOURCE.FTNCLASS.?.?:SYSL.%3.%7452
PROGRAM ERROR #24 :BOUNDS VIOLATION

RUN $OLDPASS
PROGRAM TERMINATED IN AN ERROR STATE.  (CIERR 976)
```

Figure 2
Undeclared System Intrinsic
and a Bounds Violation

```
:FTNGO DATESYS
ftneqns DATESYS, $STDLIST,$OLDPASS
run ftn.pub.sys:parm=7:nocb

 PAGE   1  HEWLETT-PACKARD   HP32116A.00.07
   HP FORTRAN 77  (C) HEWLETT-PACKARD CO. 1986   MON, NOV 10, 1986,  2:02 PM


     0       1.000        $CONTROL STANDARD_LEVEL SYSTEM, SHORT, USLINIT
     1       2.000              PROGRAM DATES
     1       3.000
     2       4.000              CHARACTER CTODAY*28
     3       4.100              SYSTEM INTRINSIC DATELINE
     3       5.000
     4       6.000              CALL DATELINE(CTODAY)
     5       7.000              WRITE (6,*) 'CTODAY = ',CTODAY
     6       8.000              STOP
     7       9.000              END

    NUMBER OF ERRORS =     0    NUMBER OF WARNINGS =  0
    PROCESSOR TIME   0: 0: 0      ELAPSED TIME        0: 0:   8
    NUMBER OF LINES =      10

End Run
Begin Prep
End Prep


CTODAY = MON, NOV 10, 1986,  2:02 PM
End Run
```

Figure 3
Declared System Intrinsic,
Successful Run

o Any character string arrays in code must be changed to represent the new, FORTRAN 77/3000, syntax. Examples are shown below:

Character String Array in FORTRAN/3000

```
CHARACTER CARRAY*64(26)
```

Character String Array in FORTRAN 77/3000

```
CHARACTER CARRAY(26)*64
```

o If a character constant will not fit onto one line, it must be broken into 2 lines, if possible. It cannot be set up using continuation lines in FORTRAN 77/3000 to serve this purpose.

Character Constant in FORTRAN/3000

```
CBLNKS="
*              "
```

Character Constants in FORTRAN 77/3000

```
CBLNKS(1:40)="                                    "
CBLNKS(41:72)="                                   "
```

o If you access character strings in subsets of one byte, the conversion program will not convert them. However, FORTRAN 77/3000 will not accept this (unconverted) character substring. Rather it expects a character string that is subscripted with the first byte and the last byte.

- If you do not correct this problem before you compile your code, the compiler will flag it as an error. An example is shown below:

In FORTRAN/3000

```
STRING[4]
```

(This string would require a manual change after it was run against the conversion program. You would make it look like the code shown below.)

In FORTRAN 77/3000

```
STRING(4:4)
```

- If you must define a string that is being passed to you through parameters, and you are uncertain of its size, it is no longer necessary (and is actually incorrect) to pass information on the size separately. Compare the following examples:

  In FORTRAN/3000

  ```
  SUBROUTINE A (STRING,SIZE)
  INTEGER SIZE
  CHARACTER STRING*(SIZE)
  ```

  In FORTRAN 77/3000

  ```
  SUBROUTINE A (STRING)
  CHARACTER STRING *(*)  (Indicates that this string will be
                          the same size as the passed string.)
  ```

- Code that extracts bits (not bytes) from integers must now use the IBITS intrinsic function from FORTRAN 77/3000. For example:

  Bit Extraction before the Conversion Program Is Run

  ```
  INTEGER TFOPT,IFOPT

  TFOPT=IFOPT|14:2|
  ```

  Bit Extraction after the Conversion Program is Run

  (The conversion assumes that IFOPT is a CHARACTER string.)

  ```
  TFOPT=IFOPT (14:14+2-1)
  ```

  A manual change is required:

  ```
  TFOPT=IBITS (IFOPT,0,2)
  ```

  (Where "0" represents the starting (right-most) BIT in FORTRAN 77/3000, and "2" represents its length, from right to left. The values "0" and "2" are explained below.)

o In FORTRAN/3000 the 16 bits of an INTEGER are extracted from left to right (numbered from 0 to 15). In FORTRAN 77/3000, the 16 bits of an INTEGER are extracted from **right to left** (numbered from 0 to 15). Refer to Table 1.

Table 1
Bit Numbering

FORTRAN/3000	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FORTRAN 77	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- [14:2] (specified in the above example) indicates that bits will be extracted from the 14th bit to a length of two places to the **right**, in FORTRAN/3000.

- However, in FORTRAN 77/3000, IBITS requests the right-most bit as the starting bit and the length is measured to the **left**. Therefore, to extract the same bits in FORTRAN 77/3000 that would be extracted in FORTRAN/3000, you must indicate that bit 0 will be the starting bit, with a length of 2 to the left.

o In FORTRAN 77/3000, you cannot manipulate bits on a LOGICAL variable. Instead, you must EQUIVALENCE the bit to an INTEGER then perform the IBITS function with the INTEGER as the parameter. For example:

In FORTRAN/3000

```
LOGICAL MINE, YOURS

MINE=YOURS [3:4]
```

In FORTRAN 77/3000

```
LOGICAL YOURS
INTEGER MINE, OURS
EQUIVALENCE (YOURS,OURS)

MINE=IBITS(OURS,9,4)
```

- When you test a LOGICAL variable as true or false, FORTRAN 77/3000 looks at the right-most bit of the left-most byte. FORTRAN/3000, however, always looks at right-most bit. For example:

  0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

  now represents TRUE, in FORTRAN 77/3000. This may create a problem if you EQUIVALENCE an INTEGER to a LOGICAL, assign certain values to the INTEGER, then test the LOGICAL variable for true or false. Look for this situation in your code.

- You can no longer assign an octal value to a LOGICAL variable. Although DATA statements used to work, you must now EQUIVALENCE the INTEGER to the LOGICAL variable and use the DATA statement to assign the value to the INTEGER. For example:

  In FORTRAN/3000

```
LOGICAL IFLAG
DATA FLAG /%736L/
```

  In FORTRAN 77/3000

```
INTEGER IFLAG
LOGICAL FLAG
EQUIVALENCE (IFLAG.FLAG)
DATA IFLAG/736B/
```

  NOTE

  Although the correct bit pattern should be in FLAG, this is not a safe place to store this value; storing the bit pattern in a LOGICAL variable is not recommended.

- Because the system moves the left-most byte (only) when assigning logical variables:

  - You can no longer assign one logical variable to another and expect the resulting logical to have the same bit pattern as the original variable.

  - Refer to the examples on the next two pages.

```
:FORTGO TESTS

PAGE 0001  HP32102B.01.11  (C) HEWLETT-PACKARD CO.  1983

00001000            PROGRAM TEST
00002000            LOGICAL AOPT, AOPT2
00003000            INTEGER AOP, AOP2
00004000            EQUIVALENCE (AOPT, AOP), (AOPT2, AOP2)
00005000      C
00006000            AOP = %2345
00007000            AOPT2 = AOPT
00008000            WRITE (6,*) 'AOP2 = ', AOP2
00009000            WRITE (6,*) 'AOP = ', AOP
00010000            WRITE (6,*) 'AOPT = ', AOPT
00011000            WRITE (6,*) 'AOPT2 = ', AOPT2
00012000            STOP
00013000            END


PROGRAM UNIT TEST COMPILED


****          GLOBAL STATISTICS          ****
****     NO ERRORS,  NO WARNINGS     ****
TOTAL COMPILATION TIME    0:00:01
TOTAL ELAPSED TIME        0:00:01
TOTAL SYMBOL TABLE USED    % 06

END OF COMPILE

END OF PREPARE


AOP2 =     1253
AOP =      1253
AOPT =     1253
AOPT2 =    1253

END OF PROGRAM
```

Figure 4
Assigning Logical Variables
(FORTRAN/3000)

```
:FTNGO LOGI77S2
ftneqns LOGI77S2,$stdlist,$OLDPASS
run ftn.pub.sys;parm=7;nocb

PAGE  1  HEWLETT-PACKARD  HP32116A.00.07
    HP FORTRAN 77  (C) HEWLETT-PACKARD CO. 1986

PROGRAM LOADED WITH LIB = S  (LOAD WARN 89)
     1      1                    PROGRAM TEST
     2      2                    LOGICAL AOPT, AOPT2
     3      3                    INTEGER AOP, AOP2
     4      4                    EQUIVALENCE (AOPT, AOP), (AOPT2, AOP2)
     4      5             C
     4      6             C>>>AOP = %2345
     5      7                    AOP = 2345B
     6      8                    AOPT2 = AOPT
     7      9                    WRITE (6,*) 'AOP2 = ', AOP2
     8     10                    WRITE (6,*) 'AOP = ', AOP
     9     11                    WRITE (6,*) 'AOPT = ', AOPT
    10     12                    WRITE (6,*) 'AOPT2 = ', AOPT2
    11     13                    STOP
    12     14                    END

   NUMBER OF ERRORS =    0    NUMBER OF WARNINGS =  0
   PROCESSOR TIME   0: 0:   0    ELAPSED TIME       0: 0:   8
   NUMBER OF LINES =     14

End Run
Begin Prep
End Prep

AOP2 =    65540
AOP =     1253
AOPT =     F
AOPT2 =    F
End Run
```

Figure 5
Assigning Logical Variables
(FORTRAN 77/3000)

o Mixed mode expressions are evaluated differently in FORTRAN/3000 and FORTRAN 77/3000. In operations of the same precedence, FORTRAN/3000 evaluates the same types within an expression first. However, FORTRAN 77/3000 evaluates expressions from left to right. The following code (from HP's conversion guide to FORTRAN 77/3000) will produce different results in the two compilers:

Example:

```
PROGRAM ADDNUM
INTEGER*4 j
j=2000000000
WRITE(6,*)1.0+j-j
STOP
END
```

- The result returned in FORTRAN/3000 will be 1.0, while the result returned in FORTRAN 77/3000 will be 0.0.

- This is so because "j" must be converted to a real number, then added to 1.0; therefore the significance of 1 is lost in the storage of the real number.

- You may use parentheses to control the order of evaluation. If, for example, you want FORTRAN 77/3000 to give the answer "1.0" to the example on this page, you should change the WRITE statement as shown below:

```
WRITE(6,*)1.0+(j-j)
```

o All FORMAT statements that use %103C for carriage control in FORTRAN/3000 will now need to use 1HC for the same carriage control in FORTRAN 77/3000. For example:

Before (FORTRAN/3000)

```
    WRITE (6,100)...
100 FORMAT (%103C,...
```

After (FORTRAN 77/3000)

```
WRITE (6,100)...
100 FORMAT (1HC,...
```

- Along the same line, the FORMAT statement containing the %320C will now have to use a predefined descriptor, NN, in FORTRAN 77/3000. If, however, you would like to print more than one line within the same format statement (if there is a / in the FORMAT statement) you must also include an NL descriptor behind the / to start the next line. For example:

  In FORTRAN/3000

```
      WRITE (6,100)
100   FORMAT (%320C,"HEADING NUMBER ONE",/,
     *"HEADING NUMBER TWO")
```

  In FORTRAN 77/3000

```
      WRITE (6,100)
100   FORMAT (NN,"HEADING NUMBER ONE",/,NL,
     *"HEADING NUMBER TWO")
```

- 2/ and 3/ are not longer valid descriptors for FORMAT statements; you should use // or /// instead.

- Similarly, 60"-" is not longer a valid format descriptor. 60"-" must be changed to 60("-").

- Examples:

  In FORTRAN/3000

```
      WRITE (6,10)
10    FORMAT (" FIRST LINE", 2/, 10"-")
```

  In FORTRAN 77/3000

```
      WRITE (6,10)
10    FORMAT (" FIRST LINE", //, 10("-"))
```

- The free format descriptor "S" is not supported in FORTRAN 77/3000. Although you may use the "A" format, it is not an exact substitute for "S".

- When you test the equality of a real number, you will see the following compiler warning:

  TEST MAY FAIL DUE TO FLOATING POINT IMPRECISION

  - You can avoid this warning by changing your FORTRAN/3000 code as shown below:

  FORTRAN/3000 Code

```
REAL PI,NUM
PI=3.14
NUM=PI-3.14
IF (NUM.EQ.0.0) WRITE (6,*)"OK"
```

  FORTRAN 77/3000 Code with Changes

```
REAL PI,NUM
PI=3.14
NUM=PI-3.14
IF (ABS(NUM).LT.0.005) WRITE (6,*)"OK"
```

  - The computer sometimes creates inaccuracies when it rounds numbers. These changes establish a tolerance level that compensates for the inaccuracies.

- The condition code for testing conditions after INTRINSIC calls has been changed from .CC. (in FORTRAN/3000) to CCODE() (in FORTRAN 77/3000, a function).

- Subprograms Without Parameters: The syntax for calling subprograms for which there are no parameters differs between FORTRAN/3000 and FORTRAN 77/3000.

  - For example, in FORTRAN/3000:

```
time=clock
```

  but in FORTRAN 77/3000, an empty pair of parentheses is required:

```
time=clock()
```

  - The INTRINSIC FUNCTIONS that you may use are:

    CLOCK
    CALENDAR
    GETJCW
    TIMER
    PROCTIME

- There were many changes made to the various FORTRAN functions in FORTRAN 77/3000. Some functions were deleted, others were replaced with new functions, and there are entirely new functions that are new features. Please refer to the next three tables for a layout of these changes.

Table 2
Deleted Functions

FORTRAN/3000 Functions	Solutions for FORTRAN 77
INUM, JNUM RNUM, DNUM	These functions can be replaced with the subroutine EXTIN (refer to HP's Compiler Library Manual), the INTRINSIC BINARY or FORTRAN's in-core conversion.
STR	Use the INEXT subroutine (refer to the HP's Compiler Library Manual), the INTRINSIC ASCII, or FORTRAN's in-core conversion.
INT	You can no longer use this function to truncate LOGICALS to INTEGERS in FORTRAN 77/3000.
BOOL	You can no longer assign an INTEGER to a LOGICAL in FORTRAN 77/3000.
IJINT	You can assign INTEGER*4 to INTEGER*2 directly. However, numbers greater than 32767 or less than -32767 will be **converted to a negative number:** they will not "blow up" as they did in FORTRAN/3000. This is caused by a bug in the version A.00.07a of the compiler.
JIINT	This function no longer exists, but you can assign a REAL to an INTEGER*4 directly.
DBLE	This function no longer exists, but you can assign a REAL to a DOUBLE PRECISION directly.

Table 3
Names of New Functions

FORTRAN-77 Function	Definition	Type of Argument	Type of Function
ZABS	|a|	COMPLEX*16	DOUBLE
DFLOAT	Converts to double precision	INTEGER	DOUBLE
DCMPLX	Converts to COMPLEX*16	REAL, INTEGER DOUBLE, COMPLEX	COMPLEX*16
DDIM	Positive difference	DOUBLE	DOUBLE
DIMAG	Imaginary part complex argument	COMPLEX*16	DOUBLE
DCONJG	Conjugation of a complex argument	COMPLEX*16	COMPLEX*16
DPROD	A*B	REAL	DOUBLE
LEN	Length of character entry	CHARACTER	INTEGER
ANINT	Nearest whole number	REAL	REAL
DNINT	Nearest whole number	DOUBLE	DOUBLE
NINT	Nearest integer	REAL	INTEGER
IDINT	Nearest integer	DOUBLE	INTEGER

Table 4
Functions Requiring Manual Changes

FORTRAN/3000 Function	FORTRAN 77/3000 Function
AJMAX0	AMAX0
JMAX0	MAX0
JMAX1	MAX1
AJMIN0	AMIN0
JMIN0	MIN0
JMIN1	MIN1
JFIX	IFIX
ISIGN	HSIGN
JSIGN	ISIGN
IDIM	HDIM
JDIM	IDIM
IABS	HABS
JABS	IABS
IAND	HIAND
JINT	INT
JDINT	IDINT
FLOATJ	FLOAT
MOD	HMOD
JMOD	MOD

- When calling procedures that pass INTEGER and LOGICAL parameters by reference, make sure the actual and formal parameters have the same word length. If parameters passed by reference are not the same, the segmenter issues an error message. When parameters that do not have the same value are passed by value, the compiler converts single integers to double integers for you.

- All subprogram calls to the same subprogram within the same program unit must pass the same number and types of parameters. If this is not possible, you should use the $CHECK_ACTUAL_PARM compiler directive. (See Figure 8.)

```
1   EXTINS.SOURCE.FTNCLASS

    1       $CONTROL STANDARD_LEVEL SYSTEM, SHORT, USLINIT
    2             PROGRAM EXTINC
    3
    4             LOGICAL LOGFALSE
    5             PARAMETER (LOGFALSE = .FALSE.)
    6
    7             CHARACTER TIME_STRING*4, PRICE_STRING*6
    8             INTEGER*2 TIME
    9             REAL PRICE
   10
   11             TIME_STRING = '1200'
   12             PRICE_STRING = '512.95'
   13
   14       $CONTROL CHECK_ACTUAL_PARM 2
   15             CALL EXTIN (TIME_STRING, 4, 0, 0, 0, LOGFALSE, TIME, IERR)
   16             CALL EXTIN (PRICE_STRING, 6, 2, 1, 0, LOGFALSE, PRICE, IERR)
   17       $CONTROL CHECK_ACTUAL_PARM 3
   18
   19             WRITE (6,*) 'TIME = ',TIME
   20             WRITE (6,*) 'PRICE = ', PRICE
   21
   22             STOP
   23             END
```

Figure 8
Example of Procedures that Pass
INTEGER and LOGICAL Parameters by Reference

- Code that jumps to the middle of a DO LOOP is not supported by FORTRAN 77/3000.

  - You must change the logic of the code to compensate for this situation. FORTRAN 77/3000's IF-THEN-ELSE construct is recommended.

Conclusion

As pointed out in these examples, there could be many situations that a programmer may run into that will need special attention even after using the utility provided by HP. While the conversion process may seem difficult, programmers will come to appreciate the new standards and features of FORTRAN 77/3000.

RECOVERY BY DESIGN

James A Depp
UP TIME - Disaster Recovery Services
4131-A Power Inn Road, Sacramento, California 95826

The Need

The need is real - data centers are destroyed. Disaster prevention can be and is practiced every day to some extent. For example, taking data offsite is acknowledging that accidents can happen, and is done to provide recovery options. Disaster recovery planning must be included among the business activities which are on-going, and which ensure continuation of business. This presentation provides an overview of the need for recovery planning, the hazards against which we protect ourselves, and goes further, into the 'how to' aspect of recovery and contingency planning.

Natural disasters can occur in all parts of the country. On the East coast hurricanes are not unusual. The Midwest has tornados. Flooding is a common hazard in some regions or seasons. Earthquakes are the major concern on the West coast. In Minneapolis an HP user watched as the building across the street burned to the ground - a victim of children playing with matches.

Disasters - natural or otherwise - can devastate a company. It is impossible to predict the timing or location of a disaster - the worst earthquake in U.S. history occurred in Missouri, in the county which is today transected by multiple, major gas pipelines.

The effects of disaster can be wide spread and lasting. In Mexico City there are 107 HP data centers - of those, 6 were totally destroyed in the earthquake - more than 20 others were located by HP as needing help - some level of equipment damage requiring replacement. Telephone lines were out for weeks, particularly long distance service. Some buildings appeared intact, but tilted so much that they were unsafe. The effect is long lasting. By February, 1986, 6 months later, data processing was beginning to catch up. In November, 1986, some companies were still not fully current - and development work was a year behind schedule.

However, despite the concern for natural disaster, the most commonly documented disasters are manmade - through accident, negligence, or vandalism, or discontent.

The Hazards

Potential hazards include overheating, fire, power surge, lightning, flooding, earthquake, lack of security of programs and facility, accidents, vandalism, terrorism, bombs and threats, toxic liquids and gases, wind, etc.

The impact is significant, with only a few days reducing the organization's effectiveness dramatically. Without preplanning recovery may be long in coming - long as compared to the 2-5 days that a University of Minnesota study indicates is the critical survival period for most companies following a disaster.

Without plan or backup, restoration of the operation may take weeks. Following a Salt Lake City fire, and with the alternate facility identified, the system was delivered in 7 days, telephone lines in 12 days. It was still 4-6 weeks before operations were approaching normal, and 6 months to recreate all that had been changed and improved, but not fully backup up prior to the fire.

By contrast, in West Germany where recovery planning was in place and a mobile hotsite available, operations were restored in 36 hours. Delivery of the hotsite included the crossing of the English Channel by ferry. In Northern England, recovery was even more swift.

Preparation, Planning, and Recovery

Disaster Recovery Planning, taken in reverse, is the preparation - of the response - to an interruption to regular operations. The magnitude of the interruption determines the degree of response. It is important in the planning process to consider the viewpoints of and involve several groups, each interested in the business function from differing perspectives - auditing, management, data center operations, user support, and the user community. Stockholder, customer, director, vendor, and public viewpoints should also be considered.

The challenge is to achieve the appropriate balance - recovery planning should seek to contain the cost of a disaster at a level which the organization can afford. That cost is the cost of loss plus the cost of recovery. It must be acceptable.

Getting Started

A popular problem solving technique moves successively from specification to generalization and back. This approach works well for recovery planning, as the scope of normal operations is narrowed to critical needs. The needs are then defined broadly as the requirements, which are in turn narrowed as options/solutions are evaluated. Finally, having defined solutions, the procedures are written to support those solutions, at an optimum level of detail.

This process takes into account various scenarios - short vs. long outage - repair to full replacement of hardware - data center damage or full destruction - available staff levels - data integrity.

An effective method of establishing priorities among applications is to convene a group of diverse users and managers and role play the various disaster scenarios - identifying as a group the options and benefits. By progressively narrowing the resources available in a recovery, and expanding the assumed damage and injury level, the group is forced to wrestle with the issue of priority and benefit.

Additionally, a matrix of outage costs and probabilities for several scenarios can give a dollar indication of projected loss. Though its precision will be limited, such a figure is another form of option evaluation.

The objective is a suitable hardware environment for processing, and software and data to maintain services to users. This is not necessarily all software, all service, all users, or on the normal schedule. It is the planning process which defines what is necessary.

The selection among critical resources of which to protect or replace is extremely difficult but essential. Data, people, hardware, facility, communications, supplies, and a plan - all are necessary for successful recovery. The two which require considerable lead time are the hardware and facility, and are candidates for careful and early option evaluation.

Each of the options available have pros and cons. In general, the options separate into categories of manual operation, fully redundant system, agreements with others, hot sites, vendor facilities, service bureaus, and empty shell sites. Rather than dismiss any or several of these options, the pros and cons should be evaluated against the list of requirements, and considered in the context of the disaster life cycle.

Reciprocals have their value in small tasks and quick answers, but not in sustained operations. A printer problem can be overcome through a reciprocal site. Redundant sites provide processing immediately on a large scale, but are costly to set up and maintain. Hot sites provide many of the same benefits while the cost is lower since the resource is paid for by several subscribers. Empty shell sites are useful only after the machine is received, but continue as a solution until the rebuilt or new permanent site is complete.

With the critical applications identified, and the requirements defined, and the hardware and facility options selected, the planning process continues into the more detailed activities. As the strategy and documentation is described, recognize that much of the information is already in run books, available for copying, or, if not, the activity of documentation will significantly benefit the routine operations as well.

Strategy and Organization

Data and software are the easiest to provide and protect - ESTABLISH OFFSITE STORAGE and the procedures to support the movement of tapes to and from the storage site. Offsite is not a specific description, nor is the cost defined - quite simply, separate the data and software copy from the machine and facility. Take it home, store it at a cross-town office, or at a neighboring data center. If choosing another data center, be particularly sure the tapes are distinctively different from that site's tapes. Commercial services are available, and provide a routine, outside impetus for moving the data offsite. Be certain that when these tapes are onsite whether awaiting transport or being returned for use that the staff recognizes their importance, and treats them accordingly.

Identification, organization, and education of the PEOPLE to be involved in the recovery will be a key part of the planning process. In general, start by involving anyone and everyone who might be able to contribute. This will enlarge the available resources when the disaster occurs, and protect against failure through injury or loss of a key person.

The organization structure should fit with the corporate structure, and should identify a very few specific lead people. The Disaster Recovery Plan Coordinator would have overall responsibilities during the disaster. Specific technology lead people would be the System Recovery Manager and the Rebuilding Project Manager. The recovery organization chart should reflect that there are specific people working in functional recovery roles, with support disciplines such as purchasing, engineering, construction, legal, and accounting working with each functional group.

Roles and tasks should be described, and teammates identified by position or function, not by name since these may change. The plan should include checklists in chronological or importance sequence, and where evaluations are required, the parameters should be provided so that everyone understands the severity of each area impacted. Communication lines should be clearly defined, with press releases and early internal announcements thought out and written in advance, and included with the plan.

Where spending limitations and corporate policy decisions are involved, these must be established in the planning process. Upper management involvement and commitment is critical. By having prior signoff on key elements, spending, and policy they will be positioned to provide informed assistance. Other aspects of the corporate activity can be pursued with confidence that the information systems recovery is well managed.

Included in the plan should be the assumptions, economic and other reasons for planning and recovery. The defined requirements should be included. The facilities should be described and any special features annotated. The escalation

criteria should be included - when will this plan go into effect, and to what degree based upon the situations foreseen.

Preparation, Procedures, and Instructions

The facility should be organized and designed to minimize loss of equipment in a disaster. Standard components, easily replaced should be used, so that undamaged equipment can quickly mate with replacement equipment. Preventive measures should be taken to protect against excess heat, power fluctuation damage, fire, smoke, water damage, and breach of security. For recovery, include a current description or set of building plans, and the current and and up-to-date wish list of improvements.

The hardware should be standard, or documented. Any wiring for power or data communications should be documented, specifically any interfaces which are non-standard or which require connector jumpers. Where nonstandard board jumpers, straps, or settings are used, document them. Beware the antique, unique, or ultra-new since replacement is more difficult.

Standard system files should contain jobstreams to recreate accounting structure, system SL, release of files, and general housekeeping activities. These should be stored first or early on the full dump tapes. Software should reference hardware by class name, not device number, and if certain devices are unavailable, such as port printers, defaults should allow alternate actions, such as printing to the system printer at low priority.

Communications equipment should be located to maximize survival in a disaster. Water, heat, and power are the primary threats. Consideration should be given to redundant equipment in the field which could be brought back to the central site, such as two 16 port multiplexers rather than a single 32 port unit. Standardization increases the probability of recovery. Do review with the vendor their support stategy for disasters.

Supplies should be partially stored offsite, and rotated into the standard processing cycle. This can frequently be arranged with forms suppliers. Samples of all forms should be stored offsite with the recovery plan. For laser printed forms, have an alternate site available, or supplies of forms, or at least a blank laser generated form for each used.

Prior arrangements should be made for space, hardware replacement, supplies. Pre-executed purchase orders are common and a good practice. Copies of support contracts and equipment specifications should be kept with such purchase orders, and copies of all of these documents stored offsite with your plan.

Corporate Benefits - the Payoff

Aside from peace of mind - which should not be understated - the benefits of recovery planning both operationally and organizationally are real. Day to day operations are improved when reviewed - scrutiny leads to correction of problems and practices. Understanding of the data processing function, and its integral relationship to profitability and company survival, leads to smoother interactions day to day. On-going expenses may be impacted by design improvements which may result in system performance improvement, simplified operation techniques, and better trained staff.

Real dollar savings are possible through reduced insurance premiums, and improved response to minor disasters and interruptions.

From the audit and legal perspective, the company is better able to remain a viable enterprise. An area of significant risk exposure is confined. Legal reprisal is reduced in that prudent measures are taken to protect the information asset, and the information will be more readily available. The Chief Financial Officer is more secure since he is one who could be charged with criminal negligence for failure to safeguard against accidents. Conviction could result in individual and corporate fines.

Disaster recovery planning is a part of a program of protecting corporate assets. It joins safety and security programs in its importance. Planning can be approached in a stepwise fashion, and should involve people at all levels in its creation, review, and revision. The first steps, however small or simple, should be taken now, before the unpredictable and devastating disaster strikes.

PUSHING THE LIMITS OF SYSTEM MANAGEMENT
Surviving that "one step beyond"

Brian Di Silvestro
Hewlett-Packard Response Center
3300 Scott Blvd.
Santa Clara, CA. 95054

Sooner or later, all system managers reach some limit of the HP 3000. This may be a physical limitation, such as running out of CPU power, or it may be a limitation with MPE. In either case the limits of the HP 3000 can usually be bypassed safely. The user who is ready to push these limits needs to be aware though that the limits are not circumvented with out a corresponding decrease in stability or performance. The goal of this paper is to let you know what problems can result from pushing the HP 3000, and methods of keeping a stable system once the limits are crossed.

Most of the limits that a system manager can run into fall into two categories: physical limitations, and limitations with MPE. Of the physical limitations, the most common ones are the limits of the system configuration. For example, there is a limit of only two high speed GICs per an IMB. (GIC's and IMB's are responsible for the physical data transfers for most I/O devices). In most cases though, the configuration can be juggled around to accommodate this restriction, but there is a price in I/O

performance. Other physical limitations include limited CPU power and limited memory.

The other type of limitation is imposed by the MPE operating system. These includes security limitations, stack size limitations (which is actually a hardware limitation), file locking conventions, and process handling. In order to bypass all of these limitations, special capabilities are available for the user, and with proper use, they are very useful. The capabilities are restricted since they contain the potential of damaging system integrity, or monopolizing system resources. Of all the capabilities, privileged mode (PM) is by far the most dangerous. Armed with PM, a user can completely bypass MPE. But without a thorough understanding of the internals of MPE, a simple change by any user can drastically affect other users. In a worst case scenario, a privileged mode user may cause undetected data corruption. Most of the other special capabilities though are not nearly as dangerous, but an unknowledgeable user can still hang a system with either process handling (PH), extra data segment (DS) or multiple rin (MR) capabilities.

Once users start using special capabilities, problems can best be avoided by careful and knowledgeable use of the capabilities. If consideration is taken for resource limitations, such as the number of data segments

configured, then most systems will remain stable. There are cases, however, when performance will start to suffer as programmers use special capabilities. This is a limit that can not be bypassed. What can be done to ensure that the system will remain stable as the system grows? An example that will be covered is the use of the tune command to avoid a specific type of hang concerning resource usage. One point is worthwhile to keep in mind, though. Sites that do not change their machines remain stable. As an example, there are still a few sites that remain on old MITs, such as DELTA MIT, without any problems. As long as they see no need for change, they have no need for an upgrade.

As users start using various system resources, they will usually run into three types of physical limitations: CPU power, I/O limitations, and memory limitations. Of these, a CPU bound system can be the most drastic. Since the HP 3000 runs best when there are occasional periods of CPU inactivity, hangs can sometimes result when a CPU approaches 100% utilization. The hang covered here can be specifically addressed by use of the tune command. The scenario for such a hang is as follows: when a processes gets CPU time, it may acquire some type of system resource, such as an image buffer. If the process then loses the CPU while it was holding a lock on the resource,

it will retain the lock on that resource. In most cases this is acceptable, since the CPU will soon go idle, and the process will have a chance to release the resource. If the CPU is running at 100%, a process such as a job may not be able to acquire the CPU for some time. If some session then needs the IMAGE buffer, he will wait behind the job until the job has released it. Thus that session will hang. Sooner or later, every image user may hang behind the job. The good news is that as all the users impede, they will usually relieve enough pressure on the CPU, so that the original holder will be able to release his resource, thus releasing all the hung users. If on the other hand, there is another user that is CPU intensive, the hang will not be able to free itself up. As in the case of IMAGE, this type of hang can also occur if users are acquiring any type of shared resource. There are two methods of avoiding this hang. The first is to not have sessions and jobs competing for the same resource. Since sessions run at a higher priority, the job's will be less likely to acquire the CPU. The other method of avoiding this type of hang is to use the tune command to restructure MPE's queue structure.

In almost all cases the tune command is never needed. The default queue structure works very well for most systems that have some idle CPU time. Tune should only be used if

a change is clearly needed. Misuse will most certainly cause more problems than it addresses. In order to understand the tune command and it's dangers it is necessary to understand some basics about how processes are queued to run.

MPE always chooses to run the highest priority process that is available. To avoid monopoly of the cpu, MPE drops the processes priority as it uses CPU time. The tune command comes into play by specifying how high the priority can be raised to, how far it will drop, and additionally how fast it will drop. For our purpose of avoiding the hang, the tune command need only be be used to change the limits on how far a processes priority can drop.

If in our image example, a process has acquired the CPU in the C queue, but at the same time, a process held a resource in the D queue. Since the C queue has preference over the D queue the job will rarely ever run. By default, the C and D queues never overlap, but if the tune command is used to overlap the queues (drop the C queue down), the job would be able to acquire the CPU as the C queue process had their priority lowered below that of the job. The job would then be able to release the resource, thus avoiding the potential hang.

There is a lot of potential for misuse of the tune command. Since most sessions block for terminal io, they usually have the CPU for a very short time. On the other hand, jobs will usually get the CPU and rarely block to release it. Thus, if there are a lot of jobs, and if the queues are overlapped too far, the jobs may get a larger portion of the CPU, and drastically affect response time. As a general rule, the best advice, if possible, is to not have jobs and sessions competing for the same resources (such as one image data base). This by itself will increase response time, and avoid a potential hang. The tune command should only be used as a last resource.

The second physical limitation that affects most people is that of I/O. As was mentioned above, the first limitation covers what can actually be configured, and how much data the hardware can handle. Specifically, only two and a half GICs running at full speed can work properly on an IMB. Fortunately, most GICs do not run at full speed, or if they do, it is in short burst of activity. If a GIC has the potential to run a full speed, such as for a disc drive, it is classified as a high speed GIC. Thus, only two high speed GICs are allowed on an IMB (a LANIC is an exception). The way around this limitation is to have a GIC share a number of high speed devices. The drawback is that I/O performance will slow down. (This is a

preferable to risk of system failures and data corruption with too many high speed channels). When sharing a GIC with other high speed devices (tape drives and disc drives), some rules of thumb do apply. Since a GIC can only operate one device at a time, it is best to arrange the configuration around these devices. For example, putting a tape drive and a disc on the same GIC will tremendously slow down back up time. Another feature to keep in mind is RPS (Rotational Positional Sensing). Disc drives, such as the 7933/7937 were designed to use the GIC most efficiently when it shares the GIC with similar devices. Rps enables two disc drives to overlap I/O by releasing the GIC during rotation latency (waiting for the data to arrive under the disc head). This feature will not help if the other devices sharing the GIC do not take this into account. For example, RPS can actually hurt, and is not supported, if a 7933, or any RPS device, shares a GIC with a non cs80 device, such as a 7925.

The final physical limitation to be covered is memory. Memory is a rare resource on the HP 3000. In order to run, it is necessary to remove the nonessential data in memory off to disc in a reserved space called virtual memory. As the data is needed, it is brought back into memory. When memory space is tight memory management will start to acquire more CPU and I/O resources. If more

memory is not available, then the space that is available must be used more efficiently. The basic users of memory that we can control are cache and system resident tables. Code and data segments also use memory, but are more difficult to control.

As far as the HP 3000 is concerned, cache is just like any other portion of memory. If a cache domain is being used, then the system will keep it in memory. In most cases, this is what we want. If we are short on memory though, the cache domains will need to be posted to disc more often, and this will further increase our CPU overhead. Cache is only affective if and only if a system has spare CPU power and memory to handle cache management.

The other use of memory that we can control are system tables. Fortunately, in most cases the system only allocates the space that it actually needs, such as in the loader segment table for autoallocate. Some tables that are critical, such as the ICS (Interrupt Control Stack), are permanently allocated in memory at their full size. Thus, it is wise to never over configure system tables. In most cases though, it is not all that effective to keep a tight reign on table sizes. Most system tables, such as the ICS are very inexpensive in memory usage, and if most tables do overflow, system hangs, halts, or failures will result. For example, a system halt 4 will result if the

ICS overflows. The best advice is to watch the tables with OPT or TUNER, and if any fill up to over 80%, it may be worthwhile to increase them. Note: This recommendation depends on the table involved. If autoallocate is enabled, tables such as the cst block table will remain at 100% utilization. This is it's normal state, and will cause no problems.

At the other end of physical limitations, are the software limitations within MPE. With special capabilities, and/or priv mode, almost all the limitations can be overcome. One very strong word of warning is required. All the limitations and conventions of MPE were designed to avoid problems and to allow for the best performance in a multi user environment. Bypassing these limitations is sure to cause problems unless the user is completely aware of how the change will affect the rest of MPE. A good example of the dangers of special capabilities can be demonstrated with privileged mode. Since PM disables all bounds checking on data transfer, a program will be able to access memory outside of their stack. For example, if the program writes over a portion of a memory header (MPE uses a linked list of headers to manage memory regions), the system is likely to fail with a system failure. Most likely a sf614.

Even considering the dangers of special capabilities, they are still very useful. The most common ones are listed bellow:

DS capability: This allows a user to acquire extra data segments, and thus bypass MPE stack size limitations. This is a reasonably safe capability, but a user can use up system resources, such as the DST table (Data Segment Table) and virtual memory.

PH capability: This allows users to create son processes. This is also somewhat safe, but a user can use up system resources and tables, such as the PCB, DST and CST.

MR capability: this allows users to lock multiple resources. this can be very dangerous, and deadlocks and hangs can result. As in MPE, certain conventions must be used to specify what order resources must be locked.

PM capability: This is the most dangerous, but also the most powerful. with PM, any user can bypass security restrictions, and completely bypass MPE.

Privileged mode actually can be generalized down into two features: One, disabling bounds checking on data moves. And two, to permit the calling of uncallable procedures within MPE. The first danger to be aware of is that if a program is in privileged mode and experiences a bounds violation, due to a programming error, the error will not be trapped, and the program may overwrite some other

processes critical data. The second danger of PM is that it allows us to call uncallable, and privileged procedures. This is powerful since most of MPE is comprised of uncallable and privileged procedures, and with priv mode, these procedures are accessible. The danger arises since these procedures assume a complete knowledge of how they will work. For example the file system has privileged mode checks in some intrinsics, such as FOPEN. If incorrect parameters are passed to the intrinsic in privileged mode, the error will not be trapped, and a system failure will be likely. Note: As an example, the entry points allow image to use global FOPENs.

When pushing the limits of MPE, it is important to know what the limits are, and where they are. Hopefully, this paper has helped inform people about the issues and dangers involved. In most cases, the dangers of pushing the HP 3000 can be avoided. For example, system failures can be avoided from incorrect configurations, but performance may suffer as a result. There is one final limitation though concerning performance. Once the HP 3000 has exceeded performance limitations, there is little that can be done without extensive research into the background of the site. Limitations however can be recognized so that methods can be developed to avoid potential problems that may arise, such as resource hangs. When the limits that

are being pushed require the use of special capabilities, such as privileged mode code, special care must be taken to ensure that the changes do not adversely affect other users. Careless use of special capabilities will ultimately cause severe problems, but careful use will greatly enhance the power of your computer.

Management in a Fourth Generation Language Environment

By Patrick Dolan
Michigan Education Data Network Association
1350 Kendale Boulevard
East Lansing, Michigan 48826

517-349-7746

Fourth generation languages are contributing in important ways to our data processing lives today, despite their infancy. At the Michigan Education Data Network Association (MEDNA), real and promised benefits have led us to utilize a fourth generation language for all current and anticipated future application development. While there are challenges associated with such a total commitment, results have already justified this decision. This paper is about management in a fourth generation language environment -- specifically, a Synergist environment.

Synergist is a revolutionary, rich new transaction processing language which unites the HP 3000 with Personal Computers to maximize the storage and processing capabilities of both. It was developed by Gateway Systems of Okemos, Michigan. The Michigan Education Data Network Association (MEDNA) was an early purchaser of Synergist and has aggressively pursued application development using this software tool since mid 1985.

We are proud of the accomplishments that have been achieved. A seven person staff, working with Synergist during the past two years, has created systems for General Ledger, Accounts Payable, Financial Report Writer, Conference Registration and Scheduling, Medical Claims Payment, Membership Maintenance and Billing, Insurance Underwriting and Contributions, and Legal Case Maintenance and Billing in addition to Education Data Filing and Retrieval. A number of other projects are underway.

Because Synergist is so different from other languages, numerous opportunities are afforded management staff to explore new relationships with end users, develop in-service programs for staff, and provide innovative support systems. This paper examines some of these challenges.

Expectations

Mention of the term "fourth generation language" seems to suggest increased efficiency in development, rapid learning curves for programmers, decreased costs and better utilization of resources. Fourth Generation Languages, including Synergist, can do many things very well. Synergist's strengths are many, largely because of its rich language. Learning to become an expert "Synergizer" however, does take time.

Programmers expect to learn very rapidly, yet mastery of a rich language requires extensive time, effort and experience. A programmer can begin handling dictionary maintenance within a day. Screen painting is possible within the same amount of time. Within a week, given a knowledge of the HP 3000, a simple application can be developed and installed. Having this background, it will now take about six months to learn advanced techniques and to begin thinking as a "Synergizer". As experience grows and examples of programming techniques are made widely available, the ability to pass the language from person to person will no doubt become easier. Still, an extensive amount of "hands on" experience is necessary to begin understanding and appreciating what Synergist can do. Immediate results are available only on relatively simple applications. Opportunity for exploration is required if the programmer is to begin thinking as a "synergizer".

Patience by management is important. In an environment where programmers are working with early releases of a software product, "bugs" and "undocumented features" are common. Work-arounds are the programmer's everyday activity until the language is more fully understood. With experience, passing the language from person to person accelerates the learning curve and a major management task is the challenge to find ways for the user group to expedite dissemination of knowledge beyond those who are rapidly acquiring it through experience and exploration.

Support Systems

Development of new support systems helps programmers become even more efficient with Synergist. These encompass hardware and software systems as well as standards and procedures.

Two years of experience have led to the formulation of a number of programming standards which make development, modification and enhancement more efficient and reliable. These standards result primarily because:

- programming standards for traditional languages cannot always be easily adapted to Synergist.
- Synergist's unique architecture and rich language provide new options for the programmer to consider.
- providing explicit direction for Synergist handling of pathing and other "fifth generation oriented" features on complex forms usually reduces debugging time.

Since development is done primarily on the PC, attention to creating a hardware configuration which increases efficiency is a valuable investment. A system of "remote compiles" can permit the programmer to "send" forms to a PC dedicated to compiles. This enables the programmer to continue working on other projects while forms are being compiled. Through the addition of memory and ram disk to high speed PCs, as well as experimentation with various configuration options, time required to compile forms on a PC dedicated to the compiling task has been reduced to several minutes, even on large forms with 600 fields which take an hour on early vintage PCs.

Another way to increase efficiency is to provide programmers with two PCs. One is used to run the application in a test configuration while immediate changes are made to forms on the other.

Changes in the data base require re-compiling all the forms which are affected by those changes and then re-installing the forms. Batch files can be written to verify successful compiles and then to install the form. On large projects, even if mass re-compiles and re-installation occurs rarely, having this process automated offers significant improvement in efficiency and accuracy.

As powerful as Synergist is, a number of utilities have been developed to increase efficiency. As an example, after re-compiling many forms in an application, review of separate "DBG" (debug) files for each form is still required. This led to development of a program (written in dBASE III Plus with Clipper features and compiled using Clipper) called "Synergist Manager" which reads the DBG files and collects information on errors found during the compile process. A report is then generated listing each form and the number of errors. In addition, a report is created which includes the text of each DBG file in which errors were found -- printed in form number order to make it easy for staff to find and resolve any problems which were identified.

Over time, more features have been added to "Synergist Manager," making it even more valuable. Some examples include:

- Obtaining a list of successful and unsuccessful compiles.
- Generating a list of successful and unsuccessful FINSTALs.
- Comparing timestamps of forms in development with those in production to determine exactly which need to be re-installed in the production environment.
- Comparing timestamps of forms under development in one disk volume with those in a "master" volume to determine which have been changed.
- Printing forms in "batch."
- Consolidate DBG files with error messages into one file.

Continuing attention to improving support systems not only improves programmer efficiency and reliability, but reinforces the value placed on creativity by continuously looking for new additions.

End User Relationships

One of the major challenges to data processing management is to facilitate development of applications which meet the real needs of end users. Traditionally, interviews with end users and supervisors together with studies of the work flow are followed by a period in which data processing staff develops the application largely in isolation from production settings. Synergist provides the opportunity to change this approach.

Because of the ease with which screens are "painted" and edits made using Synergist, end users can sit side by side with programmers during the development phase to provide on-going involvement and feedback. This substantially improves the quality of the final product. By committing to such a process, the maturity of a product when installed in production is often comparable to a second or third version using traditional approaches to user involvement.

Having discovered the impact this side-by-side working relationship between programmers and end users can have has led to looking for other opportunities for their involvement. For example, end users are able to write "help" screens and prompt lines. This also adds to the maturity of the finally installed product because documentation and help screens are often last to be developed, if developed at all. End users are also able to anticipate the impact Synergist applications will have on job routine, job structure and production. Opportunities for using Synergist features such as User Keys to achieve even greater efficiency in production settings can also be identified.

The opportunity for end user involvement doesn't necessarily cause it to happen. End user management needs to be counselled on the wisdom of identifying and releasing experienced users to work with programming staff in design, development, testing, training and installation. Efficiencies achieved in bringing an application into production as well as those gained in the new production setting usually more than offset costs of committing resources to this level of involvement.

Relationships with Software Developers

MEDNA worked closely with the developers of Synergist as this new software technology was conceived and early versions prepared. This level of involvement provided unusual opportunities for us to influence initial design questions. Later, our use of Synergist in application development revealed a number of additional design considerations which were important to meeting our needs. The very closeness of our working relationship gave MEDNA an unusual level of influence with the developers which we have valued.

Soon however, it became clear that a delicate balance needs to be achieved between sufficient involvement to influence the product and enough distance to gain stability required in completing applications. An additional sense of balance arising from receipt of early versions for testing was required to assess the importance of asking for additional features or clean-ups and understanding the limitations which competing priorities and limited resources have on schedules for software developers.

After some months of testing early Synergist versions, we realized that a project could not usually be pushed to completion while participating in testing. As a result, Synergist developers were asked to devise alternative means of quality assurance. In this transition period, stability required to put applications into production was achieved, but this advantage was somewhat offset by delays while developers obtained resources needed to assure quality control.

With Synergist being purchased by an increasing number of organizations, yet another phase of the relationship will unfold as these organizations compete with a larger and larger user base for attention to problems and enhancement requests.

Learning the basics of Synergist fills a limited need at this stage of development because new and important techniques are being found almost daily. Creating an environment in which this new information can be passed from one programmer to another is a high priority. "Team" assignments, a "bank" of examples, and a Synergist User's Conference have helped meet this need.

Working as a "team" increases creativity as team members challenge each other. Knowledge is passed not only by sharing, but also by observing what another programmer does. A team approach improves the ability to focus on and resolve programming challenges and substantially decreases the learning curve.

Having immediate access to a "bank" of examples is also very helpful. For example, a wide variety of data base access techniques are handled automatically by Synergist. This is one of the most important (and most difficult) areas for new Synergizers. Some examples available for review include:

- Verification Access (access another file to verify existence of a record)
- Chained Access (access a 2nd file based on data obtained from the 1st file; access a 3rd file based on data obtained from a 1st or 2nd file; etc.)
- Access based on hidden fields
- Launch fields
- Optional access (access a "people" file based on a unique key, Social Security Number, or name)
- Adding records using Host Generated Keys
- Combined access (ie. search for name based on user entered data and if not found search for name based on soundex)
- Access with qualifiers (including qualifiers which span several files)

As another means of extending the ability to share ideas and techniques, effort has gone into organizing a user group and hosting the first conference of Synergist users. 65 persons attended and participated in sessions covering such topics as:

- Data Base Access
- Debug Utilities and Techniques
- Security
- Application Merging
- User Routines
- Programming Techniques
- End User Training
- DP Training
- Programming Standards
- Support Utilities
- Performance Findings

In addition to sharing information between users, conferees also had the opportunity to carry on constructive discussions with the developers of Synergist.

While considerable time went into activities required to host this conference, preparing individual presentations and attending the conference, a great deal was accomplished. Much was learned from other users, ideas for improving support service were solidified, and a network for improved communication between users was initiated.

Summary

Management of an environment committed to application development using a revolutionary new language tool has offered many challenges. Some of these include:

- Balancing participation in product testing and review with attaining stability required to complete applications
- Creating utilities which yield more efficient and reliable development
- Training new staff
- Establishing new relationships with end users
- Developing a long term process for user group interaction
- Creating programming standards for a new language
- Establishing reasonable expectations
- Providing hardware configurations which maximize programmer efficiency

These challenges provide unique opportunities for staff and management to test their patience and skills. For MEDNA however, the results have already justified investing in a new language tool which provides a solid foundation for application development and maintenance well into the next decade.

DISASTER RECOVERY

CAN YOUR BUSINESS REALLY RECOVER?

Presented by:

THOMAS J. DOOLEY, JR.

Consultant to Management
9422 Braeburn Glen
Houston, Texas

(713) 774-8846

DISASTER RECOVERY - CAN YOUR BUSINESS REALLY RECOVER?

1. INTRODUCTION

DISASTER RECOVERY PLAN: A collection of well planned and tested procedures and data assembled into a single document which, when used, will direct the re-establishment of ALL functions necessary to support day to day business operations, which have been interrupted or rendered inoperative by some uncontrollable event.

One can cite many technical and business reasons for developing and maintaining a disaster recovery strategy, but the fact of the matter is that the subject rarely comes up, and when it does it always gets the lowest priority. In fact, however, it should be first and foremost on the minds and in the planning efforts of the top management of all companies. Trite as it may sound, most companies today can do very little business, if any at all, without the capabilities provided by their computers and networks. Being without a computer is comparable to not having anyone to answer the telephone.

Corporate management and the Boards of Directors are directly liable to the owners and shareholders for the stewardship of the company, and as such are constantly reminded by their auditors that they should maintain a reliable Business Recovery Plan, which includes the complete recovery of all computing facilities available for business operations. One may be led to believe that this type of admonition would cause more companies to put a high priority on developing such a plan, but unfortunately, the advice goes unheeded or gets the lowest of all priorities.

For those that heed the admonition of their auditors and shareholders more advice can be passed along. A business recovery plan is not something that can be developed overnight, scribbled into a 5 page document and then stuffed into the DP Manager's center desk drawer, merely to satisfy the auditor's request in his management report. It is a very serious facet of corporate management's responsibilities. It involves the entire company (personnel, equipment, facilities, profits, etc.) and should receive as much attention as would be given to the development of a major business system.

Preparation of the Business Recovery Plan should also follow the same procedures that are used to develop other systems. Definition, preparation, testing, implementation, and maintenance are the majors areas of concern, and the requirements, as well as the necessary project steps, are the subject of this paper.

Each portion of the development phase is important in its own right, but without the proper planning and definition, none of the other phases can be properly executed. Figure 1.1 illustrates the relative importance of these phases and tasks. Once Phase 1 or the Detailed Requirements Definition is properly completed, the remaining phases become very easy to accomplish and the time for their execution will have been greatly reduced.

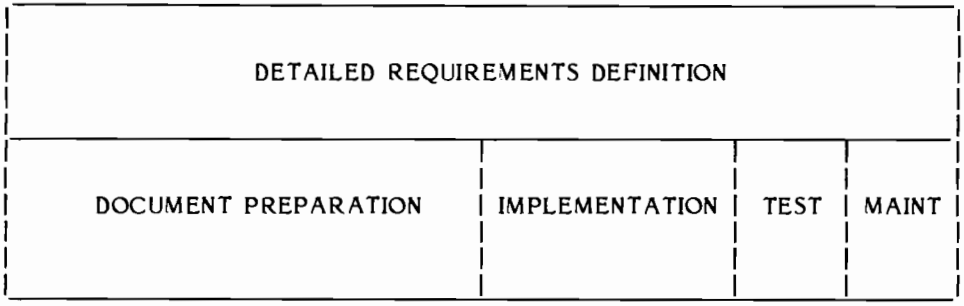


Figure 1.1

2. DETAILED REQUIREMENTS DEFINITION

There are at least eleven major tasks that must take place in order to complete the major definition of the direction and scope of the Business Recovery Plan. Just as in systems design, the more effort put into the requirements definition phase, the more complete and efficient the Plan will be. These tasks are not all encompassing, but are considered essential to a good plan. They are detailed here to give the developer some guidance and insight into the development process.

2.1 Definition of Scope

How inclusive should the recovery plan be? Document ALL areas of the business that should be included in the plan. This list should include all computer sites to be covered (mainframes, mini's and micros). Other areas of the corporation which are considered critical to its existence should be covered in the plan.

2.2 Catastrophic Events

What types of calamities should be expected? Prepare a list of all possible catastrophic events which could disrupt the operations of the business facilities previously identified.

2.3 Results of Catastrophic Events

What disruptions can be expected as a result of the previously identified calamities? A list must be prepared, detailing specific disasterous results to be expected from each catastrophe that has been identified. In addition, the impact of each catastrophe on the various areas of the corporation should also be defined.

How long should it take to recover? Determine the time required to recover from the results of each catastrophic event that has been defined. Each result of a catastrophic event should be considered individually and an estimate of the time that will be required to return to normal operation for each event should be completed.

2.4 Methods of Recovery

How will we recover? This is decision that must be made by top management. In order for them to make the decision, all alternatives must be documented and presented (Hot sites, cold sites, mutual agreements, dual sites, etc.). Each alternative should be accompanied by a broad range of associated costs.

2.5 Critical System Definition

What recovery priority should be given to each system? This is probably the lengthiest and most important aspect of the detailed requirements definition phase. Each system must be examined in light of being impacted by the defined exposures and by the resulting impact on the corporation's existence. Standard risk analysis programs are available for this process. Having determined the risk, each system should be given a priority and ranked accordingly. This list will greatly influence the definition of the recovery objectives.

2.6 Resource Requirements

What resources are required to execute each system? All resources required by each system must be identified. This includes manual systems as well as those that are automated. Resources include hardware requirements, communication requirements, special operating software, personnel, special supplies, special facilities, vendor participation, outside services, etc. The final document will be a matrix of applications and resources. It will also be used in the definition of the recovery objectives.

2.7 Plan Objectives

Based on the information that has been gathered and refined in the above project steps, a detailed statement of recovery requirements must be composed. This statement, in detail, must set forth all contingency or recovery plan objectives, all areas or functions that will be affected, and a statement concerning the importance of each function's continued operation.

After a complete review of this document, it should be presented to Management for their concurrence.

2.8 Current Plan Review

Is the existing recovery plan, if one exists, still current? If a plan exists, it must be reviewed and compared with the requirements that have been developed in this current exercise. The results of this review will be a schedule identifying the existing plan's strength for each of the currently defined critical systems. These strengths will be considered for entry into the new recovery plan.

2.9 External Services and Products.

What types of products and services are available to accommodate all requirements? In order to accomplish this task, a list of services and products that are required to ensure the plan's success, must be completed. A determination can then be made regarding vendor organizations which can

supply these services and products. Meetings should be scheduled with each reliable vendor to determine the scope of each particular service or product and to gather other pertinent information. After the meetings are concluded, a schedule must be prepared listing all requirements which can be satisfied with a particular vendor's product or service. A second schedule will list all requirements not satisfied with external services or products.

2.10 Internal Services

Are there services within the company which will satisfy some requirements? A schedule should be prepared, identifying all processing alternatives which may be available within the organization. Once this is accomplished, a determination may be made as to the capacities required and those available to absorb the processing loads. A schedule can then be prepared showing all costs related to internal processing alternatives.

Finally, a list must be prepared showing all requirements that have not been satisfied with either external or internal services.

2.11 Alternative Summary

The final process that must be accomplished in the detailed requirements definition is the preparation of analysis schedules for each critical system or area of the company, identifying all requirements and all available alternatives. All costs should also be included in the schedule.

In addition, all unsatisfied requirements must be identified and resolved, by determining possible methods, techniques or services which can be utilized to meet the requirements.

Finally, these requirements and their specific solutions must be evaluated, alternatives recommended and documented for a presentation to management. The presentation should result in management's approval to proceed with the actual preparation of a Business Recovery Document.

3. DOCUMENT PREPARATION

All of the detail requirements that have been approved by management must now be translated in detailed procedures and lists which will become the actual document, called "The Business Recovery Plan" or "The Contingency Plan". Figure 3.1 indicates the structure of the document. While this may not exactly fit every company's needs, it represents the very basic elements of a plan which can be tailored to fit any company.

3.1 Table of Contents

This, of course, is the very last section that is completed. It merely indicates the sections and all sub sections with their respective reference numbers. If the plan is developed using a good word processing package, this section will be generated automatically. The table of contents should list procedure numbers rather than page numbers.

DISASTER RECOVERY PLAN STRUCTURE	
Section 1 - - -	Table of contents
Section 2 - - -	Organization
Section 3 - - -	Team Definitions
Section 4 - - -	Backup Procedures
Section 5 - - -	Recovery Procedures
Section 6 - - -	Off-Site Recovery Facilities
Section 7 - - -	Lists/Inventories
Section 8 - - -	Specifications and Testing
Section 9 - - -	General Items

Figure 3.1

3.2 Organization

This section deals mainly with the organization of the plan, maintenance responsibility, and the overall structure of a recovery effort. It should contain entries detailing at least all of the following procedures and definitions:

A statement of objectives developed in the detailed requirements definition and formatted for inclusion in the document.

A definition of the number of copies to be distributed and the location and owner of each copy.

A procedure for the on-going maintenance of the plan, with the name and phone number of those responsible.

A form listing all revisions that have been made to the original plan.

A bar chart or gantt chart depicting each event in the recovery process as it relates time wise to all other events.

A procedure detailing the initiation of the recovery plan in the event of a disaster and for the notification of all parties affected by the disaster or participating in the recovery effort.

3.3 Teams

The successful execution of the business recovery plan depends upon the personnel assigned to carry out the various tasks defined in the procedure section of the plan. Figure 3.2 lists many of the various areas requiring some sort of team effort. Each of these areas should have a page in the plan listing the names of a team captain, an alternate captain, and team members. The responsibilities of each team must also be listed on the team page. These responsibilities will be expanded on in procedural form later in the document.

These are just a few of the essential teams needed for effecting a business recovery. Some companies may decide to name the teams differently or to subdivide others to more closely match their own situation. The smaller the business, the fewer number of teams, but the essential functions still remain the same.

DISASTER RECOVERY - TEAM REQUIREMENTS
Recovery Management
Damage Assessment
Facilities Recovery
Operations Recovery
Software Recovery
Communications Recovery
Input Recovery
Special Forms
Logistics

Figure 3.2

3.4 Backup Procedures

Most businesses have a portion of this topic under control, but more than likely it only relates to computer data and programs. The scope of total backup is much broader than computer data and programs. It should cover peripheral systems, documentation, special forms, remote site data, original data, micro computer data and programs, etc.

This section should contain procedures for all backup scenarios, including frequencies and methods for storing or filing of all backup data. All systems design should consider backup as a part of the original design project, thus insuring that the programs and the data are automatically insured against a disaster.

All areas of a business must also consider backup as a part of the normal way of conducting business. Original documents must be marked and stored for easy retrieval, and personnel must be trained in the execution of both the backup as well as the recovery procedures.

3.5 Recovery Procedures

The engine that makes the recovery plan run properly is the recovery and restoration procedure section. Procedures are required for each recovery team as well as for a variety of other recovery related tasks. Figure 3.3 lists a number of procedures that should be included in the plan. This list is by no means complete, but those procedures listed are considered essential.

Each procedure should be written in script form relative to each responsible party. No detail should be spared or omitted. A walk through should be conducted for each procedure before it is included in the plan.

DISASTER RECOVERY - RECOVERY PROCEDURES	
Recovery Management	Alternate Site Recovery
Damage Assessment	Micro Computer Recovery
Facilities Recovery	Return of Backup Materials
Operations Recovery	Emergency Shutdown
Software Recovery	Halon System
Communications Recovery	Peripheral Subsystem Recovery
Input Recovery	Evacuation
Special Forms	Salvage
Logistics	

Figure 3.3

3.6 Off-site Recovery Facilities

Even if a decision has been made not to employ a hot-site for recovery purposes, certain information relative to off-site recovery processes is essential. Information regarding contracts and other negotiations or agreements should be located in this section. Each off-site location should be described in detail, complete with names and phone numbers of contacts, maps of the immediate area as well as maps of the location itself, and a detailed listing of all available equipment and services.

3.7 Lists and Inventories

Perhaps the most difficult portion of the recovery plan is the collection of all the lists and inventories required to support the many procedures in the recovery section. These lists are varied and any particular list may have many parts, such as the Inventory Checklist. Figure 3.4 contains some of the many lists and inventories necessary for a well thought out recovery plan.

3.8 Specifications and Testing

This portion of the plan should contain specifications describing the construction requirements for the recovery library or off-site storage facility. It should also contain a list of all the required contents. This list will be used in the auditing procedure, which must also be included. The audit procedure should detail the frequency of audits and a method for evaluating the audit results.

The most important part of this section, however, is the procedure for initiating and evaluating tests of the recovery plan.

3.9 General Topics

This section is a catch all for anything that doesn't fit in any of the other sections, such as a procedure for the use of plastic equipment covers.

DISASTER RECOVERY - LISTS & INVENTORIES	
Contact Checklists	Data Entry Backup Facilities
Employees, Users	Hot Site/Shell Facilities
Vendors, Contractors	Printing Services
Inventory Checklists	Travel and Transportation
Hardware, Software	Agencies, Accommodations
Special Forms, Facilities	Delivery Services
Off-site Storage Facilities	Facility Layouts
Locations, Contents, Maps	Emergency/First Aid
Insurance Information	Equipment, Personnel
Coverage, Pictures	Radio & TV Stations
Recovery Headquarters	Emergency Phone Numbers

Figure 3.4

4. IMPLEMENTATION AND TESTING

Now that the recovery plan has been researched, proceduralized and put on paper, it can be installed properly. This is accomplished through training, a final pass to check for inaccuracies, and a thorough test cycle. Without any of these three elements, the recovery plan is totally incomplete.

4.1 Training

To properly insure that all parties involved with the recovery operation are fully informed and versed in the procedures, it is mandatory that orientation and training sessions be conducted prior to full acceptance.

The trainers must first prepare a training course outline, covering both orientation sessions for management and others who need only to be aware of the major functions of the plan, as well as full training sessions for those who will be involved in the execution of all procedures. All training course materials must be prepared for both types of sessions, followed by a schedule for all courses and sessions. Once the courses are completed, the plan is ready to be fully tested.

4.2 Implementation

Before testing commences, a quality assurance test should be made of all required items in the plan. The following assurances should be tested:

That all contracts for alternate processing sites have been negotiated and signed.

That all systems and programming changes that were to be implemented as a result of the plan have been completed.

That all additional equipment has been delivered and installed.

That all additional communications services have been or will be delivered and installed on time.

That all construction, resulting from the plan, has been completed.

That all required supplies and materials have been ordered and will be delivered on time.

4.3 Testing

The final phase of the business recovery plan, before the final stamp of approval can be issued, is the testing phase. Without a proper and conclusive test the plan is nothing but a book, but with the proper testing, management can rest assured, knowing that their business can rise from the ashes of a disaster.

The two most important steps in the testing phase are the "walk-through" and the "test audit". The walk-through will acquaint all parties with their responsibilities, without a large expenditure, and the audit will provide an outside opinion of the viability of the entire plan. Figure 4.1 details some of the items to be included in the testing phase.

DISASTER RECOVERY - TESTING PHASE
Prepare procedures for testing the recovery of all systems
Prepare a test schedule for all systems
Assemble all test materials
Conduct a user "walk-through" to review all test plans
Conduct the test
Record problems and resolve procedures and methods where necessary
Distribute corrections
Notify all parties of formal implementation

Figure 4.1

5. CONCLUSION

If you have followed and completed all the items referred to in this paper, you will have spent a great deal of time and effort for a worthy cause. You probably will have discovered a few weaknesses in some of your systems and hopefully will have corrected them. You have also had to review your hardware situation, your communication network and your micro computer proliferation, and you have probably had to make some adjustments in all three. But, for all your work, your management can rest easier now and the stockholders can feel a little more secure about their investments.

The opening of an electronic mail architecture. An Approach.

Colin Draper
Hewlett-Packard Limited
Wokingham
England

Summary.

In the current release of HPDeskManager there are various ways in which applications may take advantage of the services of the product. The future of electronic mail requires that the current solutions be made into far easier to use and more powerful interfaces.

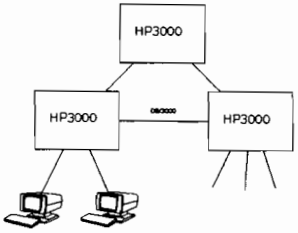
This presentation will look into possible methods of providing access to gain a more general open systems architecture, considering for the user and vendor both the costs and requirements. As upgraded feature sets are designed, these must be included without affecting the current investment but also without restricting the implementation of these facilities.

We have seen in the past that the functionality of the product is not the only consideration, it must also be easy to use, and offer lasting benefit to the users. This presentation will try to show some of the considerations now being investigated in order to meet these goals.

In the beginning.

As the future directions are born out of the experiences of the past, a brief look at the history of the current provision is appropriate.

From the very first release of HPDesk, then known as HPMail, the product has often been used in ways that the designers had not considered. The very first site was very enthusiastic about the ability to transmit files as part of the message content. This simple ability soon showed up as an extremely cost effective method for distributing program updates to geographically remote machines in their network. With the popularity of HPMail, the number of sites running the product started to grow steadily, and with it the knowledge of exactly how the product was being used came to light.

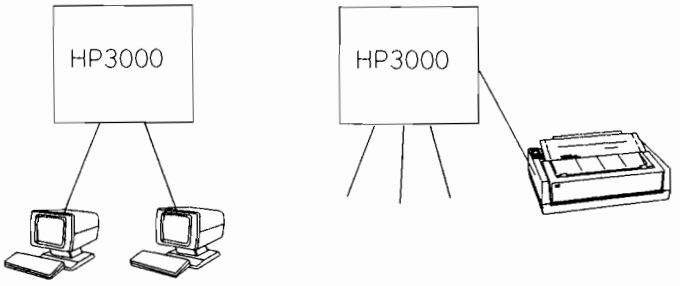


Then the first method of programatically accessing HPDesk! The user needed to automatically send out reports to a fixed distribution of users. So, the commands and input for a normal HPDesk session were placed in a job stream, the report being copied into HPDesk as an attached text document and mailed. Somewhat primitive, but none-the-less an effective solution to the problem.

Then there was light...

The following development of an interface to HPDesk came initially from the requirement of the internal HP community. At this time it was generally thought that the experiences of HP with its own network of HPDesk was approximately 1-2 years in advance of most of our customers. We had an internal electronic/paper mail delivery system (COMSYS) that was developed long before HPDesk and used by many people. This system could ideally be replaced in total by HPDesk, apart from its ability to generate paper copies. If this ability was provided it would still give a roll over for sites that had not yet brought up HPdesk for all its employees.

The FSC (Foreign Service Connection) mechanism was conceived. The initial requirement was to interface to the COMSYS system in a way that HPDesk users could send to all destinations on that system. The difference being in the ability for some of the users to be *real* HPDesk users, others receiving paper copies, without the originator knowing which type of recipient they were. The FSC connection was in the outgoing direction only. It was to be an ASCII style interface, so that the result could be as near to a printer image as possible.



The problem faced was really "What format do we use?". At that time there were few standards for electronic mail: NBS, SMTP and ARPA. The decision was to go with ARPA as it was a text only standard, and was much more widely accepted than any of the other choices (at the time).

In version A.03.00 the loop was completed. The facility for messages to be accepted into HPDesk was now provided. This gave several problems to solve in the area of recovery from unintelligible and misaddressed messages.

Looking back on the choice. Hindsight is a wonderful thing.

There are several areas of interest that have taught us a great deal about providing a user accessible programmatic interface to electronic mail. These could be summarised as: complexity; performance; error recovery and upgradability. A further complication that has recently presented itself is the emergence of a single standard for electronic mailing (despite several variants on a theme).

The first problem that really showed up was the complexity of actually using that interface. Many customers have pursued the implementation of FSC connection and have been very successful in connecting their systems to HPDesk. Being realistic though, the number of installations that have actually used the FSC capability is very small in relation to the installed base. One key factor that has deterred people from using FSC is the apparent complexity of the implementation. ARPA format relies on the data looking like a series of bytes, whereas MPE insists on presenting files as records rather than a stream of bytes. Further problems arise in the understanding of that data. If the FSC output from HPDesk is to be understood, the data stream of the FSC message requires a form of parsing to extract the relevant pieces of information. This parsing effort is not simple.

From the structure of the FSC message, and the requirement to parse the message data to gain the recipient information, a rather important problem can be seen: performance. Everyone is concerned about the performance of any system, the parsing of ARPA format on the HP3000 is not always as efficient as it might be.

With any standard, as the label implies, there is a fairly rigid definition of the message structure, hence its information fields. With simple message format definitions, a growing mail system can easily require facilities that are not incorporated in the standard. This was true for the acknowledgement system, and appointment details. Fortunately the standard did allow for proprietary defined fields (X- fields) that could be used by co-operating mail systems.

As the FSC mechanism is essentially a batch oriented system, the error recovery for problem documents is not always simple, or full of information. In this case the error mechanism is a simple IPC file to which the name of the document is written, along with a single integer error code. This does not allow a full analysis of the problem, and only works for hard problems (bad format, database full, data too big etc.). There is no way to be able to diagnose when some recipients names are not known, or their addresses are corrupt in some way. In short, there is no direct interactive communication back to the application.

Upgrades can in some circumstances cause problems. If the application reading messages from HPDesk is designed in such a way that it knows all possible header fields, then as the FSC interface is enhanced to cover more features of HPDesk, there is the possibility of new fields being presented. If the application is written such that it cannot skip these fields then there are potential problems. This is generally where a true ARPA format parsing routine is used and it is not flexible enough to handle new field names.

What can we learn from this experience?

The lessons that come from these past experiences must be factored into the design of new systems, especially as the complexity of the systems themselves is increasing. Looking into the future there are many possible ways that could be used to overcome the problems that currently face us. I would now like to explore some of the more realistic methods that can be put forward as possible methods for future implementation. One thing is obvious, the future systems must be more interactive with the calling application.

The batch interface can be seen as a rather unsuitable mechanism for connecting into the mail system. Interaction is to some degree necessary to make error condition handling acceptable. This implies that the only realistic methods of providing connection mechanism would be through some form of procedural interface. Thus, the future directions are most certainly directed into procedural interfaces, providing the direct feedback for error situations, and also the confirmation when the mail system has taken charge of the message for delivery.

There are three distinct methods that I would like to look at in more detail: multiple procedure call, generic procedure call and transaction file. Each of these approaches have their merits and drawbacks. They all are intrinsic mechanisms of various forms, hence the error reporting is directly fed back to the calling program.

Multiple procedure call.

The idea of multiple procedure calls, is simply to provide an intrinsic for every situation. For a mail operation of say, sending a text file to some user there would need to be several intrinsic calls.

Signon_as_user,
create_message,
set_msg_subject,
add_recipient_name,
add_attached_file(text_filename),
post_message,
signoff_as_user.

Advantages of this approach, are that the ease of use is obvious, the user simply selects from the intrinsic list exactly the fields/items that they require and call them with the appropriate parameters.

The disadvantage is that the upgradablity is a problem, for any new feature, such as setting an appointment on the message, needs yet another intrinsic. Very soon the number of available intrinsics rises to an unmanageable level.

Generic procedure call

This is a modified version of the multiple calls to allow grouping of the functions into a single procedure call, with an option to indicate which particular field is being specified in the call. This is the method being adopted by the X/OPEN[1] committee defining the XMI procedural access routines for UNIX[2]

For this case the calls would be more like,

identify_user(SIGNON, username),
message_action(CREATE),
set_attribute(SUBJECT, subjectname),
attach_file(FILE, filename, TEXT),
message_action(POST),
identify_user(SIGNOFF)

This does have the advantage that the underlying procedure calls can be changed to accept more functions without showing any visible external signs. This is in many cases the way that the newer MPE intrinsics have been developed to circumvent the upgrade problem, c.f. JOBINFO, PROCINFO, FFILEINFO. The characterisation of these intrinsics is that each call passes a small amount of information. This can be a fine method for performing operations in many cases, but there are drawbacks that tend to occur in non-MPE kernel operations (application style programs).

Providing such an interface can cause problems in maintaining a common area (see V/3000) that the intrinsics use to hold cross call variable information. Many of the subsystems that are intrinsically accessible have by their nature to be intimate with the internal structure of MPE, and can use the MPE DSTs to hide any necessary common area (IMAGE), or even put it in the users stack (DS in part). Mailing systems are generally viewed as applications, mainly to keep them out of the kernel (where they don't belong) as the services they require are generally speaking those provided to ordinary application programs. The common area problem remains, not only in preserving it, but also preventing its corruption.

Added to the other problems, the two previous methods also share a rather interesting problem that has always caused problems for the HPDesk processes, and could certainly be acute in these cases: *stack size*. During many operations in the HPDesk mail system, the local data requirements that would be placed on stack can easily be in excess of 4K words. For a number of programs the idea of adding another 4k to their stack would be impossible, as they are VERY close to the stack limit already. Parameter information can also be fairly large, for instance an HPDesk name with full foreign address can be 563 characters, so specifying the full distribution list in one procedure call would be impossible.

Should the intrinsic calls in their processing cause any form of interrupt that cannot be trapped by an application program (stack overflow, bounds violation) then not only does the intrinsic code flake, but also the users process with it, allowing no clean up for the application.

Transaction file

This concept relies on the application creating a *transaction file* which contains a series of transaction records, each describing a simple logical message construct. Within the record, the first few words have a fixed meaning: a transaction code (i.e. what type of data) and a result code field. The remainder of the record is dependent on the transaction, there may be several fields, for instance in the case of setting an appointment(date, time, length), but there is a fixed definition for that record. These data fields would typically describe something like the subject, or creation date, or one of the recipient names. This fixed record format makes the file ideal for creation using generally used languages such as COBOL or PASCAL.

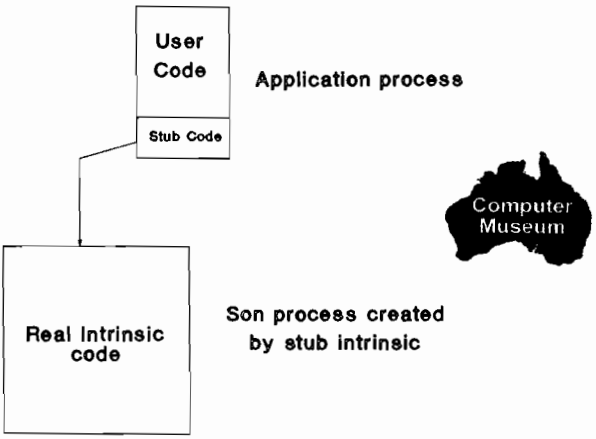
Once the file is complete, describing such things as subject, creator, dates, recipients, all attached document files, then the transaction file is passed as a parameter to a **Send** intrinsic. The transaction file would be processed, any errors in the field description being returned in the result code field of the record. An overall error status code would be returned to show if there is any reason to look into the transaction file for detailed analysis of the individual errors.

The intrinsic calls in this case are:

signon(user_name)
send(trans_file_name)
signoff

The complexity is very much reduced, having in this case only three intrinsic calls, inherent is the ability to use standard template transaction files.

For this case, as the interaction of the user program code and the intrinsics of the mail system are very small, the actual work of the intrinsic call can be passed to a son process. The son process would prevent all the normal problems of stack limits, and the problem of the intrinsics code causing a process abort.



Using a transaction file, therefore has the ability to isolate the true intrinsic code from the users process and stack. Upgrades to the product can be easily achieved by adding new transaction codes, with the firm statement that records with unknown transaction codes should be ignored. To upgrade the product, only the program file of the son process needs changing, rather than the more time consuming SL updates. The performance issue may, on initial inspection seem to be a problem, but a closer examination shows that the interaction between the father/son process occurs very infrequently due to the quantity of data passed per intrinsic call. The only possible area of performance concern that is not intuitively answered is the overhead of accessing the file. Though it may be judged to be sufficiently small in comparison to the gains in other areas.

Conclusions.

Both the generic and multiple procedure call methods offer very quick access to the underlying code, giving a similar interface to that of MPE itself. But the concerns are both in upgrading, especially for multiple procedure call method, and the requirement of stack space.

The transaction file removes the user's process stack concern, and possible process abort situations, but may well have some small overhead in the access to the file. It does however give good upgrade opportunities, is simple and allows for template definitions. This has enormous advantages on the HP3000 16 bit family as environment issues are a great problem, though many of those issues are resolved running on the HP3000 series 900 machines in *native mode*.

There is opportunity for all three possiblities to be implemented in future, though this may not necessarily be the most appropriate course of action. The route followed will very much depend on the type of interface required, either simple and easy to use, enabling full access to HP defined facilities via transaction file, or the standards route which would favour the generic procedure call mechanism.

Colin Draper.

Colin Draper has worked on the HPDesk prohject for the past five years. In that time has been involved in most areas of the product, especially the user interface and transport. More recently, the major contribution was in the area of application integration and script files for the HPDesk B.00.00 release.

Notes.

1. UNIX is a trademark of AT&T Bell Laboratories.
2. X/OPEN is a trademark of the X/OPEN group members. The X/OPEN group of 10 industrial companies aims to encourage applications portability accross different vendors UNIX systems.

The Future is now ...

THE EVOLUTION OF ARTIFICIAL INTELLIGENCE WITHIN HP MANUFACTURING SYSTEMS SOFTWARE

by
Joseph H. Dugan
Sr. Manufacturing/Management Consultant
W. D. Farlow and Associates, Inc.

THE EVOLUTION OF ARTIFICIAL INTELLIGENCE WITHIN HP MFG. SYSTEMS

The Future is now ...
The Evolution of Artificial Intelligence within HP Mfg. Systems

by

Joseph H. Dugan
W. D. Farlow and Associates, Inc.
South Holland, Illinois

As computer software and hardware technology continues to evolve, the overall successful application of Manufacturing Resource Planning (MRP II) systems continues to remain elusive, with the majority of implementation sites reflecting increased costs or no tangible benefits at all. Although not necessarily reflective of the MRP software itself, "real-world" examples abound of increased personnel costs to maintain the MRP system, ineffective or work-around approaches due to the MRP complexities (causing ineffective utilization of the MRP system), perpetual MRP implementation projects, escalating costs for consultants and required MRP education and "nightmarish" tales of key MRP personnel turnover causing complete chaos within an MRP II installation.

With the advent of the Computer Integrated Manufacturing (CIM) technology/approach and the establishment of integrated data flow throughout various hardware configurations, new applications of Artificial Intelligence within MRP environments can be established now with a greater probability for successful implementation and increased direct benefits. This year, industry will see the first artificially intelligent MRP applications become available on several computers. These applications will provide the potential for the manufacturing industry to realize the ultimate MRP optimization...maintaining primary data while the computer interactively assists the user and makes decisions/takes action from its data and knowledge base. These intelligent MRP extensions should create such powerful cost benefits to industry that a new generation of MRP systems will evolve.

The purpose of this paper is to provide insight into the application of Artificial Intelligence within MRP systems and its anticipated affect within the overall CIM environment.

Artificial Intelligence

During the past ten (10) years, significant advances have occurred using AI approaches within the custom application arena. In the past, this AI development required complex computer hardware environments and typically was laboratory or military application oriented. Recently, these "knowledge-based" or "expert" systems have begun to appear within Commercial applications available to the end user within conventional business and even PC computer environments. Specialized software environments are being replaced with conventional languages.

Artificial Intelligence, in its basic, inherent sense, is the ability of the computer or machine to acquire knowledge, analyze, take action or make decisions and expand its knowledge base. A more simplistic definition may be a "learn-analyze-respond-learn, etc..." cycle system. AI, by itself, provides little, if any, justification for excitement in the manufacturing community. Applied Artificial Intelligence or the enhancement of a manufacturing application using AI places a powerful tool in the hands of the manufacturing industry.

The application of AI into MRP II systems within a CIM environment provides enormous benefit potential to the manufacturing industry. The following represents only a few MRP/AI applications available today:

- Protection of the production plan through computerization of priority planning, dispatching, etc...
- Reaction to MRP exception conditions as they occur, reducing costs throughout the manufacturing cycle.
- Factory Optimization through AI enhanced analysis of the utilization of manufacturing resources.
- A captive, resident MRP expert to guide users through their work day, continuously educate, and standardize approaches, thus reducing personnel turnover problems mentioned previously.

- Computer Integrated Manufacturing applications guided from within intelligent processors using MRP as the foundation and expanding the closed-loop throughout vendors, material handling, work-cells, etc...

To fully realize the potential of applied AI within Manufacturing Resource Planning systems, the discrete applications within the integrated MRP system must be understood, realizing that extensions to these systems within a CIM environment can multiply the potential tenfold or more.

Manufacturing Resource Planning

Manufacturing Resource Planning Systems (MRP II) provide an optimum base for Computer Integrated Manufacturing concepts. At a minimum, these systems provide basic integrated applications which assist in the maintenance, control and reporting of data throughout the manufacturing facility, including:

- Engineering data management and control of part specifications, Bills of Material, Engineering revision and effectivity planning and control, workcenters and product routings.

- Inventory management and control of supply and demand.

- Master Production Scheduling management and control.

- Material availability analysis and control through the use of Material Requirements Planning techniques.

- Labor and machine capacity analysis and control through the use of Capacity Requirements Planning techniques.

- Production Activity Management and Control including Material Movement Tracking and Work-in-Process Plan Execution and Measurement.

- Procurement Management and Control.

- Manufacturing data integration into the Financial Accounting systems and applications including General Ledger, Accounts Payable, Accounts Receivable, Order Entry, Payroll, etc.

These systems are structured to optimize data accuracy and data flow. Performance monitoring subsystem applications are also necessary for these systems to be useful tools. On-line, interactive "ease-of-use" user interfaces should also be an integral part of any Manufacturing Resource Planning system.

The evolution of MRP systems begins with simple stock cards manually maintained for each part in inventory. When parts were received, the quantity received was added to the stock balance on the card. As issues occurred, the issue quantity was subtracted from the stock balance on the card. These inventory "carder" systems were audited and updated during physical inventories. Parts were built or purchased based on manual Bills of Material and restock points annotated on the inventory cards. Work-in-Process was typically quantity based and maintained by production expediters.

These manual systems were eventually computerized to provide computer maintenance and tracking of inventory availability and Bills of Material. Shortages were identified upon release and new supply orders were initiated. The old restock points became triggers for the computer system to initiate new supply orders. These first computerized systems were known as "order-point" applications. Bill of Material Processors were then added to enhance the order-point technique, allowing summarized quantity requirements and part/by/part inventory availability checking, annotating the need for parts to be ordered.

Further evolution of these systems lead to the "time-phasing" of material requirements and supply using projected availability of inventory based on requirement and supply schedules. The planning of supply was initiated to coincide with scheduled demand. Bills of Material were exploded based on the Master Schedule with net requirements time phased along with planned supply. Priority planning concepts were included to provide visibility and tracking to assist production and procurement personnel in providing identification of "when" material was needed to meet the Master Schedule. In order to provide the time phasing of the Bill of Material requirements, lead times were also used to schedule the exploded net requirements from the Bill of Material explosion process. These material planning systems were appropriately called Material Requirements Planning systems.

MRP II or Manufacturing Resource Planning systems are the extension of Material Requirements Planning applications throughout the manufacturing facility, including Machine/Labor Capacity Requirements Planning, along with Resource Utilization Tracking and Control. The Material Requirements Planning applications are just one application within these comprehensive integrated systems.

The Application of AI within Manufacturing Resource Planning

Comprehensive MRP II systems require enormous amounts of data maintenance, analyzation of exception conditions and determination of action to be taken. Successful implementations have, at best, been few and far between. Extremely well-educated users are required to effectively utilize these systems, and the "true" total costs of implementation continues to grow. Considering the personnel education, computer hardware and software and additional personnel expenses to keep data accurate, many companies have found themselves in a serious quandary. They are unable to afford either the cost of implementation, or the cost of maintaining these systems effectively. Granted, tangible benefits do accumulate rapidly to offset these costs when MRP II is implemented successfully, but many companies never get to the 95%+ data accuracy levels required to gain the significant benefits. Add in continuing problems of key personnel turnover and on-going education, along with top management's constantly changing priorities, and you can see the difficulty in achieving a successful on-going implementation.

Enter applied AI into the MRP II system and many of these inherent problems begin to reduce or even disappear. Using some of the general MRP II applications as a base, the following represents a few of the obvious benefits. Applied AI represents "expert" application assistance throughout the MRP II systems.

Within the Engineering data required to effectively utilize MRP II, Bills of Material and effective use of Engineering change provide beneficial opportunities for applied AI. In many companies, the same part number may be identified several different ways with the various Bills of Material. Applied AI, along with Group Technology Specifications, can eliminate this problem altogether by interactively assisting the user in properly identifying the part or Bill of Material. Duplicate parts and Bills of Material can also be tracked down and eliminated using applied AI. Bill of Material accuracy can be increased via automated feedback mechanisms to the applied AI in which action is taken by the computer to maintain the Bills of Material in a timely manner. Engineering change and cost effective use of effectivity is a natural for applied AI within the MRP II Engineering environments. Additional benefits become obvious within the product labor standards and routings, along with cost estimating applications.

Production Control Scheduling and Dispatching applications provide an opportunity for AI to ensure proper, accurate and timely priority planning and control. The computer assists or makes decisions for the most economical form of lot sizing based on the time phased requirements from the Material Requirements plan and the other resources of the factory. As changes occur, applied AI in this application can reset to protect the Master Schedule choosing alternate work-centers, deciding on order splits, re-aligning priorities and rescheduling, if necessary. The applied AI logic can effectively manage/maintain machine preventive maintenance cost effective plans without negatively impacting the production schedule. Work-in-Process inventories can be reduced, along with effective utilization of manufacturing resources and facilities. Data accuracy can be increased via automated feedback mechanisms to the applied AI where the computer will take action to maintain "WIP" data accuracy. Total quality control will also benefit from the applied AI in this arena.

Material Planning and Control provides additional applications in which AI can be effective. Maintaining inventories at the most cost effective level, whether that be Just-in-Time (JIT) or increased data accuracy of storeroom material. Utilization of AI in procurement applications can make significant reductions occur in regards to purchased material, time phasing grouped buys or releases of material requirements in the most cost - effective/timely methods available within the vendor base of the facility. As in Production Control, priority planning in the material arena is also an opportunity for applied AI, even to the extent of maintaining priorities via information analyzed and diagnosed from vendor feedback or vendor computer systems.

In addition to the above, applied Artificial Intelligence can be utilized to reduce data entry personnel costs, convert data from existing systems, provide daily prioritization of what MRP planning personnel work on and analyze performance measurement trends. Simulation capabilities for Master Scheduling, Production and Material Planning, Cost Estimating and Cost Analysis become much more precise and timely through the use of applied AI.

One of the most significant benefits within an MRP II system with applied AI is the "captive expert" concept for training new users and ensuring the MRP II data is maintained at the high levels of accuracy required. Applied AI can interact with the user to ensure continuing education and training at a pace tailored to the particular user. The applied AI can also ensure that the user performs his work in the most timely manner at the highest accuracy. Imagine a system that analyzes data throughout the day and steps the users through required maintenance in priority sequence, ensuring data accuracy. These systems exist today!

Extending MRP/AI Systems within a CIM Environment

Powerful extensions of MRP II and applied AI become evident within a CIM environment. Integration to and from Factory Automation Cells, Material Handling Systems, Automated Storage and Retrieval Systems and Inspection/Test Systems provide interactive feedback throughout the manufacturing facility, allowing AI an increased knowledge base and feedback data loop.

With an AI enhanced MRP II system forming the base within a CIM environment, it is now possible to effectively optimize factory applications. A case in point would be an integrated Material Handling System/Automated Storage Retrieval System integrated with an applied AI enhanced MRP II system. The user has set up parameters to effectively optimize cost of inventory, along with protecting the Master Schedule. A partial receipt occurs in Receiving for a critical component needed on the floor for a parent at its final operation. The parent is currently within two (2) days of needing twenty (20) pieces of this critical component, but only ten (10) have been received. The remaining ten (10) are due in on Day #3 (one day projected past due). Consider the following scenario and see the potential an AI enhanced MRP system brings to the manufacturing facility:

1. The priority of the 10 piece lot in inspection is going to be set to coincide with need.

2. The system notifies the buyer and vendor that the part is needed on Day 2. Response back to the system indicates the vendor is having difficulties and may slip another five (5) days.

3. The system makes a decision to split the parent order working on the floor into two (2) separate orders for quantities of ten (10) each. The split order affected by the shortage is analyzed for optimization. Analyzing the product data, the system decides that the component can be installed at the next level of build. The final operation is cancelled for the ten (10) affected by the shortage. At the next level of build, the operation is added to install the critical component.

4. On the day required, the ten (10) components are picked from the ASR system and sent to the workcenter and operation via an Automated Guided Vehicle (AGV). These commands are issued from the MRP system.

5. The AGV reports delivery to the MRP system at the workcenter, and the MRP system updates its dispatching data. The robot assembler at this station is told via MRP to work on this assembly next.

6. Meanwhile, the final inspection station for the parents short the critical component is notified that these parents are to be inspected less the components, which will be installed at the next level. Inspection acknowledges and accepts the semi-finished assembly. The MRP system commands an AGV to pick up these parents and take them to the next level of assembly, the MRP system updating all of its data through the CIM feedback loop. The completed subassemblies in Lot #2 are married back up with the others via AGV after inspection.

7. Using an alternate routing which the MRP system has initiated based on product routing data within the system, the MRP system updates the scheduling and queries procurement and the vendor again for additional updates. The data back to MRP indicates another partial receipt of five (5) earlier than the five (5) days slippage (on day required +3) and the remainder due in on the day promised (day required +5).

8. The system reinitiates priorities and decides to hold the second partial receipt in inspection until the remainder are received from the vendor. Procurement is notified that a second partial receipt is not required and the system will wait until all ten (10) of the shortage are received.

9. All work on the upper level is completed on the parent until the point where the component must be installed. The system re-establishes schedule and priority control placing the affected order in a "wait" status.

10. The system responds to the receipt of the ten (10) short components, expediting them through inspection via priority dispatching and an AGV picks up these components, delivering these to the location of the order waiting.

11. Assembly continues on schedule because of the response of the AI system to the exception conditions, and its use of work-around techniques to accomplish the tasks necessary to protect the Master Schedule.

Sounds a bit like Star Wars, but this technology exists today and costs no more than a standard comprehensive MRP system within a CIM environment. Reflecting on the amount of transactions it would take to accomplish the above, one can see the power of the computer effectively utilized rather than production and material personnel reacting with transactions to accomplish the same. In a typical MRP installation, there wouldn't have been enough seasoned personnel to handle these actions in a timely fashion. Compound these tasks with all the other difficulties encountered within a manufacturing facility, and you may realize the potential.

Integrating these advanced AI based MRP systems with Computer Aided-Design/Computer-Aided-Manufacturing (CAD/CAM) applications and Computer-Aided-Process-Planning (CAPP) applications again enlarges the knowledge-base and feedback loop. AI based MRP systems can ensure proper utilization of these systems and can effectively manage the data interaction.

The Difference: MRP II Today and MRP II with Applied AI

Although it would considerably lengthen this presentation in scope and subject, it may be beneficial to review some major differences between the conventional MRP II systems in use today and the advanced MRP/AI systems becoming available. The following represents a brief summary to depict major differences and requirements between the two:

	Conventional MRP II	MRP/AI
1.	To be effective requires manual data maintenance to ensure 95%+ data accuracy.	Requires data accuracy at the same levels with the inaccurate data providing the margin for error ratio. MRP/AI will take steps to ensure the data accuracy is maintained at extremely high levels.

	Conventional MRP II	MRP/AI
2.	Requires users to analyze data and perform additional data maintenance.	In many instances, MRP/AI will analyze the data and take action. In other cases, it will prioritize user actions, walking them through daily required activities.
3.	Exception conditions are prioritized and generated for user analysis. Response to exception conditions is in proportion to the amount of exceptions.	Exception conditions are acted upon as they occur.
4.	Increased level of educated key personnel required to effectively utilize the system.	Lower personnel requirements training is on-going at the users pace controlled by the computer.
5.	Data analysis and action is left to user interpreting reports.	MRP/AI takes action as conditions require or recommend action based on detailed analysis.
6.	Mundane data entry, whether interactive or batch, is required to maintain MRP data.	Much less data entry depending on the feedback loop (Note: CIM environments significantly lessen data entry requirements.)
7.	Scheduling and dispatching priorities are maintained through manual interactive interface with the system.	Scheduling and dispatching priorities are constantly maintained by the computer.

	Conventional MRP II	MRP/AI
8.	At times difficult to respond to manufacturing environment changes.	Much more flexible to a changing environment.
9.	Implementation/conversions require massive efforts in some instances.	Implementation and conversions utilize AI logic for increased effectiveness and shorter implementation lead times.
10.	Simulation logic is limited at best.	Simulation of complete facility is dynamic, comprehensive and detailed, allowing users a complete simulation/estimate with reactive timeframes.

Within each application of MRP II, additional detailed differences can be defined. The best way for you to appreciate these systems is to see them yourselves.

MRP IV?

Does this mean that the MRP II systems in place or being sold today will be obsolete in the years to come? Most probably, the answer is "yes", although not overnight. The enormous tangible benefit potential of these new systems will certainly force software developers to a new generation of MRP systems within the next five (5) years. Only the inherent fear of users and management personnel towards AI and new MRP systems may slow down this evolution. There are MRP/AI installations currently on-going throughout the manufacturing industry sponsored by management who can foresee the potential. If any one of these becomes immediately successful, the evolutionary process may speed up.

If you are considering implementing an MRP system, remember to review these new systems prior to your selection. Many of the comprehensive MRP/AI systems allow the user to control the amount of computer action taken, providing the capability for the user to grow into the AI environment. Another consideration, as always within the MRP implementations, is the fact that these systems are not available from major manufacturing software firms. You will have to look at smaller, technology leader firms for MRP/AI solutions until the large software firms evolve.

MRP IV, applied Artificial Intelligence within Manufacturing Resource Planning and CIM environments, is evolving now; "heralding-in" a new generation of Computer Integrated Manufacturing systems.

OPTIMIZING HARDWARE UPGRADES

AUTHOR :

Brian Duncombe
Triolet Systems Inc.
41 Horner Ave., Unit 1
Toronto, Ontario
CANADA
M8Z 4X4

(416) 255-5767

ABSTRACT:

At one time or another, most of us are faced with the prospect of upgrading our current hardware configuration in the hope of improving system performance. In some cases, the hardware expenditure produces a satisfactory improvement in performance. Unfortunately, in many cases we do not see the expected return on our investments.

This paper identifies and discusses the various performance problems that can be related to hardware including raw CPU power, availability of main memory and disc subsystem throughput. Methods are introduced that help you to identify hardware related problems. Alternative solutions are presented where they exist and the probability of success is discussed in each case.

An on-going methodology for preventative or at least predictive maintenance of hardware related performance is introduced as well as a feedback mechanism to report on the effectiveness of actions taken.

ABOUT THE AUTHOR

Brian Duncombe is the founder of Triolet Systems, a company that specializes in performance related consulting, training and software tools for the HP3000 environment. As a former System and Performance Specialist with Hewlett-Packard in Canada, Brian brings with him a broad knowledge of the subject area as well as more than 20 years experience in the data processing industry.

HISTORICALLY SPEAKING

While the subject of system performance has become a popular topic for widespread discussion within the HP3000 community in the last year or two it is a subject that has always been part of the dialog when users meet to talk shop.

The rapid advances in hardware processing capabilities within the HP3000 product line have tended to diminish the attention spent on optimizing the throughput of computer systems. There have, of course, always been users who have had to carefully justify every dollar spent and therefore wring every possible ounce of performance from their systems. Most installations, however, have been battling a shortage of talented manpower combined with an abundance of processing power there for the buying and have focused their attention on people productivity.

As a former HP Performance Specialist and now an independent consultant, I can speak from first hand experience about the problems encountered when trying to make recommendations based on a very limited view of the problem. I received my initial performance training with Hewlett-Packard during the days when the solution to most problems was "take two megabytes and call me in the morning". While this often turned out to be good advice, I remember customers who paid dearly for the memory upgrades on Series II and III's only to realize little or no improvement in the throughput of their particular systems. Not only does the hardware vendor look bad in the eyes of the customer but the customer has to face his users who were probably active in the campaign to obtain the budget for the upgrade. This problem has certainly not gone away in recent times. On the contrary, we now upgrade a Series /48 to a /58 or /70 only to discover that we have increased the percentage of time that the central processor sits waiting for the disc subsystem or we add disc caching to a Series /44 only to see the system actually slow down under the increased CPU loading.

HOW CAN WE MEASURE PERFORMANCE

Performance is quite easy to measure. The Hewlett-Packard motto that "Performance is measured by results" is probably one of the most concise definitions that I have come across. From the pragmatic point of view, only response time and batch throughput define the level of performance of a given system. The sophisticated tools that are frequently flaunted as performance measurement tools are for the most part windows into the utilization levels of the various system resources and not performance monitors at all.

The measurement of system performance is one endevour in which the principles of "Murphy's Laws" are demonstrated admirably. In my consulting practice, it is very common to spend a day watching a system that is suppose to be experiencing performance problems only to see a system that in my view is performing excellently. In some cases, this is a matter of customer expectations compared to my experience of what would be usual in the situation but in many cases the problems are reluctant to manifest themselves while the system is under scrutiny. The system that runs well until the consultant goes for coffee or

TRIOLET SYSTEMS INC.
(416) 255-5767

lunch is the bane of the industry. Only an on-going project to characterize performance over long periods of time seems to be reliable in measuring system performance with a great deal of accuracy.

In measuring system response time, we are faced with a number of challenges. The very definition of response time and what is a transaction is heavily influenced by the specific application. Response time is usually defined as the delay between hitting RETURN or ENTER after entering the specifics of the transaction and the time when we can begin to enter a new transaction. While this seems straightforward, in many cases it is anything but. We can stand behind someone operating a terminal and measure their response times with a stop watch but this can only be done periodically and will often intimidate the operator thus skewing the results. We can use some sort of generalized monitoring tool but while it provides precise measurements, the lack of knowledge of what constitutes a "real" transaction poses a serious limitation on the accuracy and relavence of the data collected. Many software systems perform multiple terminal reads for a single logical transaction thus corrupting the results obtained from a software tool that simply uses terminal read completions as a delimiter for transactions. The most accurate measure of response time is achieved by instrumentation within the application code itself since at this level, the specifics of transaction definition can be customized as required. Most people would find it difficult to justify the expense of instrumenting existing code although it is certainly a good thought for inclusion as code is being developed. The case of package software and the lack of source code precludes this approach for a great many applications.

The measurment of batch throughput is much simpler. Here we have the case where we must determine if the results are being achieved in a reasonable amount of time. It doesn't matter if the job runs for many hours as long as it is finished by the time the results are required. In measuring batch performance we are simply checking to see if jobs are complete before the users require the output. This can be measured by the frequency of complaints from the user community about the timeliness of batch output. A secondary input is the computer operations area which must have sufficient time to perform backups and system maintenance as well as get all "exclusive" processing done during the periods when interactive users are not competing for the system.

Of course, it would be helpful to detect a pending problem early enough to react to it and prevent it from becoming critical. The tracking of trends must also be part of the mandate of system performance measurement.

WHY DOES PERFORMANCE VARY

The variation in system performance between one system and another and indeed between one time and another is regulated by a simple equation of supply and demand.

On the supply side, we have the finite availability of three resources. The first and most obvious of these resources is the rate at which the central processor (CPU) is capable of

executing instructions. The second and no less important resource is the ability of the memory subsystems to handle the demands placed upon them. This memory resource includes the actual volume of main memory available as well as the ability of the peripheral devices to transfer storage between main memory and the external media of the physical device as required. The third and final resource is the time available to perform the required processing. This time resource involves the problems of scheduling and load balancing that can play such an important role in the overall satisfaction of the system as a whole.

The demand side of the equation appears more complex on the surface but in fact is just as simple to understand. The basic definition of "processing" involves the aquisition of data from one or more sources, the manipulation of that data according to some algorithm and the storage of the resulting data. This simple model thus requires several distinct services from the computer system. The first requirement is to move data into a form and medium where the computer can perform manipulations on it. The manipulation of the data aquired then involves machine readable instructions that define the algorithms that will be used to solve the problems at hand. These instructions must be deciphered and carried out by the central processor in the course of performing the task. Once the data has been transformed by the processing algorithms, it must be stored into some medium more suitable for either human review or more cost effective longer term storage. The final demand is of course the timeliness of the performance of this processing.

As you can see, the supplies and demands involved within a computer system parallel one another very directly. This would seem reasonable since the computer system is an attempt to provide a more effective means of performing the data processing function and should therefore address the needs directly.

HOW CAN HARDWARE AFFECT PERFORMANCE

Hardware can only affect the supply side of the performance equation.

By providing a central processing unit that is capable of executing more instructions in a given amount of time, there is more potential for being able to service the algorithm related demands of the processing.

By making more main memory available within the system, the virtual memory features of the operating system can keep more of the active storage available for reference. By maintaining more of the problem in the higher speed types of memory such as main memory, the references to the data being processed as well as the instructions that must be executed as part of the processing algorithm can be executed more quickly.

Since a disc device has a certain throughput capacity, by having multiple disc devices you have the ability to service disc access requests in parallel. The same can be said for tape drives and printing devices.

When you attach several peripheral devices to the system using the same cable, controller, channel or CPU bus, common sense dictates that at some point the sharing of the connection will involve contention for the data path. The addition of device controllers (older disc drive models), GICs and IMBs provides a hardware solution to the delays caused by data path contention.

The upgrading of data communication facilities will provide additional capacity for communicating between devices. In the case of line speed upgrades, the data can be transmitted faster thus reducing the transmission time component of the performance delay. Where the existing facility involves sharing the data path amongst several users, the upgrade from a multiplexor or multi-drop network to dedicated lines or channels will obviously decrease the chances of contention for the supply of that resource.

WHEN DOES HARDWARE NOT EFFECT PERFORMANCE

A solution is only effective if it addresses the problem. As I learned during my days in an IBM environment, the objective is not to find a faster swapping device but rather to stop swapping.

The introduction of an additional hardware resource within the computer system can only be effective in improving system performance if that hardware provides additional capacity of the specific system resource that is currently the critical one. Of course, it is true that almost any hardware addition will usually yield some improvement in system performance but that increase will only be significant if it relieves the current throughput bottleneck.

The upgrade from one CPU model to a faster member of the family will not have an appreciable effect on system performance if the primary cause of processing delays is the ability of the disc subsystem to move data between main memory and the physical disc media. A particular disc drive model is not any faster simply because it is attached to a faster CPU. There will be a minor improvement in throughput because the faster CPU can reduce the fraction of the total disc access time caused by the processing required to initiate and then complete the software component of the total disc access but this improvement will be minor. Again, the faster CPU can search main memory disc cache faster and possibly improve the effective rate of the disc subsystem but this will also be fairly minor in most cases. Similarly, a faster CPU does not address the performance delays caused by a shortage of available main memory.

The addition of more disc devices will certainly solve problems associated with a shortage of disc storage capacity. In most cases, the spreading of disc accessing will also reduce the accessing times associated with physically moving the read/write heads across the surface of the disc media. This improvement will be relatively minor if the most serious resource shortage is raw CPU power. If the critical resource is the lack of sufficient main memory to satisfy the demands of the load on the system then the addition of more disc devices may have some effect on the speed with which memory can be swapped between main memory and disc memory although the addition of main memory would be much more effective in yielding improvements.

By adding additional disc controllers, GICs and IMBs where possible, we can provide an opportunity for parallelism in the transfer of data between main memory and the physical storage devices. This will only yield noticeable results if the workload on the system is capable of generating a backlog of requests on the devices. In the case of everyone accessing a single IMAGE database, the software has very little potential for generating more than one disc request at a time and even though disc throughput capacity may be the current bottleneck, there is seldom a time when requests queue at the devices and could be serviced in parallel even if the hardware paths were provided.

With the advent of third-party memory suppliers and more aggressive pricing and marketing, adding additional main memory has become the rage. With the popularity of main memory disc caching, the addition of main memory resource will most likely yield at least some noticable results. The point to be remembered is that the results achieved by directly addressing the primary resource bottleneck will almost certainly be rewarded with more improvement.

MAXIMIZING RETURN ON INVESTMENT

It is very unlikely that by spending money on a hardware upgrade the performance of the system will actually suffer. The few instances of this happening are usually associated with

some form of temporary imbalance in the distribution of the workload amongst available resources.

It is not at all uncommon to spend a large sum of money on hardware upgrades and experience only a very small improvement in performance of the system. As we have seen earlier, adding capacity in an area that is not a critical resource will not yield very noticeable benefits.

In order to maximize the effect of hardware changes, you must add those additional resources in the area of the critical resource. To do this, you must know what the critical resource is. As we have seen, the critical resources that can be addressed by hardware solutions are the capacity of the CPU to process instructions, the availability of sufficient main memory to minimize swapping between main memory and disc storage and the throughput capacity of the disc subsystem.

In order to determine if the CPU is the critical resource, we must simply determine the overall utilization level of the CPU over a period of time. There are a number of software tools available that allow you to see how busy the CPU is within your computer system. These tools include both contributed library programs such as SOO and SURVEYOR as well as supported products such as OPT/3000 from Hewlett-Packard and PROBE/3000 from Strategic Systems in Seattle, WA. No matter which tool you use, the objective is to determine how much of the total CPU resource is actually being utilized. If the CPU is being utilized less than about 75% of its capacity, it is not likely to be the major bottleneck within the system and upgrading the CPU is not going to be a wise use of upgrade money. If the CPU utilization is greater than 75% and you are not using disc caching, the addition of a faster CPU would surely yield noticeable improvements in system performance. If the utilization is above 75% but disc caching is active then the addition of a faster CPU should speed up the system although if the utilization of the CPU for disc caching is greater than 10% or 15% then it is possible that a faster CPU would be forced to wait for physical disc accessing and the increase in speed not fully taken advantage of.

The determination of disc bottlenecks is fairly simple. If the disc drives are vibrating noticeably or the access lights are constantly illuminated, then it is a good bet that the disc subsystem is a major contributor to performance delays. Most of the software tools mentioned previously provide at least some metrics of the utilization levels of the disc subsystem components. The question of whether some form of hardware upgrade would be effective is not nearly so simple. The hardware alternatives that address disc bottlenecks take one of two forms.

In the first case, you can add one or more additional disc drives to the system. This increase in the number of disc devices, in addition to yielding more storage capacity, provides an opportunity to spread the accessing load over more physical devices. Since a single disc device can accomodate from 20 to 30 physical accesses per second, by providing multiple devices, you can increase the theoretical maximum throughput capacity. In addition, by spreading accessing over multiple devices, the probability of frequently accessing files on the same device is reduced and the requirements for long access head movements (seeks) should be reduced. Of course, multiple devices also provide the opportunity for parallelism

within the disc subsystem. In the case of a system with a single disc drive, by adding another device and spreading the load fairly evenly between them, you can gain noticable improvements in system performance. Of course this is assuming that the disc subsystem is the primary cause of delays. As the number of disc devices on the system increases, the advantage of adding another device decreases very rapidly. In my experience, it is not worth adding a disc device simply to distribute the workload once you have three disc devices.

The second opportunity to relieve disc subsystem bottlenecks using hardware solutions involves providing more parallelism in the data paths. This can take the form of upgrading "slave" devices to masters in the case of the older devices, increasing the number of GIC interfaces that have disc controllers attached to them and in the case of the 6x and 7x CPUs adding additional IMB busses. Before contemplating this type of hardware upgrade, you should determine if this type of improvement would help. In order to see noticable gains in throughput, there must be a backlog of requests for the disc devices so that there will be an opportunity to actually do disc accessing in parallel. Both the OPT/3000 and PROBE/3000 software tools provide for monitoring the queueing of requests on the various disc devices as well as the accessing rates. If there is usually a queue of requests for each disc drive and the disc drives are being accessed at a rate below their rated capacity then adding data paths will usually improve performance. If the drives are already running near capacity (20+ accesses per second) then adding additional paths will not likely improve the situation although adding disc devices would be a reasonable improvement. Once you decide that the system could benefit from adding data paths, the question is what to add. If you have the older 792x disc drives and they are a mixture of master and slave configurations then upgrading the slaves to masters should be the first step. The addition of more GIC interfaces with disc devices connected would be the next alternative. There is a limitation within the hardware that only two GICs per CPU bus are allowed to have high speed devices such as discs attached. Since the Series 4x and 5x processors are limited to a single CPU bus, they are also limited to two "high speed" GICs. For any number of reasons, many systems have been installed with all disc devices connected to one GIC and a tape drive connected to the other high speed GIC. While this is necessary for 7970 tape drives, for 7974, 7976 and 7978 tapes the only logic is to provide better channel balance during backups. While the backup may go very slightly faster in this configuration, normal disc accessing suffers. If your system has all discs on one GIC and a 7974, 7976 or 7978 tape on another by itself, you would be wise to move half the disc devices to the same GIC as the tape. If your system does not have 2 high speed GICs per channel and you have more than two discs on a GIC device, a relatively low cost upgrade would be to add an additional GIC to the system. In my experience, there is no noticable improvement in doing this unless you have more than three disc devices on a particular GIC. For the 6x and 7x processors the option exists to add one or more additional IMB busses. This upgrade is fairly expensive and unless you have another reason for doing so, I don't think this upgrade justifies the small performance improvement that might be gained. Of course, if you have multiple IMBs then you should spread your disc drives out on additional GICs in the additional IMBs.

The question of whether or not to add main memory capacity is often the most difficult to answer. In the case of serious memory shortages, the various software monitoring tools will all show excessive memory manager activities that can be seen fairly clearly. In these obvious cases, additional memory is a very wise investment. The other side of the question

is also relatively simple to answer. If the monitoring tools do not show any appreciable memory manager activities then the addition of main memory is likely not a good investment. The middle ground is the difficult one to cover. If you are using disc caching then a simple experiment is to turn it off and monitor the system for a while. If this step improves performance and you were not previously experiencing a CPU bottleneck problem then the main memory freed up by the removal of the caching overhead has probably relieved a memory shortage problem. In this case, the addition of main memory is a good investment.

DEVELOPING AN ON-GOING MONITORING PLAN

As was mentioned previously, short term studies of system performance are often not representative of the long term picture. To get an indication of the longer term performance issues you must really conduct an ongoing study. This can usually be done by using such available tools as HPTREND and the logging facilities of the various monitoring tools. What you should be trying to do is present a picture of the longer term utilization trends of the various system resources that affect performance. By doing this data gathering and then reducing the data into some simple charts or graphs, you can get a very good feeling of how the various resources are being utilized over a day, week, month or longer. Once you are in control of this information, you have the opportunity to spot changes in trends and act upon those changes before they become real problems.

Winning with MPE/XL

by
David Elward

MPE/XL Overview

What is MPE/XL? MPE/XL is the operating system on Hewlett-Packard's new line of HP Precision Architecture computers. MPE/XL is similar to MPE/V with many enhancements. MPE/XL provides full compatibility for almost all MPE/V commands, User-Define Commands (UDCs) and job streams

The first thing you'll notice when you sit down at a terminal connected to an HP/3000 Series 930 or 950 is that when you hit return, it doesn't print a colon prompt but prints "MPE/XL:" instead. This immediately tells you that you aren't on good ol' MPE/V anymore. But once you enter your HELLO command and get logged on, things seem back to normal. In fact, from this point on you probably won't notice any differences between MPE/V and MPE/XL. This should make you much more comfortable.

New Syntax

When trying to learn one of the new commands, you will note that there is a new feel to MPE/XL commands. The new feel comes from the new scanner/parser that has been implemented in MPE/XL. On MPE/V, most of the commands are parsed by the MYCOMMAND intrinsic, but on MPE/XL most of the commands are parsed by the new scanner/parser. Under MPE/V, all command parameters must be either positional or keyworded depending upon the exact syntax of the command. However, on MPE/XL, parameters may be entered as *either* positional or keyworded and the new scanner/parser will figure out what you mean. This is a much more flexible approach than MPE/V, but it does have the drawback of being incompatible with MPE/V. For example, on MPE/XL there is a new command called COPY that does a fast copy of disc files. The syntax of the COPY command is:

```
COPY FROM=<fromfile>;TO=<tofile>
```

The following are equivalent ways to enter a COPY command:

```
1.    COPY FROM=MYFILE;TO=YOURFILE
2.    COPY MYFILE,YOURFILE
3.    COPY TO=YOURFILE;FROM=MYFILE
4.    COPY MYFILE;TO=YOURFILE
5.    COPY MYFILE,TO=YOURFILE
```

The following ways of entering the COPY command will produce an error.

```
1.    COPY MYFILE;YOURFILE
2.    COPY MYFILE YOURFILE
3.    COPY FROM=MYFILE,TOFILE
```

If you scrutinize the examples, you will notice that keywords always work, no matter what type of delimiter you have between parameters. Parameters may only be used positionally when

Variable Substitution

Variables can be used in commands in exactly the same way as UDC parameters can be used in MPE/V and MPE/XL. That is, by putting an exclamation point in front of the variable name. This will cause the variable name to be replaced in the command line with the variable value. There is a new MPE/XL command called :ECHO that we will use to show some examples. The :ECHO command just prints everything after it. It is like the :COMMENT command except that it prints the comment. I'm sure many of you have seen "OPTION LIST" UDCs print things using the :COMMENT command. Some examples of the :ECHO command and variable substitution:

```
:ECHO My name is Dave.
My name is Dave.
:SETVAR NAME, 'Dave'
ECHO My name is !name.
My name is Dave.
:ECHO I am logged on as !hpjobname,!hpuser.!hpaccount.
I am logged on as DAVE,MYUSER.MYACCT.
```

There are two little known facts about MPE/V UDC parameter substitution that also apply to MPE/XL variable substitution:

1. You may use a double exclamation point (!!) if you do NOT wish to substitute a variable. When you do this the two exclamation points are turned into ONE exclamation point, and the variable is NOT substituted.

2. You may put variable name in quotation marks if you wish to follow the variable name with alphabetic characters. This allows you to specify when the variable name stops and the trailing characters begin.

Examples:

```
:SETVAR NAME, 'Dave'
:ECHO My name is !!name
My name is !name
:SETVAR NAMENAME, "!!NAME"
:ECHO My name is !namename.
My name is Dave.
:SHOWVAR NAME@
NAME = Dave
NAMENAME = !name
:SETVAR NAME, 'Jose'
ECHO My name is not !namename.
My name is not Jose.
```

Changing Your Prompt

On MPE/XL, there is variable called HPPROMPT that contains the value of the prompt character. When you logon, MPE/XL sets this to ":". If you wish, you can change it to something else. Some examples:

```
:SETVAR HPPROMPT, '>'
>SETVAR HPPROMPT, '!HPUSER.!HPACCOUNT:'
MYUSER.MYACCT:SHOWVAR HPPROMPT
HPPROMPT = MYUSER.MYACCT:
MYUSER.MYACCT:SETVAR HPPROMPT, '(!HPCMDNUM) '
(5) SHOWVAR HPPROMPT
HPPROMPT = (5)
(5) SETVAR HPPROMPT, '(!!HPCMDNUM) '
(7) SHOWVAR HPPPROMPT
HPPROMPT = (!HPCMDNUM)
(8) SHOWTIME
SUN, AUG  9, 1987, 4:16 PM
(9) SETVAR HPPROMPT, '[!!HPHOUR:!!HPMINUTE] '
[15:18] SHOWVAR HPPROMPT
HPPROMPT = [!HPHOUR:!HPMINUTE]
```

Now you know almost everything there is to know about variables and variable substitution on MPE/XL, almost. MPE/XL also lets you substitute expressions as well as variables. What you do is simply put your expression inside of square brackets ([]) and precede it with a bang sign (!). Some examples:

```
:ECHO Two plus two equals ![2+3].
Two plus three equals 5.
:ECHO It is time to go home: ![HPHOUR >= 17]
It is time to go home: FALSE
```

Enhanced :REDO Command

A redo stack has been added to MPE/XL that keeps track of the most recent commands you have entered. Any one of the commands in the redo stack may be redone simply by entering the command number of the command you wish to redo as part of the :REDO command. This feature is really nice for people like me who have a habit of typing "REOD" instead of "REDO". To figure out what's in your redo stack, MPE/XL has provided the :LISTREDO command. This command lists in the order you entered them, the entire redo stack. Let's look at some examples:

```
:HELLO DAVE, MYUSER.MYACCT
MPE/XL ....
:SETVAR HPPROMPT, '[!HPCMDNUM] '
[2] SHOWTIME
SUN, AUG  9, 1987, 10:12 AM
[3] ECHO The time are !hphour:!hpminute
The time are 10:12
[4] REDO
ECHO The time are !hphour:!hpminute
```

```
                dddiis
ECHO  The time is !hphour:!hpminute

[4] ECHO  The time is !hphour:!hpminute
The time is 10:13
[5] LISTREDO
       1) SETVAR HPPROMPT, '[!!HPCMDNUM]
       2) SHOWTIME
       3) ECHO The time are !hphour:!hpminute
       4) ECHO The time is hphour:!hpminute
       5) LISTREDO
[6] REDO 1
SETVAR HPPROMPT, "[!!HPCMDNUM] "
                  DDDDDDDDDDDDI:
SETVAR HPPROMPT, ":"

[6] SETVAR HPPROMPT, ":"
:DO 2
:SHOWTIME
 SUN, AUG  9, 1987, 10:14 AM
```

What's that :DO thing there? That :DO thing is another MPE/XL command that operates exactly like the :REDO command, except that you don't get a chance to edit the command. It just does it. When :DOing or :REDOing a command from within the redo stack, there are three ways to access the command:

When requesting to :DO or :REDO commands, there are three ways that you can specify which command you want to :DO or :REDO.

1. You can specify the absolute command number. You can find out what this is by using the :LISTREDO command. The number in front each command is the absolute command number.

2. You can specify the relative command number. A relative command number of -1 means the previous command, a relative command number of -4 means 4 commands previous. When you enter :DO or :REDO with no parameters, a default relative command number -1 is used. The :LISTREDO command with the ;REL parameter may be used to display the relative command numbers of all the commands in your redo stack. It looks like this:
   :LISTREDO;REL

3. If you specify a string instead of a number as the :DO or :REDO parameter, MPE/XL will search backward through your redo stack looking for a command that matches the string.

Some examples. For all of the examples let's assume that are redo stack looks like this:

```
1) SHOWJOB
2) PURGE MYFILE
3) FCOPY FROM=NEWDATA;TO=DATAFILE;NEW
4) FILE OUT;DEV=LP,2
5) RUN BIGPROG.PUB.BIGACCT
6) SHOWTIME
```

Command	Result
:DO 4	DOes command 4
:DO	DOes command 6
:REDO r	REDOes command 5
:DO -4	DOes command 3
:REDO SHOW	REDOes command 6
:DO "FCOPY F"	DOes command 3
:Do h	Error, command cannot be found
:REDO -7	Error, stack only contains last 6 commands
:DO showj	DOes command 1

Redo Substitution

Commands from the redo stack may be substituted into commands as though they were variables by putting an exclamation point (!) in front of the command number. Using the sample redo stack above:

```
:ECHO My most recent command was "!-1".
My most recent command was "SHOWTIME".

:ECHO My first 2 commands: !1 & !2.
My first 2 commands: SHOWTIME & PURGE MYFILE.

:!4 ;NOCCTL
:ECHO -1
FILE OUT;DEV=LP,2;NOCCTL
```

:WHILE Command

The :WHILE command is a new MPE/XL command that is very similar to the :IF command, except that it repeats a group of commands instead of only executing them once. The :WHILE command is based upon the WHILE statement of many programming languages. A :WHILE loop is terminated by an :ENDWHILE command.

The following example will purge all one hundred of the work files named WORK00 through WORK99:

```
:SETVAR COUNTER, 0
:WHILE COUNTER <= 99
:   SETVAR FNUM, "!COUNTER"
:   IF COUNTER <= 9 THEN
```

```
:     SETVAR FNUM, "0"+FNUM
:   ENDIF
:   SETVAR FNAME, "WORK"+FNUM
:   ECHO Purging file !FNAME
:   PURGE !FNAME
:   SETVAR COUNTER, COUNTER + 1
:ENDWHILE
```

Command Files

Perhaps the interesting feature of MPE/XL is the addition of command files. Command files are similar to User Defined Commands except they don't need to be catalogued. A command file is simply a file like a job stream that contains MPE/XL commands. The user may execute the job file by entering the :XEQ command followed by the file name. Example:

```
:EDITOR
HP...
/ADD
    1    SHOWTIME
    2    ECHO This is my command file.
    3.   ECHO I like command files.
    4    //
/KEEP MYXEQ
/EXIT

END OF SUBSYSTEM
:XEQ MYXEQ
SUN, AUG  9, 1987, 2:27 PM
This is my command file.
I like command files.
```

The Implicit XEQ

The best thing about the XEQ command is that you don't need it! Any time you enter a command that MPE/XL doesn't recognize, it will *assume* an :XEQ command. In the above example, only ":MYXEQ" needs to be entered, not ":XEQ MYXEQ". Another feature of the :XEQ command is that it can also be used to run programs. Let's look at some examples:

:MYXEQ	Will execute command file MYXEQ.
:XEQ SPOOK5.PUB.SYS	Will run SPOOK.PUB.SYS
:MYPROG	Will run program called MYPROG.
:DBUTIL.PUB.SYS	Will run DBUTIL.PUB.SYS

HPPATH

One of the interesting features of the :XEQ command is that it doesn't look in only one place for the file. By default, all :XEQ

commands first search your logon group for the name of the file to be executed. If it doesn't find an executable file in your logon group, it will search the PUB group of your logon account, and if it still doesn't find an executable file, it will look in the group PUB.SYS. This searching order is controlled by the variable HPPATH. The default value of HPPATH is "!hpgroup,pub,pub.sys". What all this means is that you don't need all those UDCs you have to run programs in PUB.SYS! Look at the following example:

```
:KSAMUTIL
...Runs KSAMUTIL.PUB.SYS

:PAYROLL
...Runs the PAYROLL program in the PUB group

:FILEEQS
...Executes the command file in my group called FILEEQS

:SETVAR HPPATH,"!!hpgroup,prog.lib"
:SHOWVAR HPPATH
HPPATH = "!hpgroup,prog.lib"
:XREF
...runs the program called XREF.PROG.LIB
:DBUTIL
...Error because the PUB.SYS group is not searched
```

Parameters

Parameters may be passed to command files in a manner very similar to UDC files. The parameters are passed by having the very first line of the command file be a PARM command that describes the parameters of the command file. The parameters are described in exactly the same as they are in UDC files. Look at this example:

```
:EDITOR
HP...
/ADD
    1    PARM ERRNUM
    2    ECHO CI error !ERRNUM is:
    3    SETVAR TEMP, CIERROR
    4    SETVAR CIERROR,!ERRNUM
    5    ECHO !HPCIERRMSG
    6    SETVAR CIERROR, TEMP
    7    DELETEVAR TEMP
    8    //
/KEEP ERRMSG
/EXIT

END OF SUBSYSTEM
:ERRMSG 975
CI error 975 is:
UNKNOWN COMMAND NAME.   (CIERR 975)
```

```
:XEQ ERRMSG 629
:CI error 629 is:
OUT OF PCB RESOURCES.  (CIERR 629)
```

This example uses the PARM line to accept the value of the ERRNUM parameter. Since ERRNUM is a required parameter because it does not have a default value, an error will occur if the user does not enter an error number. The variable HPCIERRMSG is a system variable that contains the CI error message based upon the CIERROR JCW. The TEMP variable is used so that the previous value of CIERROR is not destroyed by this command file.

Other Commands

The :PRINT command is a command that allows the user to print a file to either the terminal or an output file such as the line printer. The nice thing about the :PRINT command is that you don't have to take the time to enter an editor, text, and list the file just see what's in it. When printing to your terminal, :PRINT stops after every page and gives you the opportunity to continue printing, stop printing, or continue printing from any record in the file.

The :CHGROUP command is a very convenient command that allows the user to change the group he is currently logged in to. It is functionally equivalent to logging on again to the new group, however it is much quicker and doesn't destroy your temp files and file equations. Of course, if the new group has a password, the user will be required to supply one. If the :CHGROUP command is entered without any parameters, then the current group will be changed to the user's home group.

The :INPUT command is similar to the :SETVAR command, except that it is used to prompt the user for the value of a variable. It could be used to prompt the user for a password or lockword within a command file or UDC. A timeout is available on the :INPUT command so that the user doesn't take too long answering.

The :COPY command was mentioned earlier. It is used to perform a very fast copy of an old file to a new one. It is considerably faster than FCOPY.PUB.SYS.

Miscellaneous

The above discussion includes *most* of the new features of MPE/XL. There are several new commands used for managing private volumes that replace existing MPE/V commands. Many existing MPE/V commands have been slightly modified to conform to the MPE/XL environment.

Summary	Many new features have been added to MPE/V to create MPE/XL. These features combine to form a very powerful set of commands and an enhanced enhanced environment. MPE/XL is certain to become widely used and highly regarded by the HP/3000 user community.

Controlling Your Production System's Environment

by

David L. Fastenow
Collins Air Transport Division
Rockwell International Corporation
400 Collins Road NE
Cedar Rapids, Iowa 52498

Controlling Your Production System's Environment

by
David L. Fastenow
Collins Air Transport Division
Rockwell International Corporation
400 Collins Road NE
Cedar Rapids, Iowa 52498

Table of Contents

Appendices

Controlling Your Production System's Environment

by
David L. Fastenow
Collins Air Transport Division
Rockwell International Corporation
400 Collins Road NE
Cedar Rapids, Iowa 52498

Achieving complete control over the environment is essential to the security and integrity of any system. With today's integrated systems, this is particularly true because of their diversity and the variety of different users throughout the company. With the very "lifeblood" of the company comprising the system's databases, the integrity and security of the system is one of the most crucial issues for a company to deal with. Yet, it is surprising how many companies do not have the proper control over their systems environment. The key to this control is discipline. Webster defines descipline several different ways. Two definitions apply to the control of your production system's environment:

1. training that develops self-control, character, or orderliness and efficiency
2. a system of rules and methods.

Studying the definition causes some interesting observations. First, control does not just happen. It has to be carefully set up with the end objectives of complete security and integrity in mind. Training of personnel is needed to insure that the discipline is carried out. Although a lot of hard work goes into establishing the system of rules and methods, a very interesting phenomenon occurs once that discipline is in place. Namely, self-control, orderliness and efficiency! The control will come naturally and will generate a smooth orderliness and increased efficiency in the total environment. Therefore, not only is integrity and security achieved, but also an efficient and smooth running operation.

If all these benefits are there, why don't all companies implement the necessary control over their systems? There are many reasons why, but it all boils down to two facts:

1. many companies do not know what they need for effective control, and
2. it is a lot of work to establish control.

The purpose of this paper it to outline what we at the Collins Air Transport Division of Rockwell International have done to achieve effective control and how you can simplify the establishment of the necessary disciplines. The paper is broken into six sections:

1. Operating System Control and Security
2. User Control and Security
3. Production System Procedures
4. New Application Package Releases
5. Ongoing Program Changes/Enhancements
6. Jobs and Work Packages

Each section is presented as generically as possible. Specific examples are shown in the appendices, however, to clarify the discussion and show how Rockwell's Air Transport Division has implemented the disciplines.

The installation of the proper disciplines in each section is essential for total control over your production system's environment.

I. Operating System Control and Security

Without Operating System control and security, all the other control procedures we could implement would be to no avail. This level of control defines the system as a whole and establishes certain functions that are needed for separation of duties.

From the personnel perspective, different job duties need to be defined:

1. Computer Services and Service Administration. This area of responsibility provides continuing operation of the computer system and operational support to the users. This function will also carry out certain relevant security procedures, such as system backup, and perform necessary preventative maintenance on computer hardware and peripherals.

2. Computer Technical Support. This area of responsibility encompasses the HP System Manager functions such as maintaining the operating system software, system recovery from failure and the development of necessary system routines and interfaces. This function must also carry out many of

the security procedures that define system security. This area is extremely critical and you must have complete confidence, faith and trust in the individual(s) assigned. You must realize that this person can, if he/she wanted to, circumvent any and all security on the system. The duties and related passwords of this area must be tightly guarded and controlled!

3. Security Officer. This area is responsible for the data security and integrity of the computer system. The Security Officer must insure all security procedures are followed and must review and approve all application systems for adequate security and integrity. This area also controls user access to all application systems.

4. Program Librarian. This area is responsible for the control, security and integrity of the Production Release Account and performs the actual release of all application system changes to the Production Release Account.

5. Documentation Analyst. This area is responsible for maintaining the proper documentation for all application systems. This includes insuring the programmers and analysts produce the proper documentation.

6. Application System Managers / Analysts / Programmers. This area is responsible for the installation, development, documentation and maintenance of application system software and for providing user assistance and support for the application systems.

These six areas do not imply that different individuals be assigned to them. You can combine several functions under one person. However, area 6, the Application Programming Area should be separate from any of the other areas. This is crucial in order to guarantee the required separation of duties required by auditors.

Now let's discuss the computer system organization. Of paramount importance is the separation of Production and Test. The production accounts are strictly controlled as they contain the "lifeblood" of the organization. Users only access the production accounts via released programs that perform specific functions. Application programming does not have access to production, except for well defined

and approved "read only" access. The Application System Manager may need a special category defined that allows the ability to correct problems with production databases, etc...but that ability must be as carefully controlled as that of the HP System Manager.

The Production Accounts are quite finite:

1. SYS, and any other related accounts, including operating software aids accounts (REGO, VESOFT, COGNOS, etc). Appropriate safeguards should be present to insure that only authorized people can log on into these accounts. Most of the miscellaneous accounts never need to be entered except whan a new software release is installed, so the CPU and CONNECT limits for the account can be set to zero to prohibit all logons to the account. MANAGER.SYS must have extremely well controlled passwords. Only one account should have SM (and preferably PM) capability and that is SYS.

2. The Production Release Account, called PROD in this paper. This account contains all internally developed source, including modified purchased package source, executable programs, job streams and other released files. The Program Librarian is normally the only user of the account. Related to this account are purchased software accounts containing all "vanilla" source and current executable programs.

3. The Production Database account (there could be several of these accounts). We call this account AT for Air Transport. All application production users log on to this account. The production data is separated into different groups in this account, depending upon the applications. The account manager, MANAGER.AT, is the Security Officer. All production batch jobs run under a unique user, in our case XMAPROD. All user security functions, such as registration and functional area maintenance are done by another unique user, in our case SECURITY.AT.

In general, all production accounts allow write access only to account users. Non-sensitive data will allow read access for ANY user in the system. Sensitive data, however, will have tighter control placed upon it. Any changes to a database done by means other than a released program must have an Update Log Approval form completed and signed. Appendix J has a sample of this form. This includes changes made by Query, Adager, QTP or any other unreleased program.

All of the other accounts in the system are classified as test accounts. Normally test accounts are used and maintained by programming personnel. Test accounts have security similar to the production accounts but may change according to the needs of the test account manager.

II. User Control and Security

All users are individually registered. See Appendix A for an example of Rockwell's registration form. Per Rockwell security policy, each user has a unique user ID (of the form XMAnnnn for Air Transport) and a private password maintained and known only by that user. The password must be between 6 and 8 characters long, must be changed at least every 60 days and must be one-way encrypted for complete privacy.

Since HP's standard security does not have those capabilities, we have developed our own user password security system, integrating it with our application systems. Although the changes we made to our application system's security scheme are not necessary for all organizations, they will be described because of the tremendous control they provide. The concept is analogous to the flexibility achieved with indirect addressing possible in some programming languages. Our Security database contains each user's ID and password, but does not directly reference the application commands that a user has access to. Instead, it indicates the Functional Area(s) that individual can access. For example, all the buyers may be in the functional area PURCH but their manager, due to his unique requirements, is in functional area PURCHMGR. Each functional area has the application commands associated with it that are needed for those users to perform their jobs. This mapping is achieved via a "mapping" database, where the functional area, obtained during logon, is mapped into the valid commands for that user.

If a user has access to more than one functional area, which may often be the case, a prompt is issued so the user can select the functional area to be used during that session.

The beauty of this approach becomes apparent when one considers the hundreds of users accessing your application but the relatively few functional areas needed to group them. Therefore, only a few static "functional area to

application command" mappings need to be defined. As users come and go in the job, the dynamic process of setting them up in their functional areas is easy.

Any changes to a functional area's application command mapping is handled through a special form (Appendix B) that requires the approval of all the major functional area managers. Thus, each area can be assured that some other area is not infringing on their responsibility, insuring the integrity of the database.

Reports are generated weekly that show:

1. All users, their user ID's and their related functional areas
2. All functional areas and the users in them
3. All functional areas and the application commands they map to
4. All application commands and the functional areas that have access to them.

The original approach for user access was to have them all log on the same user.account specifying their user ID as the session name:

```
:HELLO XMAnnnn,M.AT
```

M.AT has no HP password but is controlled via a LOGON NOBREAK UDC. The first program prompts for a password. If the user ID or password is invalid the session is BYE'd; otherwise the application is invoked. The problem encountered was that 1200 logons and logoffs per day consumed 20% or more of the HP3000/70.

The solution, and our current scheme, uses a menu processor developed in-house called FAST MANMAN (or FM for short). The benefits of FM are not only the elimination of 1200 sessions per day and the associated overhead, but also faster access to our applications, better response time and a great increase of control over the user.

Today, a user sits down at a terminal, presses RETURN and a menu instantly appears on the screen. One option is to Sign On, which must be done before any application can be accessed. Other options allow access to our applications and to Sign Off, which in our environment will break the Port Selector connection. Since no user actually "logs on", system security is also increased.

Perhaps the most significant aspect of FM, however, is the ability we now have to control the user. FM's layered menu system allows a user to access any of our applications. This control is just another level of indirect screen access that allows our environment to grow as needed without any constraints or complications to our user control and security.

III. Production System Procedures

Perhaps the single most important aspect of our control, from the user's perspective, is the production system procedures that have been developed. On the Computer Operation's side, a complete set of "Service Administration Procedures" have been created. Appendix C contains the Table of Contents for this 3 inch thick manual. Appendix D contains an example of one of the procedures. The manual covers all aspects of the Service Administration job, from user assistance to cool starting the system, from various preventive maintenance functions to using the HALON fire control system.

On the Computer Security side, a complete set of "Computer Security Procedures" have been developed. Appendix E contains the Table of Contents for this 2 inch thick manual. Appendix F contains an example of one of the procedures. This manual covers all aspects of our computer security, from computer room access to release procedures, from user registration to the nightly system backups and their storage requirements, from disaster recovery to report distribution control.

On the User side, a complete set of "Functional Area Procedures" have been developed. Appendix G contains the Table of Contents from four sections of this two volume manual. Appendix H contains an example of one of the procedures. This manual details all aspects of each user's job and their interaction with the application systems.

Although all of these procedures are continually kept up to date by individuals and committees, the Functional Area Procedure Committee is the most formal. This committee has delegates assigned to it from each functional area. They meet every Friday morning to review modifications and discuss needed changes. All procedure changes are approved by each of the major functional area managers. The members of this committee are responsible for making revisions to the procedures under their respective functional areas.

This is in fact a real opportunity for the committee members to really understand how the entire company works. As new enhancements and updates come along, this committee develops all the necessary user procedures to control the changing environment. This causes the users to really think about new enhancements and normally solve issues well before they become problems.

IV. New Application Package Releases

A new application package release is a lot of work. However, our policy is to keep current with the latest releases. Appendix I is the general procedure we follow to install a new release.

The first step is the examination of the new release and the changes that will affect us. Included here is a cursory review of the application program changes we have made and whether our changes still need to be incorporated in the new release. Unfortunately, they normally do.

The next step is to lay out a conversion plan and associated timeline. The timeline will be set up for the release to occur over the Christmas shutdown so adequate time is available for the database conversions that are normally required. This includes the tentative assignment of individuals to conversion tasks.

The conversion is a very time consuming process. Fortunately, the vast majority of our application programs have not been changed. Identifying the programs and the code within them that we have changed is an easy job. All changes go through a formal release process with a "changed program to vanilla program" compare filed with the changed source. Usually the determination of whether the change should be in the new release is easy, since most programs do not under go major modifications from release to release.

The largest part of the task is usually taking care of programs we have written. We have hundreds of QUIZ report programs and scores of Fortran programs. The changes needed to accommodate the new release will vary. Some may not change at all, some may need major modifications. For example, the next release of our manufacturing system, MANMAN, will probably have a great affect on our sophisticated Automated Purchase Order (APO) system. The conversion to the 9xx Spectrum machine will necessitate

converting all programs to Fortran 77 and making changes to take advantage of the optimization that machine will allow. Other programs we have written may require some changes where application package routines or databases are used.

As the programs are converted, unit testing is performed to insure the new code works. This testing is done by the individual programmer using an uncontrolled test database. When all programs have been converted, full system testing will begin using a very controlled test database. Users from all functional areas will be involved, especially members of the Functional Area Procedure Committee. It will be their responsibility to insure that all of their procedures are still accurate and the programs work as required. This testing will last approximately two weeks. Another week or two will be allowed for the Procedure Committee members to train their co-workers on the new application release.

By now it will be time for the Christmas shutdown, during which time the actual implementation will occur. Databases will be converted, UDC's will be released to access the application from the new account and all modified programs, jobs, etc. will be re-released per our release procedures. After all that is complete, checks will be made to insure everything is working. When the users come back after the shutdown they will have a new application release.

It is unfortunate that each new application release seems to add more inefficiencies to our system. We expect this next release to be more inefficient than ever. Before we can implement the new release we must have an HP3000/9xx installed. We plan on running production on the HP3000/70 up through conversion, using the 9xx purely for development until the new release is implemented. We anticipate the conversion preparation lasting about four months, giving us time to learn everything we need to about the 9xx machine.

V. Ongoing Program Changes

All program changes go through a formal release procedure before becoming effective. A program change will normally begin as a request from a user. This request is evaluated for feasibility and worthiness. If it passes those tests, a time estimate is made. Trivial requests (less than a couple of hours) will normally be done as part of our routine maintenance activity. All other requests will go

into our Project Tracking System as a formal Project. At that time, the analyst/programmer(s) will be assigned and the estimated start/completion dates will be determined based upon the current workload and the new projects priority. The Project Tracking System will track each project through to completion.

The actual release procedure is very formal, with absolute adherence to the rules. Prior to release, the analyst/programmer is responsible for having all testing and documentation completed. A release of a new program is not accepted without its documentation. The analyst/programmer creates a release notice that indicates to the program librarian exactly what must be done. The form must be signed by the Manager of Operations Services to indicate approval before the release action takes place. Appendix K contains a sample Program Release Notice and Appendix L is a page out of the detailed release procedures. The Program Librarian initiates the release by editing a canned job and submitting it. This job produces a "compare" of new versus old source code. This compare is filed with the new compile listing, which the job also produces, always compiling the program after the source has been moved to production. This program file resides in a special "pre-release" group until the compile is verified by the cognizant programmer. Then the final move is made to production. QUIZ programs, except those run online, are always kept in pure source form and are "compiled" by QUIZ at execution time. Therefore, those program releases are simpler. Appropriate procedures also cover RL and SL releases. Many application releases must be "finalized" during the nightly backup process as no user can be running the application.

VI. Jobs and Work Packages

All production batch work is done only through released jobs. The release is similiar to the QUIZ source releases, with the compare and the move of the job to the PROD account. A work package is the basic documentation we establish for each released job. The work package is used by the Service Administrators to help answer any questions they may have during the execution of the job. The work package consists of:

1. a description of the job
2. a flow chart depicting the programs and files used
3. special concerns or contingencies for the job
4. possible error messages and how to respond to them
5. any special restart instructions.

Each released job is also included on a "Cognizant List" that tells the Service Administrator who the cognizant and backup programmers are for the job. The priority of the job is also indicated: critical (call immediately if a problem occurs), important (call during normal waking hours) or minor (create a Trouble Report to be acted upon during the next normal working day). All problems cause a Trouble Report to be written for documentation of the problem and its resolution.

All released jobs are also scheduled in our Scheduling system and report distribution is set up in our Distribution database. These databases allow us to have good control over the production jobs and over the distribution of the reports.

The joblogs, as well as the reports, have the recipients of the reports listed. In addition, production jobs are stored in PROD without passwords and also have many special parameters embedded in them. This is made possible by a SUBMIT utility that modifies the job stream appropriately prior to streaming. This utility obtains information from the distribution database, the system and the Service Administrator, as needed.

As reports are broken down and distributed, the joblog is checked off, person by person, to insure all the correct individuals get the reports they requested. The joblogs are filed, providing an excellent audit trail of the reports produced and distributed.

Conclusion

This paper has presented the necessary ingredients for controlling your production system's environment. All of the pieces work together to achieve a total discipline. Without all the pieces, integrity is not there. Webster defines integrity as:

1. the quality or state of being complete
2. the quality or state of being unimpaired; perfect condition; soundness.

If integrity is what you want for your company's production environment, use the guidelines presented here to develop the necessary disciplines for Controlling Your Production System's Environment.

ROCKWELL INTERNATIONAL INFORMATION SYSTEMS CENTER

COMPUTER USER REGISTRATION FORM

DATE _____

USER PROFILE

NAME _____ SOC. SEC. NO. _____
Last First Initial

LOGONID _____ HOME DIV. _____ DEPT. _____ GROUP _____

LOCATION CODE _____ JOB FUNCTION CODE _____ TEMPORARY PASSWORD _____

PHONE: _____ INT. MAIL CODE: _____ COMP. MAIL CODE: _____

REGISTRATION REQUIREMENTS

	ADS	ATMS	BATCH	CBT	CDC	CICS	FOCUS	IMS	MKIV	NDS	ROSCOE	SHOP	SPERRY	SPY. CTS	TSO	VM/CMS	VSPC	AT HP
ECC					■								■	■		■		
MCC					■								■	■		■		
SCC	■	■	■	■		■	■	■	■	■	■	■			■		■	
SWC1					■								■	■		■		
SWC2					■								■	■		■		
WCC1					■								■	■		■		
WCC2					■								■	■		■		
WCC3					■								■	■		■		
WCC4	■	■	■	■	■	■	■	■	■	■	■	■	■			■		
CCM												■						

LOGONID ACTION TYPE

- ☐ NEW ID
- ☐ REASSIGN ID
- ☐ DELETE SERVICE
- ☐ DELETE ID
- ☐ CHANGE REGISTRATION (CIRCLE ITEMS TO BE CHANGED)

ROUTING: SOURCE _____ REMOTE DEST. _____

SPERRY PARAMETERS: C010 ☐ I80 ☐ W80 ☐ W132 ☐ JPL ☐ APL NO. _____

SPERRY CONSOLE PRIVILEGE: BASIC ☐ LIMITED ☐ JOB SUBMISSION PRIORITY LIMIT _____

ACCOUNTING: ONLINE STORAGE CHG. NO. _____ SPERRY ACCT. AUTH. _____

MAN/MAN FUNCTIONS _____ ACCT. REP. SIG. _____

SIGNATURES

I have read, understand, and will comply with Finance Policy 03:02:06, "Use of Computing Resources," and Finance Policy 03:02:03, "Computer-Based Systems — Access Control."

_____ MANAGER (DATE) _____ IS EXEC (DATE)

_____ USER (DATE) _____ LOCATION ISSO (DATE) _____ LOC CONTROLLER (DATE)

FORM SENT TO CENTER

_____ DATE _____ ISSO

DATE USER DATA BASE UPDATED/CSO INITIALS

ECC MCC SCC SWC1 SWC2 WCC1 WCC2 WCC3 WCC4

_____

114-A-30
074-8293-100 (Rev 11-84)

APPENDIX A

RETURN TO: LORI HUFF 108-247
X 3318

FUNCTIONAL AREA MAPPING

ORIGINATOR

ADD/DELETE A FUNCTION

FUNCTIONAL AREA NAME _____

☐ ADD A FUNCTION

☐ DELETE A FUNCTION

☐ ADD A COMMAND TO A FUNCTION

☐ DELETE A COMMAND FROM A FUNCTION

ADD COMMANDS	DELETE COMMANDS

REQUESTORS SIGNATURE _____ DATE

INSTRUCTIONS:

TO ADD OR DELETE A FUNCTION PUT THE FUNCTIONAL AREA NAME IN THE SPACE PROVIDED AND CHECK THE APPROPRIATE BOX. TO ADD OR DELETE A COMMAND PUT THE FUNCTIONS NAME IN THE SPACE PROVIDED, CHECK THE APPROPRIATE BOX AND LIST THE COMMANDS. THE REQUESTORS SIGNATURE IS NECESSARY AND IT SHOULD THEN BE RETURNED TO THE ADDRESSEE AT THE TOP OF THE FORM. IF THIS IS CONSIDERED AN URGENT REQUEST, THE SECURITY OFFICER WILL LIST THE AUTHORIZATIONS NECESSARY AND THE REQUESTOR MUST WALK IT THROUGH. OTHERWISE, THE NORMAL PROCEDURE WILL BE VIA INTERCOMPANY MAIL.

SERVICE ADMINISTRATOR

DEPARTMENT	AUTHORIZATION	APPROVED	DATE
ENGINEERING			
FINANCE			
MANUFACTURING			
MARKETING			
MATERIAL CONTROL			
OPERATIONS SERVICES			

RESPONSE TIME EFFECT IS NEGLIGIBLE ☐ _____ INITIAL _____ DATE

_____ REQUEST COMPLETED BY _____ DATE

ATCSF II.D.1

APPENDIX B

SERVICE ADMINISTRATORS PROCEDURES MANUAL

TABLE OF CONTENTS

APPENDIX C

Rockwell Air Transport Computer Room Service Administration Procedures

TITLE ATCRSAP Change Procedure
PROCEDURE NO ATCRSAP I A 1 00
PREPARED BY [illegible] Gonzicke

REVIEWED BY [signature] DATE 7-11-84
[signature] DATE 7/11/84
[signature] DATE 7/11/84
[signature] DATE 7/11/84
DATE

APPROVED BY DATE

1 PURPOSE

This policy and procedure defines the method for maintenance and distribution control of the Air Transport Computer Room Service Administration procedures

2 SCOPE

These policies and procedures apply to the Air Transport Service Administrators and the HP/3000 Computer Room.

3 REFERENCES

None

4 POLICY

a The aT Service Administrator and Security Officer are responsible for the control and distribution of procedures

b Periodic review of existing procedures should be accomplished to insure compatibility with system changes or restatements of policy

5 DEFINITIONS

SA – Air Transport HP/3000 Service Administrator
Manager – Manager of Operations Services, Air Transport Division
ATSO – Air Transport Security Officer

6 PROCEDURE FOR SECURITY PROCEDURE CONTROL

ITEM #	RESPONSIBLE	ACTION
1	Requestor	Requests to change existing procedures or to develop new procedures should be submitted directly to the SA
2	SA	After the modifications are made a "Documentation Change Notice" (ATCRSAF I A 1 1) must be completed and the entire procedure submitted to the Manager for approval.
3	Manager	Those packages not receiving approval will be returned to the SA for further action
4	Manager	Those packages receiving approval will be returned to the ATSO for documentation and distribution
5	ATSO	A global "Documentation Change Log" (ATCRSAF I A 1 2) will be maintained on file with the signed master copy of the procedures
6	ATSO	The last few digits of the procedure number are the revision numbers which are assigned to correspond with the change number on the Change Log. All procedures are initially revision 00. This change number must also be recorded on the Change Notice
7	ATSO	The Change Notices are filed in numerical order by procedure number behind the Change Log
8	SA/ATSO	A limited "Distribution List" (ATCRSAF I A 1 3) will be maintained by the SA/ATSO to insure distribution of procedures to key individuals only

APPENDIX D

Rockwell International
OPERATIONS CONTROLLED PRODUCTION DOCUMENTATION

1.03 NOTICE OF IMPLEMENTATION OR CHANGE No.

SYSTEM # SYSTEM TITLE
WORK PACKAGE # WORK PACKAGE TITLE
EFFECTIVE DATE PROCESSING TIMES HOURS REQUIRED BY SYSTEM

DESCRIPTION

CONTINUED

ENCLOSURES
PAGE NUMBERS
RUN SHEETS
JCL DECK
CONTROL CARDS
RETENTION DATA*
SCHEDULING DATA*
NONE OF ABOVE
NONE

DELETE OR REPLACE
PAGE NUMBERS
RUN SHEETS
JCL DECK
CONTROL CARDS
EMERGENCY CHANGE NO
NONE OF ABOVE
NONE

SIGNATURE PREPARED BY APPROVED BY CONCURRED BY
NAME TYPED

*REQUIRED ONLY WHEN INFORMATION HAS NOT BEEN SUBMITTED PRIOR TO ISSUANCE OF WORK PACKAGE

WORK PACKAGE NO. ATCRSAP I.A.1.1

Rockwell International
OPERATIONS CONTROLLED PRODUCTION DOCUMENTATION

WORK PACKAGE # ROCKWELL AIR TRANSPORT COMPUTER ROOM SERVICE ADMINISTRATION PROCEDURES (ATCRSAP)

1.0c NOTICES OF IMPLEMENTATION OR CHANGE LOG

CHANGE NUMBER	EFFECTIVE DATE	ENTERED BY SIGNATURE	PROC. NO.
00			
01			
02			
03			
04			
05			
06			
07			
08			
09			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			
32			
33			
34			
35			
36			
37			
38			
39			
40			
41			
42			
43			
44			
45			
46			
47			

CHANGE NUMBER	EFFECTIVE DATE	ENTERED BY SIGNATURE	PROC. NO.
48			
49			
50			
51			
52			
53			
54			
55			
56			
57			
58			
59			
60			
61			
62			
63			
64			
65			
66			
67			
68			
69			
70			
71			
72			
73			
74			
75			
76			
77			
78			
79			
80			
81			
82			
83			
84			
85			
86			
87			
88			
89			
90			
91			
92			
93			
94			
95			
96			

WORK PACKAGE NO. ATCRSAP I.A.1.2

Rockwell Air Transport Computer Security

Table Of Contents

APPENDIX E

Rockwell Air Transport Computer Security Policy

TITLE: AT HP/1000 Computer Room Access
PROCEDURE NO: ATCSF I A 00
PREPARED BY: D L Fastenow

REVIEWED BY: David R. Clymer (signature) DATE 6/8/84
DATE ________
DATE ________
DATE ________
DATE ________

APPROVED BY: ________ DATE ________

1 PURPOSE

This policy and procedure establishes the Air Transport HP/3000 Computer Room as an area of restricted physical access

2 SCOPE

This policy and its related procedures cover access to the Air Transport HP/3000 Computer Room

3 REFERENCES

a Corporate Computing Manual 125 22 07, Physical Protection of Resources Procedures

b Corporate Directive F-10, Computer and Data Security

c IL to J Wright - MANMAN Computer Room Access

4 POLICY

a The Air Transport HP/3000 Computer Room is considered an area of restricted physical access Only authorized personnel are permitted in the Computer Room without an escort Unauthorized persons must be admitted and escorted by authorized personnel in accordance with Reference c

b Access to the room is controlled by a combination lock with key override Distribution of keys to the Computer Room will be kept at an absolute minimum with a distribution list maintained by the Air Transport Security Officer (ATSO)

c Knowledge of the Computer Room combination is considered privileged information and may be disclosed only to authorized persons Authority is based on primary work station at the time, and/or on need to know as determined by the Air Transport ATSO The Air Transport ATSO will maintain a current list of authorized personnel

d The access combination on the HP/3000 Computer Room door will be changed every 60 days or when a person is removed from the authorized personnel list

5 DEFINITIONS/ABBREVIATIONS

SA - AT HP/1000 System Administrator
ATSO - The Air Transport Security Officer

HP/1000 Computer Room - The HP/1000 Computer Room based in Building 109, and the computers used for the MANMAN and TRACS Systems

6 PROCEDURE FOR MAINTENANCE OF COMPUTER ROOM KEY DISTRIBUTION LIST (REFERENCE 4 b)

ITEM #	RESPONSIBLE	ACTION
1	ATSO	Review Key Distribution List (ATCSF I A 1) every 60 days, or whenever a person on the distribution list changes function, for any potential changes
2	ATSO	If a person is to be removed from the Distribution List, obtain the key from said person
3	ATSO	If a person is to be assigned a new key, assign a key using an available key if one exists If no available key exists, contact the locksmith (Page 72-350 or ext 5595) and arrange for a new key to be made The receiving person must present a legitimate need to receive the key
4	ATSO	Complete the changed Key Distribution List (ATCSF I A 1) and sign it Maintain all previous lists to insure a permanent record is kept for all key distributions
5	ATSO	If a key is lost or becomes unaccounted for, the locksmith must be contacted immediately to change the key on the lock, and make new keys for all persons on the Distribution List The ATSO will then distribute the new keys to the persons on the Key Distribution List

7 PROCEDURE FOR MAINTENANCE OF COMPUTER ROOM COMBINATION AUTHORIZATION LIST (REFERENCE 4 c)

ITEM #	RESPONSIBLE	ACTION
1	ATSO	Upon changing the Computer Room combination, review the Combination Authorization List form (ATCS Form I A 2) for any potential changes
2	ATSO	If any changes (deletions or additions) are to be made to the list, complete a new list and sign it Maintain previous lists to insure a permanent record is kept

8 PROCEDURE FOR CHANGING THE COMPUTER ROOM COMBINATION (REFERENCE 4 d)

ITEM #	RESPONSIBLE	ACTION
1	ATSO	Every 60 days, or when a person is removed from the Computer Room Combination Authorization List, or whenever the combination is believed to have been compromised, the combination is to be changed
2	ATSO	Fill out a Maintenance Work Request and send to Facilities and Maintenance, specifying the date to change the lock Facilities and Maintenance will coordinate the appointment time. In case of a compromise, or known security risk, the combination must be changed within 24 hours In such a case, contact the locksmith (page 72-350 or ext 5595)
3	ATSO	Contact all persons on the Computer Room Combination Authorization List (ATCSF I A 2) and inform them what the new combination will be, and the date/time it will be effective The combination will be communicated verbally whenever possible, else it will be communicated via confidential memo Indicate on the Authorization List the date each individual was informed, and the date of the combination change
4	SA/ Locksmith	Meet at the Computer Room at the arranged time
5	Locksmith	Opens the lock
6	SA	Changes the combination to the new value
7	Locksmith	Closes the lock

MANMAN COMPUTER ROOM ACCESS LOG

DATE	TIME	KEY ISSUED TO	GUARD ISSUING KEY	REASON

ATCSF I.A.1

APPENDIX F

AT HP/1000 COMPUTER ROOM <KEY> DISTRIBUTION LIST

Key #	Name	M/S	Ext	Function/Reason

Authorized _____ Name of AT ATSO _____ Signature of AT ATSO _____ Date

ATCSF [A 2

AT HP/1000 COMPUTER ROOM COMBINATION AUTHORIZATION LIST

Seq #	Name	M/S	Ext	Function/Reason
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				

DATES OF COMBINATION CHANGE AND NOTIFICATION

CHANGE DATE	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

Authorized _____ Name of AT ATSO _____ Signature of AT ATSO _____ Date

CONTINUED ___

ATCSF [A 3

APPENDIX F

AIR TRANSPORT DIVISION

FUNCTIONAL PROCEDURES

MANUFACTURING OPERATIONS

PROCEDURE REV	
P MFG.01 02	Item Master Source Code
P MFG.02 01	Wire and Sleeving
P MFG 03 01	Finished Goods
P MFG 04 01	Production Tracking
P MFG 05 01	Sequence Assembly Area
P MFG.06 01	Request for Unplanned Issue - Free Stock
P MFG.07 01	Request for Unplanned Issue - Scrap, LIP, Eng Revision
P MFG 08 01	Request for Unplanned Issue - Stock Issue Discrepancies
P MFG 08 02	Input and Maintenance of Item Master Data File
P MFG.10 01	Work Order Operation Completions
P MFG 11 01	Work Order Completions to Stock
P MFG 12 02	Factory Work Order Control of Service Parts Orders
P MFG 13 01	Factory Work Order Control of GSI Orders
P MFG.14 01	Assembly Scheduling and Work Order Release
P MFG.15 01	Customer Returned Goods
P MFG 16 01	Request for Unplanned Return
P MFG.17 01	Serialization and Reworkable Control
P MFG 18 01	Diversion of Work-In-Process

AIR TRANSPORT DIVISION

FUNCTIONAL PROCEDURES

MASTER PRODUCTION SCHEDULING

PROCEDURE REV	
P MPS.01.01	Master Production Schedule Procedures
P MPS.01 01	Manufacturing Assembly Scheduling Procedure
P MPS 03.01	Procedure for Closing and Deleting an Open Work Order for Manufactured Assemblies in Support of the Master Production Schedule and the Material Requirements Plan
P MPS 04.01	Manufacturing Assembly MANPAK Rework Order Procedure
P MPS 05.01	Subcontract Orders
P MPS 06 01	ABC Code Calculation and Maintenance Procedure
P MPS 07 01	Class Code Procedure
P MPS 08 01	Fixed Lead Time Assignment and Maintenance Procedure
P MPS 09 01	Economical Order Quantity Assignment and Maintenance
P MPS 10 01	Order Policy Code Assignment and Maintenance Procedure
P MPS 11 01	Safety Stock Assignment and Maintenance Procedure
P MPS 12 01	Shrinkage Percentage Assignment and Maintenance Procedure
P MPS 13 01	Fenside Assignment and Maintenance Procedure
P MPS 14 01	Number of Days Supply Assignment and Maintenance Procedure
P MPS 15 01	Product Code Assignment and Maintenance Procedure
P MPS 16 01	Procedure - Material Exception
P MPS 17 01	Source/Status Code
P MPS.18.01	Remodates
P MPS 19.04	Procedure for Planning and Releasing Work Orders for Manufactured Assemblies in Support of the Master Production Schedule and the Material Requirements Plan/or Standard Production

AIR TRANSPORT DIVISION

FUNCTIONAL PROCEDURES

STOCK ROOM

PROCEDURE REV	
P STK 01 02	Receipt of Material from Receiving Inspection into Stock
P STK.02 02	Cycle Count by A, B, C
P STK.03 01	Issue of Planned Work Orders
P STK.04 02	Unplanned Issues
P STK 05 01	Unplanned Receipts into Stock
P STK 06 01	Stores Location Control
P STK 07 01	Relieve a Shortage
P STK 08 02	Completion of Work Orders into Stock
P STK 09 02	Receiving Process - No Inspection Required
P STK 10 01	Issue to a Sales Order
P STK 11 02	Stock Room Issue Discrepancies
P STK 12 01	Record a Shortage
P STK 13 01	Defective Parts Within WIP, Rec. or Prod Stock
P STK 14 01	Cycle Count of WIP

AIR TRANSPORT DIVISION

FUNCTIONAL PROCEDURES

PURCHASING

PROCEDURE REV	
P PUR.01.01	Traveling Requisitions
P PUR.02 00	Engineering CPRs
P PUR.03.01	Sample Orders (No Charge Samples)
P PUR.04 01	Orders to Government Avionics
P PUR.05 01	Orders to Telecommunications (CTPD)
P PUR 06.01	Orders to General Aviation (Melbourne)
P PUR.07.01	Orders to Switch Lab (Government Avionics)
P PUR 09	Purchase Material Returns (PMRs) (not distributed)
P PUR 10 01	Return to Vendors from Receiving
P PUR.11 01	Use of Register Sales Slip
P PUR 12 01	Tool Procurement and Administration
P PUR 13	Government Rated Sales Orders (not distributed)
P PUR 14 00	Source Development
P PUR 15	Clearance of Invoices with Prices Higher than PO Price (not distributed)
P PUR 16 00	Requirements for Obsolete Parts (not distributed)
P PUR.M 01 01	Lambtime, Buyer Code, Prime Vendor Code
P PUR.M 02.01	Lot Sizing Fields
P PUR.M.03 00	Obsolete, Nonprocurable, and Last Buys
P PUR M 04 01	Use of Prime Vendor Code
P PUR.M 05.01	Bill To/Ship To and Special Rates
P PUR.M.06 01	Vendor Master File
P PUR.M 07.01	Manufacturer/Vendor Cross Reference File
P PUR.M.08	Changes in Source Code (not distributed)

APPENDIX G

ROCKWELL, AIR TRANSPORT DIVISION — FUNCTIONAL PROCEDURE

TITLE: REQUEST FOR UNPLANNED ISSUE - SCRAP, LIP, ENGINEERING REVISION, AND OTHERS

DEVELOPED FOR: Manufacturing Operations

PROCEDURE NO: P MFG.07.01 — REVISED:

PREPARED BY: J.W. Schrichte

REVIEWED BY: _____ DATE 1-31-84

_____ DATE 1-31-84

_____ DATE 1/2/84

_____ DATE _____

_____ DATE _____

_____ DATE 2/6/84

APPROVED BY: _____ DATE 6-25-84

I. PURPOSE:

This procedure defines methods and responsibilities for the submission of material requisitions for replacement parts due to scrap, lost-in-process (LIP), and Engineering revision.

II. SCOPE:

The methods and responsibilities defined in this procedure apply to the areas of Manufacturing Operations and Stockroom Operations.

III. REFERENCES:

Unplanned Issue Procedure P.STK.04

IV. DEFINITIONS:

Standard Order - Orders released for assemblies which involve standard production effort and which have defined Bills of Material. Standard orders are defined with 10 in the second and third positions of the work order number.

Non-Standard Order - Orders released for assemblies which will involve production effort not covered by the assembly standard or which may involve undefined material requirements. non-standard orders are defined with the 2 digit job charge prefix in the second and third positions of the work order number.

Product Code - a 6 character code that relates an item to the product type of the seed or top level end item.

Character 1 - Build Department
Character 2 - MFG. Family i.e. Envr Kntr, etc.
Characters 3-4 - System Product Code (SP)
Characters 5-6 - Top Level Identifier

Job Charge No. - a unique 6 digit order code used to define the type of authorized non-standard effort to be performed i e. 15-XXXX (CMS's), 45-XXXX (reworks), 29-XXXX (MJO's), 39-XXXX (Engr. projects), etc.

V. DOCUMENTS

Request for Unplanned Issue	Exhibit A

VI. POLICIES:

1. Miscellaneous issues of material to the production floor in support of standard orders will be charged against a Material Usage Variance (MUV) account defined by the assembly's Product Code.

2. Miscellaneous issues of material to the production floor in support of non-standard orders will be charged to the work order account defined by the Job Charge Number.

VII. PROCEDURE:

Item No.	Responsibility	Action
1	Manufacturing Operations	Initiate Request for Unplanned Issue (Exhibit A) for replacement parts Retain originator copy in suspense file as a record of the request. Submit request form directly to the Inventory Transaction Control Center.
2	Inventory Control	Process Request for Unplanned Issue submitted by manufacturing operations in accordance with stockroom procedure (P.STK.04).

REQUEST FOR UNPLANNED ISSUE
TR, 101
002422

DATE _____ (1)

PN (2) — UM (3) QTY REQUESTED (4)

INVENTORY LOCATION — REQUESTED BY (5)

LOT NUMBER — PHONE NO (6)

QUANTITY ISSUED — DELIVER TO WORK CENTER (7)

W O NO (8) — ISSUED BY _____ DATE _____

MATL REQ NO (9) — CLASS CODE _____

ACCT NO (10) — APPROVED BY _____

REMARKS (11)

(12) ESTABLISH REQUIREMENTS FOR SHORTAGE QUANTITY

TRANSACTION

DFA-5161-200

Items 1 thru 12 are to be completed by the originator. All other information will be added by the material transaction clerk.

Items 1 thru 7 - Self explanatory

Item 8 - Enter "None"

Item 9 - A 10 character, non-duplicating, material requisition number

Characters 1 thru 6 - Enter the form serial number (Example: 002422).

Characters 8 thru 10 - Enter the deliver to work center.

Item 10 - A 10 character debit account number.

Characters 1 thru 6 - Enter the account number from the work order packet.

For standard orders '10' this will be the Product Code
For standard orders '15,45,29, etc' this will be the Job Charge Number

Characters 7 thru 9 - Enter the requesting work center.

Character 10 - Enter the Reason Code

S = Scrap
L = Lost-in-process
R = Engineering revision requirement
O = Other

Item 11 - Enter any pertinent information which will aid in the routing of the issued parts, i e., Work Order Number, Used on assembly PN, etc.

Item 12 - Check (X) if a shortage should be established (R,304) for any unsatisfied quantity. Leave blank if a shortage should not be established for the unsatisfied quantity.

Exhibit A

APPENDIX H

Rockwell Air Transport Computer Security Policy

TITLE: MRMRMR Release Installation
PROCEDURE NO: ATCSP IV.B.23
PREPARED BY: S. W. Pasker

REVIEWED BY: _____ DATE: 11-2-86
_____ DATE: 11-2-86
_____ DATE: 10-2-86
_____ DATE: 2-2-86
_____ DATE: 10/6/86

APPROVED BY: _____ DATE: 11-2-86

1. PURPOSE

This procedure describes the steps and responsibility for installing a new MRMRMR release.

2. SCOPE

This procedure covers the following items:

1. MRMRMR System Manual
2. MRMRMR Programs
3. Rockwell Developed Programs and Interfaces
4. Testing
5. Implementation

3. REFERENCE

None

4. POLICY

All new releases will be installed.

5. DEFINITIONS/ABBREVIATIONS

AT - Air Transport MRMRMR User Community
SM - MP System Manager
SS - Software Support
UA - User Analyst
MRMRuuuu - Latest ASK Software Release Number

6. PROCEDURES

ITEM #	RESPONSIBLE	ACTION
1	AT/UA	Review pre-release announcement and related documentation.
2	SS	Request new release tape as soon as the release is made available by ASK.
3	SS/UA	Obtain ASK Manual updates, and create a new "MASTER" Manual for the new release. Reproduce and distribute the new release manuals.
4	SS/UA/AT	Review changes in the new release. Determine pre-release installation candidates.
5	SS/UA/AT	Determine schedule for new release implementation.
6	SM	Establish a new MP account "MRMRuuuu" where uuuu is the latest ASK software release.
7	SS	Establish groups in the new "MRMRuuuu" account as necessary, with the same capabilities as the MRMRMR account groups.
8	SM	Load the new release tape to the "MRMRuuuu" account.
9	SS/UA	Print and review the new release documentation files.
10	SS	Apply Rockwell security mods and program changes per change log, to the new source.
11	SS	Modify all Rockwell written programs and interfaces as necessary.
12	SS	Modify DISPATCHER to allow MRMRMR to run from "MRMRuuuu" account.
13	SS	Add Rockwell "specials" to the "MRMRuuuu" account from the MRMRMR and PROD accounts.
14	SM	Establish test account "TESTuuuu" where uuuu is the latest ASK software release.
15	SS	Create test databases and related files in the "TESTuuuu" account from the XMRM157P test account, converting databases and files as necessary.
16	SS	Establish user "TESTER" on "TESTuuuu" account, with LOGON MGRREAK UDC executing MRMRMR.MPUB.MRMRuuuu.
17	SS/UA/AT	Checkout new release of MRMRMR on "TESTuuuu" account.
18	SS/UA/AT	Update documentation, procedures, and training manuals as necessary to support the new release.
19	UA/AT	Train users on the new release.
20	SS	Implement new release, per the predetermined schedule. This includes the following steps: a. Backup files to be converted/changed. b. Convert files/databases/etc. as necessary, per ASK installation instructions.
21	AT	Provide final go-ahead on new release.

APPENDIX I

DATABASE UPDATE LOG

DATABASE _____

DATE UPDATED _____

PROGRAM _____

UPDATED BY _____

LOGON ID _____

REQUESTED BY _____

DESCRIPTION OF UPDATE

APPROVED BY _____

Rockwell International

BUSINESS COMPUTER SYSTEM DOCUMENTATION

H/P 3000
RELEASE NOTICE NO.

System ID | System Title:

DESCRIPTION: EFFECTIVE DATE _____

	SOURCE FILE	PRODUCTION FILES			*TYPE	CAP=	MAX DATA	LANGUAGE	
		SOURCE	RBM	PROGRAM					
G						IA,BA,MR,PH,DS,PM	MAX ☐		
A		TITLE:						48	68
G						IA,BA,MR,PH,DS,PM	MAX ☐		
A		TITLE:						48	68
G						IA,BA,MR,PH,DS,PM	MAX ☐		
A		TITLE:						48	68
G						IA,BA,MR,PH,DS,PM	MAX ☐		
A		TITLE:						48	68

☐ CONTINUED

*TYPE LEGEND	INDICATION OF USER ACCEPTANCE
P — Program Release RL — RL Program Release RUL — RL and USL Program Release SL — SL Program Release NS — Compile and Prep Only - No Source RLSE IN — Include Release IS — Image Schema Release Q — Quiz Program Release QS — Quiz Schema Release QG — QKGO File QK — Quick QT — QTP U — UDC	

SIGNATURE	PREPARED BY	RELEASED BY	AUTHORIZED BY
NAME TYPED			

074-8289-303 (Rev 10-85)

C. PROGRAM RELEASE, TYPE = "P", (Cont.)
(H00, XMAN, RE, UT, PH)

2. If the program is a stand-alone program
(ie. no RBM, and the program is not MANMAN)
the Library will:

a. Insure the $CONTROL card as the first statement of the source, specifies the LIST parameter. USLINIT is alright. (/T source.group.acct; /L "$CONTROL").The contents of the $CONTROL card is the responsibility of the programmer.
NOTE: Fortran 77 does not have "$CONTROL", it has only "$". LIST=default, need not specify.

b. Modify and submit job "RLSEPGMA.S" (A.4) which:
NOTE: H00 & XMAN stay in SOURCE.PROD.

c. Modify and submit job "RLSEPGMB.S" (A.5) which:
NOTE: RE,UT,PH are in MSOURCE.PROD.
1) Compares new source to current production.
2). Copies the source from the test file to "file.xSOURCE.PROD".
3). Compiles the released source into "$OLDPASS".
4). PREP's $OLDPASS into "program.RELEASE.PROD".

c. The compile and prep will be verified by the cognizant programmer.

d. The Library, upon approval of the compile and prep, will move "program.RELEASE.PROD" to:
1). "program.xPUB.PROD" via the modified job "RLSEPPGM.S" (A.6) or the UDC "RLSEPPGM" (:RLSEPPGM H003L___,PUB)
-OR-
NOTE: Same as RENAME HxxxLxxx.RELEASE, HxxxLxxx.PUB
NOTE: All XMANxxx and HnnnLnnn programs go to PUB.PROD.

2). "program.MPUB.MANxxxx" via the modified job "RLSEMPGM.S" (A.7) or the UDC "RLSEMPGM" (:RLSEMPGM program)
NOTE: All REnnn, PHnnn and UTnnn programs are actually part of MANMAN, and are released to MPUB.MANxxxx. (Generally, all others go to PUB.PROD)
NOTE: Library must be logged on as ":HELLO LIBRARY.MANxxxx" to release MANMAN UDCs

as indicated on the release notice.

e. If the release went to MANxxxx account, notify HP Service Administration to release the job that night after all users have been stopped, and before the nightly backup. (Refer to the Release Assistance Request Form - No need to Deallocate and Allocate except if the release is for XMAN002,003 or 008)

EQUIPMENT REPLACEMENT IN DISASTER RECOVERY

Charles H. Finley, Jr., CDP
ConAm Corporation
1661 Nineteenth Street
Santa Monica, California 90404

Most of us don't like to think about becoming seriously ill, or even consider our own death, let alone make arrangements for these things. They are unpleasant topics. It would seem that disaster recovery is almost as unpleasant a topic. This, coupled with the extremely high reliability most HP 3000 computer users experience, makes it so that not much happens in the HP 3000 world with regard to disaster planning.

Fire, earthquakes and floods are real things, and just like gravity they don't give a damn whether you believe in them or not. Probably very few of us will have to deal with a disaster of major proportions, but many more of us will have to deal with a smaller problem, such as the loss of a computer system for three or four days.

This essay addresses one small, yet essential, aspect of the disaster recovery subject and that is where will all of the computer equipment come from if you need it in a hurry. We will identify sources of replacement equipment, describe hardware-related installation considerations, discuss the most important factors that influence hardware replacement, and discuss ways to minimize hardware replacement time.

REPLACEMENT EQUIPMENT INSTALLATION CONSIDERATIONS

Equipment replacement is only one of many considerations in disaster recovery. There are other installation considerations that must be addressed if the computer is to be used in a recovery. Software and data must be available and up-to-date. Personnel must be available to run the computer. Various support issues must also be addressed such as supplies, data recovery procedures and vendor installation support.

In fact, computer equipment is not the only equipment needed for recovery; there must be a proper site to run the computer. A proper site means the right power, air conditioning, communications connections, etc. These installation considerations are discussed elsewhere and are the subject of many books and magazine articles. Our topic is limited to the replacement of the computer equipment only, and more specifically, replacement of HP 3000 computer hardware.

MAJOR REPLACEMENT EQUIPMENT SOURCES (Outside of Company)

Obviously, the best source for replacement equipment is within one's own company. Most of us cannot afford and/or cost-justify a duplicate configuration. That leaves the following choices: the manufacturer, disaster recovery firms, majorresellers, other resellers and brokers, service bureaus and finally, other neighboring firms.

IMPORTANT FACTORS IN HARDWARE REPLACEMENT

Each disaster is unique and the "best" source will depend on the situation. Figure 1 shows some of the factors that must be considered in choosing the best source. The factors listed on the left column are discussed below. Each equipment source is evaluated with respect to each factor.

FACTORS VS. SOURCES

Figure 1 provides a summary of an analysis of how each source for replacement equipment addresses those factors identified in its left vertical column. As the reader can see no one source is perfect for all cases.

Degree of Loss

Obviously there is a wide range of loss possible. Risk of loss runs the gambit from a major catastrophe, in which all equipment is lost, to a minor component loss such as a terminal.

We see from Figure 1 that the only source that can replace all equipment lost is HP, while some other third-party vendors may be able to replace most lost equipment. The remaining sources can only help with critical applications. It is important to note, however, that these remaining sources do not offer permanent replacement hardware. They provide the temporary use of hardware, usually at a remote site. The problem of permanently replacing the hardware still remains.

Size Installation

Hewlett-Packard should be able to replace the equipment in any size installation providing the equipment is of fairly current manufacture. It is widely believed that their commitment is to provide the next machine off the assembly line to a customer under their maintenance program. There is some chance of getting most needed equipment in the third-party used equipment market. Major resellers maintain a large inventory of HP equipment, and other resellers and brokers may also have various combinations of equipment available at any given time. The larger the computer or peripheral the more likely that it will not be in stock, however. The remaining sources can also be relied on primarily for critical applications. Disaster recovery firms, service bureaus and neighboring installations normally expect to provide you with resources to run critical applications for only a short period of time.

Financial Questions

Just because your company is experiencing a disaster and may go out of business unless it gets replacement equipment, is no reason to expect vendors to bypass their standard credit policies. Face it, they still have to do credit checks and they may require pre-payment and/or deposits. All of this takes time.

There are things one can do in advance to speed up the financial process. These vary depending upon the vendor. HP will allow you to maintain an open, pre-approved purchase order with them. Other third-party resellers', brokers' and service bureaus' policies will vary. Disaster recovery firms and neighbors can be expected to respond to subscribers or firms with prior agreements in place.

No matter what, before you get the equipment, a responsible party with the ability to pay must be identified to the supplier or you're out of luck.

Insurance

The right insurance policies are important if you are to use someone else's equipment and/or site. Insurance policies should consider fire, natural disaster, water damage, power failure, sabotage, etc.

One can expect insurance to be included by either a service bureau or a disaster recovery firm. A major reseller should be able to obtain special insurance within 24 hours. The ability to provide insurance varies for other vendors.

Transportation

Since the equipment is usually not moved in the case of disaster recovery firms, transportation is not an issue. There is one notable exception, Uptime, but this is included in their service. In all other cases the vendor relies on other outside services for transportation.

Acquisition Method

Only HP, and certain resellers and brokers, allow for rental or purchase of the equipment. All other options allow only interim use of the equipment.

Mean Time to Replace

Consult Figure 1 for an indication of the mean time to replace equipment. Basically, HP can replace equipment in as little as 48 hours, but more likely it will take 10 days. Disaster recovery firms are most responsive in this area in that they can respond in as little as 4 hours, and typically no more than 72 hours. Depending upon the customer location and what's in stock, third-party vendors can replace equipment in as little as 4 hours, or as much as 2 months.

Reliability for Quick Complete Replacement

HP cannot and will not replace all equipment. It does not manufacture some items or they may be out of production. Also, there may be a limit to how many Series 70s they are willing to pre-empt for one customer. Disaster recovery firms are the most reliable source for their subscribers. That's why they're in business. However, disaster recovery firms do not offer complete replacement; they offer a place to run critical applications. All other sources may or may not have what you want, when you want it or, in the case of your neighbor or a service bureau, may not be able to accommodate you when you need it.

MINIMIZING HARDWARE REPLACEMENT TIME

Given enough time and money any computer hardware configuration can be replaced. Unfortunately, a company can go out of business if replacement takes too long.

In order to minimize replacement time consider the following:

- o Create and maintain a disaster recovery plan
- o Contract with a disaster recovery service
- o Make a reciprocal agreement with your neighbor
- o Know how your insurance company will handle things in the event of a disaster
- o Periodically reassess, confirm and evaluate the above

SUMMARY AND CONCLUSIONS

TIME is your ENEMY in a disaster. It takes time to get up and running. You must locate equipment, take care of financial considerations, transport the equipment and provide insurance for it. You must be able to keep the equipment long enough (until you can get a permanent replacement). You must know in advance what are realistic timeframes to obtain equipment and what delays you can live with. If you can't live with what's "normal", then you have to do something NOW to get the odds in your favor.

\ HARDWARE \SOURCE FACTORS \ HDW REPLACEMENT\	HP	Dis Recov Service	Major Res'lrs	Oth Res'lr & Brokers	Service Bureau	Neighbor
Degree of Loss	Any	Critical Appl	Most	Most	Critical Appl	Critical Appl
Size Installation	Any	Critical Appl	Most	Most	Critical Appl	Critical Appl
Financial Questions	Open P.O.	Subscribers Only	Open P.O.	Varies	Open P.O.	Prior Agreement
Insurance	Varies	Included	24 Hr. Lead Time	Varies	Included	?
Transportation	Others	Included	Others	Others	N/A	N/A
Acquisition Method	Prchse/ Rental	Interim	Prchse/ Rental	Prchse/ Rental	Interim	Interim
Mean Time to Replace	48 Hrs - 2 Weeks	24 Hrs - 72 Hrs	4 Hrs - 2 Mos	4 Hrs - 2 Mos	4 Hrs - Never	4 Hrs - Never
Reliability For Quick/Complete Replacement	80%	Most for Subscribers	0%- 100%	0%- 100%	0%- 100%	0%- 100%

Figure 1

Effective Systems Development using 4GL

Patrick Fioravanti
Infocentre Corporation
7420 Airport Road
Suite 201
Mississauga, Ontario
Canada L4T 4E5

Introduction:

As Fourth Generation software becomes more prevalent throughout the HP3000 community it seems that the 3GL versus 4GL debate has subsided. Organizations have recognized that Fourth Generation software offers a viable solution to many of the challenges facing their data processing departments.

Many organizations using 4GL have found that the actual gains do not quite measure up to the acheivements and benefits anticipated from using Fourth Generation software. The inadequacy lies not with the software itself but within the system development methodology in which the 4GL is used. Any data processing organization committed to using 4GL must be prepared to examine its application development methodology in the light of the characteristics of the new tools.

Our methodolgy and general approach must adapt to capitalize on the strengths of the 4GL tools. Fourth generation software can bring with it a whole new approach to system design wherein the user can take a more active participatory role, and the DP staff while getting off to a faster start, can remain flexible and responsive to changes in user requirements.

Let's pursue this opportunity offered by a 4GL, identifying the benefits offered by the tool in the development process. We will address how the tool can be used effectively, looking at the resources and skillsets required, and also pointing out the potential dangers and pitfalls to be avoided.

Let's begin by examining how we approached things in the past. Structured design methodologies gained acceptance and developed a large following during the seventies. This methodology systemized the application development process. It typically involves the formal definition of phases, into which every development project is subdivided. Each phase is completed, one after the other in a linear fashion. Signoffs, intended to signify user acceptance (compliance maybe) mark the conclusion of each phase. Throughout each phase, the project team members work in strict adherence to pre-defined standards and procedures.

The phases followed in a development project vary from one installation to the next, but typically follow the same pattern:

Phase	Description
1.	Requirements Definition
2.	Analysis
3.	Physical Design
4.	Programming
5.	Testing and Documentation
6.	Implementation.

Usually there is little or no overlap between the phases. The outputs generated by the first three phases entails a lot of paperwork. It's not until the programming phase gets underway that the computer system plays an active role, and in this phase the role consists of 3GL compiles and creation of Database and file structures.

The design of this methodology is based on several assumptions:

1) That the process of developing application systems is linear, and can be approached in a multi-phased linear fashion.

2) Users know all (or at least most) of their needs up front and can effectively communicate those needs to DP analysts.

3) The users needs won't change during the development process.

This methodology enjoys varying degrees of success, depending on: the length of the project, the DP sophistication of the user, the seasoning of the DP staff.

In reality the progression of the development project through the phases tends not to be linear but rather somewhat iterative. Users typically have difficulty knowing/expressing all of their system requirements up front. Analysts discover missing pieces of the puzzle when the design phase is underway. Programmers stumble across design inadequacies or deficiencies during the programming phase. Each of these roadblocks in turn send the project temporarily back to an earlier phase.

Even without these regular setbacks, the development process takes a long time

because it is manual labour intensive. The user interviews and fact finding undertaken in the early stages drags through seemingly endless meetings. Data flow diagrams take an eternity to produce, while the programming staff spends many hours with a text editor writing and fixing programs in a Third Generation language. By the time the system is ready for implementation it's already partially obsolete due to the constantly changing business environment. To add to our problems, we are chasing a moving target.

Adopting Fourth Generation software presents some differences in the application development tools. Let's take a few minutes to identify the major differences between Third and Fourth Generation software.

As mentioned above, with Third Generation software the process is manual labour intensive. The programmer works with a text editor to create/maintain his source file(s). The programmer works at a relatively low level concerning himself with tasks such as: routine data editing, screen handling, accessing the file system, creating/maintaining the Data structures and so on.

Fourth Generation Software should provide you with a complete application development environment. The software is more than just a language, complemented by other modules which provide:

* automated generation of all system components including Database structures,
* relational access to Database files,
* automated production of user documentation,
* end user report generation,
* micro-mainframe networking capability,
* production of presentation graphics.

The language component lying at the heart of the software environment, is non procedural, action driven enabling the programmer to work at a much higher level than is possible with a traditional Third Generation Language. Typically with a 4GL, the programmer no longer concerns himself with coding the lower level routines handling Screen I/O, File/Database I/O, data editing and validation, basic reporting and so on. With a non procedural language, the programmers job is to provide the language processor with the specifications of what is to be done, leaving the *hows* (detailed processing logic) to the language processor. Built into the language processor are standard pre-determined algorithms dictating the method by which the typical Business oriented DP functions are to be processed. These typical functions are:

* Menus
* Data entry/update/inquiry Screens
* Reports
* Bulk update routines.

The language component aside, the development environment provided by the Fourth Generation software opens up some new frontiers. With a system designer

tool and a documentation module it becomes obvious that we can let the computer do a lot of the work for us. The process can become far less manual labour intensive. Other modules open up the possiblity of more flexible Database access and end user computing, thereby relieving the DP staff of servicing the ad hoc reporting needs of the application. This end user computing capability can be extended through the micro-mainframe networking link, porting the users data to the PC world where the end user can further process it with the adopted PC software.

As a result of these 4GL characteristics, system development can be accomplished in far less time than with the traditional tools. But many organizations who have implemented a 4GL are finding that the actual gains are not meeting their expectations. Replacing Third Generation programming tools with a 4GL has been known to create havoc as frustrated teams of programmers struggle with a new environment which offers them less programming flexibility than their favourite 3GL. DP managers, while wondering why their expected productivity gains are not materializing, are confronted with claims like:

"I could have done this faster in COBOL.", or
"This could be done better in COBOL.", or
"This can't be done using the 4GL.".

These experiences are real, but in most cases can be avoided. The problem is not related to the capabilities of the software. Fourth Generation software has evolved over the past ten years into sophisticated and powerful tools. With proper implementation of the tools and a development methodology adapted to the characteristics of 4GL software we can begin to realize significant returns on our investment.

Assuming that our first priority is to develop quality applications faster - reducing the application backlog, then the sensible approach is to capitalize on the strengths of the new productivity software tools. What it comes down to is the fact that Fourth Generation software enables a fully functional application to be developed in the time it used to take to produce a complete system specification document. We should spend our time creating applications rather than writing up specifications.

This approach is what is generally referred to as *prototyping*. Let's not dwell on the mechanics or techniques of effective prototyping. Those topics have been treated at length by other authors. Rather, let's examine the general nature and benefits of prototyping and how this technique fits in with Fourth Generation tools.

Prototyping is an iterative exercise wherein a working model of the application (or a subset thereof) is built and refined. This exercise involves active participation of the end user. A natural result of using this technique effectively, is the design of an application that most closely addresses the users requirements. During a prototyping session, an analyst and the end user are seated in front of a terminal using the system designer module of the Fourth Generation software. In this mode, the user *gets what he sees*, and the analyst is able to clearly confirm whether or not he understood the users communicated

need. Building a working model of the application during the Design phase of a project implies taking a scientific approach to system development, in a sense you are constructing an experiment to test and verify an hypotheses;
ie:

* Is this what they said they wanted?
* Is it what they really want?
* What else do they want that they haven't told us yet?
* How well will this function (perform).

These types of questions can be answered during the Design phase by constructing a working model of the application. The approach is made feasable by the capabilities of the software.

The prototyping activity typically begins in the physical design phase of the project. At this point the Analyst has been sufficiently involved with the application to develop the knowledge base from which a prototype model can be started. Beginning the prototype too early in the project is counter productive. The analyst does not have a sufficiently firm grasp of where he is headed. Leaving it too late on the other hand, can have a stifling effect, where new ideas or objections are discouraged. There is a window of time during the Design phase when prototyping should commence. You will identify this window after some trial and error.

When done properly, the output from the Design phase is a working model of the application tailor made to the users specifications. This model is passed on to the programming staff who fill in the details: complete edits and checks in the on-line transactions, polished versions of the production reports, complete programming of the batch update processing, and so on. The whole process flows a lot better. Fewer problems are encountered in the programming phase since:

* Design deficiencies would have sufaced when building the model,
* The application was designed from scratch with the 4GL.

Let's be certain not to confuse the issue however, prototyping is not a substitute for solid, up front analysis, nor for good programming and documentation. Rather it is a crucial part of the Design phase which is appropriate with 4GL tools, and can provide immense benefits during subsequent phases. As we all know, benefits don't simply start to accrue without some problems to overcome, and this scenario is no exception. Some of the problems you can expect to grapple with are:

1) Inadequate understanding of the tools, by all parties. End users as well as analysts need a working knowledge of the prototyping tools. The DP manager must deal with a shortage of readily available programming expertise in the 4GL of his choice. This generation of software is less than 10 years old, the tools are not standardized (and arguably nor should they be), they are not widely taught in colleges/universities, and they require an entirely different thought process to approach a given program than do the traditional languages. The DP manager must develop his own in-shop expertise.

2) Acceptance of the methodology by systems people. A revised methodology constitutes considerable change in the way people do their work. We must be prepared to deal with people's natural reluctance to change.

3) User resistance / incapability. One of the most often cited problems plaguing the prototyping approach is lack of User committment. The user's role changes from a passive to an active one. Many users feel they are doing the DP departments work for them. Beyond that is the question of capability. Have the users ever done anything in the past to prepare them for this new role? Are they computer literate? Are they comfortable in front of a keyboard? Are they creative analytical thinkers?

4) Software capabilities/characteristics. Your chosen Fourth Generation software must be well suited to prototyping. During this activity, system components must be adaptable, especially the Database structure. The interface offered by the system designer tool must be a very natural one for specifying system processing and the underlying Database structure simultaneously.

There is no foolproof, cookbook approach to follow that will ensure success with Fourth Generation software. There is however some issues to address in order to significantly increase the odds of success. In no particular order, they are:

1) Develop expertise in using the tools, equal to or greater than the expertise aquired in the use of 3GL tools. All team players are involved in this education:

Analysts:

Must learn how to effectively use the modules that facilitate prototyping. They must also be aware of the underlying functioning of the language processor, including how it likes to process the DP functions and equally importantly what it doesn't like to do.

Programmers:

Must obtain a thorough working knowledge of their Database Management System/File system as well as all facets of the 4GL. There is no question that the logic built in to the 4GL language processor for handling the typical DP functions is what provides the *Programmer Productivity*. It is also the cause of reduced programming flexiblity leading to tremendous amounts of frustration. With a 4GL the programmer must know the full scope (capabilities) of the language otherwise he is stuck.

Over the long run the programmers in your organization will be (or must grow to be) elevated to a higher level of DP professional. They will become Application System Specialists, with their expertise shifting from the details of lower level programming tasks, to all facets of application system development and functioning.

Users:

Must become aquainted and comfortable with the prototyping process. This includes learning the rules (by which we will develop systems), the terminology (jargon) and the techniques.

This step is very important. We need a very structured and skilled team - each player's role clearly defined, each member a specialist in his own right.

There are some obvious resources to draw on for the development of the required expertise. Some very helpful courses are offerred by the 4GL vendor. Investigate these course offerings. Other organizations (not directly tied to the HP3000 or your chosen 4GL) offer courses as well. Consider on-site consulting. Keep in touch with your 4GL vendor. Their people by the nature of their work are perhaps more intensively involved day to day with the 4GL product than their customers are. They have a lot of acquired expertise to offer.

Some not so obvious educational resources include User Groups (especially the SIGs that have emerged for the major 4GL products), and reference sites. Do some *Networking*, exchange success and horror stories.

While developing the skills, some very practical advice is to experiment for a while in a pilot environment. Don't try to learn while producing a new Production system.

2) Undertake more effective analysis and system design. As pointed out above, the use of a 4GL does not replace the need for solid up front requirements definition and analysis. If anything it makes the successful completion of the early phase more critical because of the collapsed time of the programming phase. The Fourth Generation software ean be effectively used as a Design aid. Implement your design with a prototype. Use the prototype to test your design hypotheses. Correct the design flaws early in the process. Your users active involvment in the development and testing of the prototype will quality assure the accuracy of the design.

3) Put the necessary resources in place.

People:

Users freed from their regular duties so they can undertake their active participatory role. Analysts and programmers with the required skillset.

Machine:

Additional computing resources are required to support prototyping activities. The prototyping team should have hassle-free ready access to terminals, printers and any other necessary peripheral devices.

Software:

The software requirements are: System Design tool that enables fast development of a prototype including the Database structure in addition to the procesing. Behind the system designer module must be a powerful, full functional language processor. These core modules should be rounded out with an automatic documentation tool, presentation Graphics system, and end user reporting module in order to provide a complete Fourth Generation development environment.

Bear in mind that none of these resources will stay in place without the complete backing of senior management.

Let me close with two thoughts for your consideration:

1) Failures will happen and must be accepted. A shakedown period is to be expected after adopting new tools - don't get discouraged.

2) You will know you are on the right track when the limiting factor in delivering new applications becomes the users ability to assimilate new systems. Not to be taken lightly, this also is a significant problem, but one to be addressed at another time.

Summary:

Yes, the powerful Fourth Generation tools open up many new frontiers, and they are exciting opportunities for the DP professional. As these new tools constitute significant advances in application development software it only stands to reason that our development methodologies must change to keep pace with the tools.

A methodology adapted to the strengths of the new software, coupled with a sound working knowledge of the tools will go a long way towards achieving the anticipated benefits offered by the tools.

SOFTWARE DEVELOPMENT STRATEGIES

Janine Firpo
Operations Control Systems
Palo Alto, California

As HP data centers become more sophisticated, users are attempting to introduce more standards and controls into their environments. During the early stages of growth, the need for standard controls resulted in automated batch processing, access restrictions, menu drivers, and non-user scheduling. In today's environments more sophisticated concepts such as file management and accountability are beginning to come under scrutiny.

File Management concepts are not new. In fact, IBM mainframe users have been controlling their development files for years with the aid of PANVALET and ADR/LIBRARIAN. These products separate test and production files, control source code libraries, archive modules and perform audit functions.

Many HP users now recognize the benefit to be derived through the utilization of these techniques. This paper discusses the inherent problems of current control techniques. It will also describe improved methodologies specific to the HP 3000. Before describing possible solution pathways, however, it is essential to agree unequivocally that the current approach to development tracking is inadequate.

Whether it is recognized as such or not, one of the major functions of any data processing department is the manufacture and maintenance of software. This is not a process relegated to development houses only. In all cases the efficiency and cost of development are instrumental to the corporation's success because computer applications have become an integral factor in the competitive struggle for market position. Everyday, MIS departments receive requests to modify software and data. The software alone can represent corporate assets valued at hundred of thousands or even millions of dollars. Data is even more valuable since it is drawn from customer investment and confidence.

A dichotomy begins to appear when it is discovered that many HP 3000 centers continue to operate in a reactive mode. Direct user support is considered the primary function of development and operations. This translates to a daily goal centered on processing user-requested jobs, completing batch production, and distributing reports. Although these functions are the composite result of operations, one should not overlook the basic fact that production's efficiency and accuracy relies upon the cohesiveness of the software components from which it is derived. In much the way that a solid house cannot be built upon shaky foundations, reliable computer processing cannot be achieved unless the associated code contains internal integrity.

The component integrity problem is one which HP 3000 centers are not managing effectively. Instead the issue is currently swept under the rug, where it is expected to escape notice. This policy is not a solution. Although Operations can proceed error free for months, inevitably a simple oversight snowballs into a catastrophic effect. Every experienced manager can

undoubtedly recall such a situation. Perhaps a new production job overwrote entire datafiles. Or perhaps a job ran out of sequence and rewrote vendor checks worth millions of dollars. Or perhaps automatic teller machines never posted a user debit. All these cases have been described to me in the past. The list of potential errors is endless.

The important point is that software development and file maintenance procedures directly impact operations. Should programs fail, run out of order, or produce invalid data, the operations department will be required to rerun production and be pressured to establish procedures.

Unfortunately, once the crisis has passed, so does the direct pressure for change and the site continues to ignore the problem of inadequate controls. It is time to accept the fact that file management standards are required and then to address and resolve the problem.

Remember, inadequacies can be introduced into the development cycle in a variety of ways. The issue now becomes how to assess the cycle and identify areas of concern. Examples are described below.

Programmers often access production source directly. This approach should never be allowed for two reasons. First, the original source code can be destroyed. If a production error results from programmatic changes, the original version cannot be easily restored. Therefore, production can be delayed for several hours, or in the worst case - for days. Second, there is no method to track production access. Programmers are not restricted from making unauthorized modifications. Controlled access of production modules should be a goal of all development paths.

More often than not, current development procedures result in a discrepancy between source and object code. In other words, the current source does not recompile into the current object. It is extremely difficult and time consuming to recreate source code from a load module. Should the source require a change, there is no guarantee that the resultant object will include all the features of the old load module. Production can and does often fail as a result. A bond between source and object code is another development goal.

One of my favorite scenarios involves multiple programmers who make simultaneous changes to the same source module. In this case, the last programmer to update production wins because previous fixes were obliterated. The net result is wasted effort and invalid object code.

In addition to direct product impact, there are other issues involved in discussions of development procedures. For example, visibility to coding changes is often elusive or non-existent. Additions, deletions, or line changes are untraceable. Companies developing software components for other divisions face the additional problem of validating the completeness and accuracy of each product release.

Obviously, many potential problems can result from uncontrolled software development. It would be inaccurate to give the impression that HP users are

not addressing this problem. Some are, but the degree and depth of resolution varies widely. Let's build upon the simplest development pathway and describe successive control layers. This should provide a picture of the various management methods along the control spectrum.

The simplest procedure is no procedure. In other words, programmers merely log directly into the production account, modify the code, test it, and recompile. All three steps occur within the production location. This method has no safeguards against any of the potential hazards enumerated earlier. A slightly more sophisticated approach requires programmers to FCOPY or CHECKOUT source code from the production account and move it into a development location. Unless stringent controls exist, there is no guarantee that only one individual has checked out a particular module. Furthermore, programmers are not restricted from accessing production accounts nor is there any way to audit their access.

The production-to-development strategy can be envisioned at several levels. The development area may be nothing more than an amalgamation of programmer groups. In this case, each programmer copies, develops, and tests in his own group within the development account. This account structure, however, does not ensure a standardized testing environment. A better approach is to maintain a development account which duplicates the production account. In this way each programmer can be confident that alpha testing is occurring in an environment that closely resembles production.

Access to accurate, up-to-date object and load modules can be assured. Ideally, production and development accounts should exist on separate CPUs to completely eliminate the possibility of direct access to master files. Of course, this approach is not always possible and separate accounts on one CPU will produce adequate results.

Usually a split between production and development accounts is as far as most data centers go in their efforts to establish development procedures. Unfortunately, eliminating visibility beyond this point is extremely dangerous. It is tantamount to building a car and leaving out the engine. Aside from external accoutrements, nothing has been obtained. A plethora of considerations still exist.

Alpha testing, for instance, does not represent an adequate test level. Programmers who test their own work can easily overlook bugs. Besides, they tend to test what works rather than attempting tests to "break the code". For this reason, it is strongly suggested that a Quality Assurance (QA) process be initiated. There are several levels at which to install such a procedure depending upon company size and resources. Smaller companies may have reciprocal programmers QA test development efforts. Such retests are often performed in the development group and account. Larger organizations have hired an individual or staff members whose sole function is QA testing. When the process develops this extent, there is generally a separate QA test account which reflects both the production and development environments.

Should a QA effort exist, it is vital that the developer relinquish all claim to the code when it moves to the QA phase. Only one copy of the developing

code should reside in either the development or QA area. If both accounts contain separate copies, inadvertent changes which might not be incorporated in the QA version may be made. Thus, the final production version would not include all code changes. This is a very commonly overlooked source of error.

By the same token, should the QA analyst locate a discrepancy in the modules tested, the code should be returned to the original developer for revision. When this is done, it is QA's turn to relinquish all claim to the code until it is returned by the programmer.

At the conclusion of QA analysis, another vital step should exist, but it is very rarely the case. A higher level manager should perform a final approval on the development effort to ensure all previous conditions have been accurately met.

Following final approval, the enhanced code is ready to be moved into the production location. An FCOPY or move will overlay the original modules. The destruction of the earlier version could be problematic if the revised code contains errors. It would be detrimental if no backup copy of the initial code existed. Therefore, the original modules must be stored to tape prior to enhancement installation.

This final update step can prove to be both time-consuming and error prone especially when large numbers of files are involved. To make matters worse, a compile step is usually required. Re-creation or update of the processing JCL must also be coordinated. Many times new code has been run under old JCL. The net result is production which aborts in the middle of the night.

Companies that develop or change code at a host location and send the code to several remote sites have numerous problems to consider prior to production updates. In these cases, version changes must be accumulated over time and sent as a packet.

These corporations must track the deliverables and be accountable for the software at all remote sites. The scope of this paper does not allow a detailed discussion of this complicating issue. It is described merely as an area of potential concern.

When they exist, all the aforementioned procedures are usually tracked via a piece of paper - a form. An originating service request form often moves with the code from production to the programmer to the QA analyst. Each step along the process is documented on the form. Such a tracking method is inadequate for several reasons. Most simply the paper can be lost or misplaced. Second, any member of the chain can fail to record his involvement. Most importantly, there is no assurance that the form really reflects what has occurred. Individuals have been known to misrepresent information for a variety of reasons - often in the name of speed. A manual paper tracking method can be synonymous to no tracking method. An appropriate, complete and accurate tracking procedure should be a primary concern to the development and audit staff.

It is possible at this point to describe an idealized development pathway. Needless to say each environment will require addendums to or deletions from this depiction. However, the following does describe a general solution based on my own experience.

In this idealized scenario, three accounts exist - production or master, development, and test. Each account structure is a carbon copy of the others to the extent that files moved from location to location retain their original jobname and group designators. Only the account names change. In this way, it is easy to visualize the link between a developing program and it's originating master.

When files move between the production location and the development area, a copy should be made. The original source should never be destroyed. However, movement between development and test should result in only one copy with the deletion of the file in the development account. Thus a programmer cannot access code while it is undergoing QA analysis. Should QA locate a bug, the code should be rejected and returned to the development account for rework. The file in QA should then be purged.

Once QA has approved all changes, a project leader or manager should verify that adequate and accurate test procedures have been followed. Only at this point would code be moved into the production library. Such updates could occur once a day if desired and should be performed by operations or a production librarian. The latter is my recommendation as it restricts the responsibility, control, and audit functions to one individual.

Prior to update, the original production should be verified and stored. Without this step it is much more difficult to return to a prior version in the case of error.

These procedures are often tracked with a service request form as mentioned earlier. After the completion of each step, the form is signed and passed to the next appropriate individual, who accesses and moves the associated file(s). This method has several drawbacks including loss of the form and the inability to verify procedures. For this reason, there are software tools available which perform the tracking and auditing procedures automatically without information loss. The tools also force participants to conform to structured rules to ensure steps along the development pathway are performed in sequence. In order to implement a structured development strategy with some components of the idealized route in your environment, it is vital to define goals. Possible goals can include but are not limited to:

1. Creating a set of rules to minimize file transfer and maximize efficiency.

2. Developing a methodology to accurately track file movements.

3. Ensuring a link between compatible object and source code.

4. Verifying all associated files, such as JCL and databases are saved and moved to production concurrent with source and object updates.

Development strategies, an idealized solution and the goals to consider in achieving the ultimate solution have now been described. To configure your site along the ideal path, four steps are necessary. First, identify and flowchart the specific attributes of your best solution to software development. Second, assess the components of the development strategy that are currently employed. Third, compare the current structure to your idealized goal and prioritize change requirements. Fourth, develop a project plan from the priority list and implement the necessary changes.

Unfortunately, it is not possible to provide flowcharting assistance in the context of this paper. Each development effort is too unique. However, the scenarios previously described should provide hints and suggestions for the first step.

Several areas of concern can be identified during the evaluation process of current strategies. Questions that should be scrutinized during this analysis include:

1. FILE LOCATION: Where do the production files reside? Are all files contained in one account? Several accounts?

2. DEVELOPMENT ACCOUNTS:

   How are the development area(s) configured? Does each programmer maintain a unique development group or does all development occur within the same group?

3. ACCESS RULES: What rules are implemented to control file movement between production, development, and test? Who has access to these locations? When does access occur? Are there any pre-conditions, such as management approval?

4. ACCOUNT STRUCTURE: What is the structure of the production, development, and test accounts? Are they duplicates of one another?

5. VISIBILITY: How much visibility to file movement exists? Can only one programmer gain access to a file at any given time? Can all changes made by each programmer be verified in the content of the final code? Does a validation check for compatibility of source and object occur?

6. QA ANALYSIS: Is there an established need for separate Quality Assurance testing? Does the current development effort include QA testing? Are there plans to move in that direction? Where will QA testing occur? Does the account structure duplicate production and/or development accounts?

7. FINAL APPROVAL: Does tested code pass through a final checkpoint prior to re-entrance into the production account? Who is responsible for this final check?

8. UPDATE STEP: Who is responsible for moving tested code into the production account? At what time(s) does this occur? How much error exists within the current strategy? What can be done to reduce inaccuracies at this step?

9. VERSION CONTROL: Are original production files replaced by newly modified code? Where do copies of old versions reside? Are all versions verified and tracked? What type of recovery procedure is available if new code fails? How does this occur? How complex is the recovery?

10. AUDIT TRACKING: How are files movements tracked? Is there visibility to when, why, how and by whom files were moved? Can inadvertent purges be identified?

With this type of data available it is now possible to perform a comparison between current methodologies and the site specific optimal development path. This third or comparative step is once again, subjective. Each individual must determine for himself where the current procedure diverges from the idealized goals. Prioritization of differences are dependent on site defined needs. For example, the auditors may be clamoring for file movement control. Thus the implementation of tracking procedures would be the primary concern. On the other hand, QA testing may be the vital link. In either case, auditors can prove to be a wonderful resource in this evaluation process. The comparison step tends to be the easiest in the four step strategy. It draws results from the contemplation required to accomplish the previous two steps.

Following prioritization, a project plan should be implemented to attain the stated goals. The possible scenario suggested earlier will have to suffice as an example for it is not possible to itemize a list of solutions due to their complexity and variability. However, generalized solutions should have evolved through the process of current assessment, goal derivation and priority setting.

To summarize, the need to control and track the software development cycle is becoming increasingly apparent. Without standardized controls it is virtually impossible to establish the validity of software modifications. This is most important in larger environments especially those that manage enhancements for remote locations.

Therefore, it is important to move toward an idealized softwware development process. This goal can be accomplished by comparing the optimal solution to present controls and implementing plans to minimize strategy gaps. Manual

tracking procedures can be utilized for this purpose, if necessary. Fortunately, the industry also offers software tools for automatic solution.

RUNNING THE HP 3000 IN AN OPERATORLESS ENVIRONMENT

Brayton A. Fisher
Operations Control Systems
Palo Alto, California

INTRODUCTION

A number of factors have hastened the search for an operatorless HP 3000 environment. Many believe there is currently a shortage of trained operators which hampers data center operations. Further, the lack of a career path and job enrichment can make this career choice unappealing for future data processing professionals. In addition to a future operator shortage is the cost of manually monitoring production. During peak workloads production may have to be run nearly 24 hours a day. This requires operator presence at all times even for mundane, predictable tasks. Sophisticated production environments have already been developed without operators through the use of PC-based local area networks. The question remains, "Can an operatorless environment be attained in the more powerful HP 3000 environment?"

The introduction of the HP 3000 Series 37 and the Micro 3000 have provided more powerful tools for automated operation. In using these systems as an example, we may foresee the direction of automation for more sophisticated HP machines.

PRESENT AUTOMATION METHODS

This paper will discuss the issues of running a data center without an operator. Hardware and software is available to automate many tasks. However, this does not mean that all operator's duties can be eliminated or automated. Instead, remaining duties can be scheduled at predictable periods, so that actions are performed at predetermined intervals by non-dedicated personnel with little technical experience. The focus will be on:

* Software and hardware available in automating operator tasks, and
* Evaluating the time period and techniques best suited for minimizing supervision of the operations environment.

Although an HP 3000 data center can be setup without a full-time operator, certain tasks require operator intervention:

1. Starting the system
2. Monitoring the execution of jobs and sessions
3. Managing peripheral devices
4. Controlling the spooling of input and output files
5. Distributing operator capabilities among standard MPE users
6. Backing up system and user files
7. Shutting down the system

I will address each of these areas and discuss software techniques and hardware to minimize manual effort.

STARTING THE SYSTEM

The Series 37 and Micro 3000 ease startup by having a simple keyswitch, built-in system clock, automatic initialization and automatic warmstart.

With the introduction of MPE V/E T-MIT, the various system parameters for all HP 3000's can be automatically assigned for each type of startup - warmstart, coolstart, coldstart, etc. This is done by placing commands in the SYSSTART.PUB.SYS file. This reduces the operator tasks associated with the startup procedures. A description of the SYSSTART command can be found in Volume 2, Issue 3 of the HP Communicator.

Hardware and software is available to recognize a system crash, automatically telephone the system manager, request the type of startup, and perform the indicated startup procedure.

An automated method of reducing system problems and thus the need for constant supervision is the utilization of Hewlett-Packard's Predictive product, which can be scheduled to automatically perform system diagnostics and notify the HP response center of any problems.

MONITORING THE EXECUTION OF JOBS AND SESSIONS

The tasks of creating a schedule and streaming scheduled jobs can be automated with scheduling software. Job scheduling rules are entered beforehand and a single daily jobstream begins dispatching. This eliminates the tedious tasks of manually tracking jobs to determine successful completion and handling the complex dependencies of production. System users can also schedule their own jobs and can be assigned a unique job queue. In this manner the automated dispatcher handles dependencies that were previously manipulated by the operator through altered jobfences and job limits.

Other aspects of job maintenance still require manual intervention. Someone needs to be responsible for correcting and restreaming jobs which terminate abnormally. On-line reporting capabilities may be used to periodically track job status. It is also possible to not only notify a remote console of abnormally terminating jobs, but also to dial-up and notify a specified person of the occurrence via voice synthesis. In this case, periodic monitoring is minimized and job maintenance and modification is handled by those designated on a job-by-job basis.

Another area of job monitoring is launching jobs once a vague activity is finished, such as "Did accounts payable complete successfully?" or "Is order entry finished for the day?". Scheduling software may be used to send run book questions of this type to the console. A positive reply will produce

automatic launching of the dependent jobs. The software can also forecast when these activities will be performed. This will significantly reduce continual monitoring of the system, making way for specific and predetermined activities.

BACKUPS AND MANAGING PERIPHERAL DEVICES

Common tasks requiring operator intervention are the physical mounting of tapes and the replies to tape requests. These functions are typically performed for the purposes of backing up the system, restarting the system, or transferring user data. By using the HP 35401A Autochange cartridge tape drive, up to eight 67-megabyte tapes, contained in a single removable magazine, may be mounted at one time and automatically accessed in sequential order. This tape drive may be configured for Auto-Reply so that automatic replies will be issued to tape requests eliminating a common operator activity. With this configuration system backups may be scheduled and performed without any operator intervention. For more information on Auto-Reply refer to the T-MIT and UB-DELTA-2 HP Communicators. Users' tape requirements should be rescheduled as often as possible. When automatically scheduling a day's production it is possible to determine the daily tape requirements. Using scheduling software the order of production tape requests can be predicted and enforced, so tape usage will match the order of tape cartridges loaded in a magazine. Perhaps tape mounting can then be reduced to the mere activities of selecting and loading production cartridges into the magazines when necessary. Consider, for example, a data center operating with a single 536 megabyte "production" magazine and one "system backup" magazine. This could reduce sixteen or more operator-supervised tape mounting activities to two scheduled magazine loads.

Loading paper must still be done on an as needed basis. Methods to reduce this activity include scheduling output from lengthy reports to be printed at a certain time, deferring output, and issuing the command SET $STDLIST=DELETE at the end of a jobstream to reduce unwanted standard lists.

SPOOLING activities are typically tied to printing activities and should be scheduled as much as possible for actions like loading special forms or labels. With Hewlett-Packard's new Easytime menu-driven interface, user and operator spooling activities such as deferring, deleting or releasing printouts, altering print priority, and stopping and restarting the printer may be performed by non-technical help.

DISTRIBUTING CAPABILITIES or commands which are generally restricted to the operator can further reduce operator intervention. These features may be preset using the SYSSTART file discussed earlier.

SHUTTING DOWN THE SYSTEM. Due to the seriousness of this action, it is recommended that system shutdown activities - warning users, closing communications lines, storing off spool files, and issuing the =SHUTDOWN command are performed manually. However, the HP Easytime product simplifies these actions with its menus, on-line help and warning messages, so a fairly unsophisticated user can perform a shutdown safely.

FUTURE DEVELOPMENTS

What future technical advances may assist in automating the data processing environment?

Important developments in both hardware and software may be anticipated. The ability to store ever greater amounts of information on smaller media may obsolete the tape drive, just a terminals have replaced the card reader. Compact disks are available with hundreds of megabytes of data, and prototypes

of read/write optical disks have been produced.

As computers increase their capacity to process greater amounts of information, the field of Artificial Intelligence will become more accessible. Expert systems may be employed to accept a range of information and provide a response based on rules. Many of these rules must be learned through experience, reasoning, and intuition which are difficult to program. As these barriers fall the computer will be able to accept ambiguous, verbal instructions, interpret, and execute those instructions - possibly even making suggestions. In this environment, job recovery could be automated. An expert system would be given recovery instructions and it would perform appropriate actions.

Developments in printers have made the user less dependent on the centralized data center for hardcopy. This is a result of the distribution of printers into the users' environment. A few hundred dollars provide an adequate dot matrix printer while a few thousand dollars can purchase a high-quality laser printer. Just a few years ago, these machines would have cost an order of magnitude more, if available at all. In the future, every person may have a personal printer, copier, and fac machine in lieu of stationary. As portable data storage media grows in capacity, most hardcopy may be replaced by a floppy or card with universally available, portable viewers.

All these developments will generate more independent and autonomous users. These users will have control over their own computers and peripherals, using the centralized computer for its CPU power, and less for its printing and offline storage facilities.

In conclusion, a data center can be run without a dedicated operator by automating many tasks, forecasting and scheduling manual tasks at predictable

times, and using available software and hardware to monitor the system and notify people of ad-hoc needs. Manual tasks may be simplified to such a degree that a non-technical person may perform them instead of a trained operator. As computers and peripherals become more sophisticated, many of these tasks will be automated, eliminated, or distributed to the end users. This does not mean that large, complex data centers can be completely devoid of operators. As long as human experience, reasoning, and intuition are necessary for their operation, operators will be needed.

CIM - THE FUTURE IS STILL TOMORROW
TERRY H. FLOYD, CPIM, CDP
BLANKET RESOURCES

THE WALKIN', TALKIN', EATIN', SLEEPIN' MFG BLUES

Born to workin' folks in a Company town;
Grew up early, fightin', saw my best friend drown.
Raisin' hell an' playin' football 'til I broke some ribs;
Went to the Army to avoid workin' and baby cribs.

Purged fightin' from my system in the War with Hanoi;
Saw things I wish I hadn't, Boy, Oh Boy!
Started screwin' up the system, left and right;
Lots o' boys did it, not smart, not real bright.

Gotta job with my brother after I got back;
Didn't need no schoolin', thought I had the knack
For singin' and playin' and drinkin' my fill.
Didn't owe nobody nothin', didn't owe no bill.

So I worked in the factory in the daylight
And stayed up late at night:
When I first got started, thought I'd be a star
Singin' country songs and playin' the guitar.

CHORUS:

Workin' long hours, earnin' my pay.
Ev'nin' news says Japan's takin' it away.
Cause they can do it better than the American Way.
We gotta learn to save that dime
By doin' it right the very first time.

The job was no fun cause the paperwork's all wrong
And the days dragged on and on and on so long.
The clerks were always behind, everything so slow
That the orders were late or didn't even go!

We were generatin' rework as fast as we could.
They bitched about the O.T. like they always would.
Could see I was gonna live my whole life that way,
Screwin' up, livin' for nights, sufferin' the day.

Kept it up a couple o' years, thought I'd never get a break.
More I tried, more I could see it was gonna take.
Got a promotion at the plant, even got a raise,
Tables finally turned, started a new phase.

But it wasn't like that Miller Beer ad,
Deadline slipped; I lost the friends I had.
The union said we'd strike if we didn't get our way,
So they closed down that division, our game they wouldn't play.

CHORUS

Thought I'd hit the bottom, tired of holdin' them at bay,
When I started goin' with the waitress at the corner cafe.
She paid my way through college: MBA, CPIM.
Went to work for the same company, dreamin' CIM.

So the company got a new computer system
And they stuffed it down our throats, on a whim.
Big boys gonna change us, gonna go to class:
Learn how to work smart, kick the competition's ass.

And we all had to study, learnin' how to win.
Bosses stayed at Hilton, we at Holiday Inn.
They say we're all gonna tighten our belts,
The number of notches differs with the hand you're dealt.

The stock market's up and the dollar's down,
Thing's gonna get better next time around.
Lost my wife from workin' too hard.
Maybe I oughta quit, be a security guard.

CHORUS

They say the next job's gonna be the one.
It'll put us in clover then we'll have some fun.
Tho' they say it's gettin' better, we know it ain't true.
The more we get done, the more there is to do.

And those guys down on the ladder a rung or two
Screwin' up the system like I used to,
Are doin' it to me now, but I understand.
Think I'll go to the beach, see if I can get tanned.

And the perfect plant of which I dreamed
Gets farther away, not like it seemed.
The people are the system and sometimes its not controlled.
How do you get a handle? How do you get a hold?

I'm thinkin' of a system that thinks for me...
And all my people, it would help them be
A part of the whole process, not just a cog
In a big computer system, sinking in the bog.

CHORUS

Spendin' too much time tellin' it the same things.
For what we type in, know what it brings?
Reports of the same stuff that we put in,
Neatly typed and sorted, where's it gonna end?

It's time we got a lot more back
Than lists of the very same old facts.
We read about the promise of machines that think,
Remember where we screw up, fix it faster than a blink.

The company's gotta change 'fore things get better,
Products to market faster like those foreign go-getters.
Programs too hard to learn and applications don't fit.
There's too much paper and too much bullshit.

So we won't have to work and fret 'til we die
And have no time for family, Oh me, Oh my!
Yeah, I'm dreamin' 'bout tomorrow, but it may never come.
Cause what the bosses know of MFG will fit upon my thumb.

Chorus

ABSTRACT

A large percentage of Hewlett-Packard's computers are used in manufacturing companies. These systems, ranging from small shop-floor controllers to corporate "mainframes", are someday (somehow) supposed to tie together in Computer Integrated Manufacturing (CIM) to run the factory of the future.

CIM has made great progress in shop floor automation, but what about the MRPII systems that will supposedly "control" the overall process? If most of the companies using MRP can't trust it to order parts automatically, how can they allow "the system" to run the company?

Today's manufacturing user spends too much time telling the system what's going on (imputing Sales Orders, Work Orders, Purchase Orders, etc) compared to what the system give back (lists of Sales Orders, Work Orders, and Purchase Orders). MRP explodes the Bills Of Material and back schedules dates; more advanced systems even do availability checks at order entry. But, a really automated system would tell the customer if a Sales Order was going to be late because a component for a Work Order three levels down was going to be late due to a snow storm in Idaho or a teamsters strike in Cleveland. Todays' systems don't even tell the order entry user about the component's shortage, much less why.

Manufacturing systems need much improvement before they can become the "mastermind" of CIM. They're going to have to be a lot smarter than they are now.

CIM - THE FUTURE IS STILL TOMORROW

Terry H. Floyd, CPIM,CDP
BLANKET RESOURCES

The MRPII (Manufacturing Resource Planning) systems used in manufacturing companies today range from loosely related files and programs to integrated databases with sophisticated, well documented subsystems. There is little correlations between whether the system was purchased or developed in-house and its quality and usefulness to the company. One thing is clear - no one system available today does all the MRPII functions well enough to be the ultimate controller of Computer Integrated Manufacturing (CIM).

Each of the good ones has several outstanding features, but because of the massive amount of computer code required to perform all of the major functions, the systems lack consistency. Even in the primary functions of the very best and most expensive packaged manufacturing software, differences in the user interface are apparent. This must be because so many different programmers, with different styles, over many years, have been involved with their development. When we consider the span of CIM from design engineering, CAD/CAM and shop floor automation to A/R, A/P, G/L and decision support systems (and everything in between), we see a diverse and sometimes confusing tangle of methods and approaches. Throw in third party report writers, spread sheets, graphics generators, word processors, and all that PC software and one wonders how the users can cope.

Great strides are being made in the battle to define, perfect, and integrate the islands of automation. Considered separately, there are some nearly perfect subsystems for specific applications. Yet, the future holds promise for more than just tying them together and reducing retyping and re-entry related errors. More standardized user interfaces and consistency across different subsystems will help, but the major advantage of CIM is that the information in one arm of the system is available to and used by the other parts of the system.

Today's systems don't make very many decisions. They are just beginning to provide "decision support". The systems "know" a lot about how manufacturing systems work - "standard" manufacturing software - but, they don't "learn" fast enough. It's mostly embedded in the program code and data structures, not the data. It has always been said that people are the most important part of the system. The programs will have to become more "alive" before CIM can become a reality.

The speech that accompanies this paper will discuss twelve real-world examples of advances that must be made. Following is a summary of these concepts.

1. One of the challenges facing American manufacturers is getting products from design to production faster. There are several improvements possible in the Design Engineering to Manufacturing Engineering interface. MRP should also look into the future product plans for component ordering information and to assist engineers in choosing parts.

2. Many opportunities to decrease paper and smooth document flow should be investigated. The elimination of hard copy Purchase Orders, Invoices, and checks is possible. Reverse integration of standard interfaces such as Purchasing and Receiving to Payables and Invoicing/Credit activity to Inventory can be accomplished with intelligent integrated electronic mail.

3. Inspection and test equipment should not just accept or reject parts but feed a Quality Control data collection systems. This information could be used by a Preventive Maintenance system so that equipment calibration can be more than just a periodic schedule. It should also be able to stop the line when quality get out of control.

4. Repetitive manufacturing and JIT systems must be self correcting. If planners want to build 500 units per day, but production only reports 450 for Monday, the system should be able to figure out what to do with the 50 unproduced units. If production cycle times are to increase, the standard backflush transaction should be able to run from inside the finished goods shipping transaction.

5. Related activities need to be recognized as such. Labor collection systems must feed not only production efficiency files, but also Payroll and G/L. Salesman's commissions should be calculated during the Sales Order, Invoicing, Cash Receipts, and Credit Memos transactions and feed A/P vouchering on-line. Purchasing's Return-to-Vendor transactions should update not only vendor quality, inventory, and Purchase Order files, but also automatically process A/P's Debit Memo.

6. MRP, net change or regenerative, falls short of the real need. RTMRP™ (Real-Time MRP) must be available in on-line lists and during Order Entry. This handles the problem mentioned in the Abstract above.

7. The Master Production Schedule and MRP forecasts must be automatically consumed by actual customer orders through an intelligent interface. This is not a trivial calculation, but must be controlled by fields on the Sales Order or Quotation.

8. Finite load Capacity Requirements Planning and the automatic re-prioritization of production in the shop must be accomplished. Better and more easily updated Rough-Cut Capacity Planning systems must be developed.

9. Better Decision Support Systems will perhaps provide the biggest gains. True artificial intelligence and expert systems need to be "designed in" from the ground up and top down; not be merely reports of highly summarized existing data.

10. More efficient archival of details from live databases will help speed up the processing of most subsystems. All subsystems must be able to access the on-line as well as archived data.

11. Better real-time integrated error correction is needed in all areas. If CIM is to integrate all the islands, error correction must go into all the affected subsystems.

12. On-line help must become on-line assistance. If a user in finance is to relate to data in engineering, tools must be provided to bridge the jargon and data barriers.

These are samples of some of the ideas which must be in place in order to fully automate manufacturing. Many of these ideas would also be applicable outside of manufacturing companies. While some of these are probably not new ideas, we are waiting for someone to put them together with a consistent interface and make life more tolerable for future users.

RTMRP is a Trademark of Blanket Resources.

DATA INTEGRITY AND RECOVERY

CAROLIAN SYSTEMS INTERNATIONAL INC.
3397 American Drive, Unit 5
Mississauga, Ontario
L4V 1T8
CANADA

Database failures and resultant recovery efforts cost HP3000 users thousands of dollars every day in lost processing time and inconvenience. While this paper provides a discussion of IMAGE failure types and various methods of recovery, it also intends to educate the reader as to how some basic database design and implementation procedures can act as proverbial "ounces of prevention" - protecting you and your company from having to needlessly exist and suffer with logically and physically broken databases.

IMAGE FAILURE MODES

A quick review of IMAGE failure modes reveals that such common occurrences as system failures or hangs, disc media failures, datacomm line failures or application failures can all bring database processing to a screeching halt. Failures can also result in physical or logical damage to the database, with physical failures resulting from having bad data on disc (filesys), broken chain pointers or inconsistent root files. Logical failures can result because of missed updates, puts or deletes or missing delete flags. Whatever the cause, there are several standard and some new ways to repair the damage.

CLASSIC IMAGE RECOVERY

A standard method of IMAGE recovery is to restore your most recent copy of your database and forward recover using DBRECOV. However, there are some major deficiencies with this method of recovery. First and foremost, the process is extremely time consuming as it keeps users away from productive processing.

Also, DBRECOV uses a technique of recovery known as staging, whereby the restored DB is updated from the log file via staging files. The problem with staging is that large numbers of transactions can be ignored if an "end" is not found, resulting in these transactions not being applied to the database. The result can be a great deal of time and effort spent on forward recovery, with no guarantee that the recovery will be complete.

INTRINSIC LEVEL RECOVERY - ILR

If a failure occurs during an actual DB intrinsic such as updating, ILR can ensure physical integrity of your database by undoing the intrinsic. The problem with "undoing" is that with IMAGE databases, in some instances the log file and the database may not agree! Improvements made to TurboIMAGE have alleviated this problem.

TURBOIMAGE RECOVERY

With Turbo, ILR will complete the intrinsic call so that the logfile and the database agree, as opposed to just "undoing" it. Turbo allows you to forward recover with DBRECOV as does IMAGE, but it also allows you to initiate a rollback recovery.

Rollback recovery is a more timely method of recovery as it eliminates the need for a DBrestore and to reapply logged transactions to the database. Rollback recovery allows users to bring their current database up, and back out the last incomplete transactions, while complete transactions are left in place.

The use of ILR and Rollback recovery will generally ensure that more data is recovered than is possible with roll forward techniques. This is due to the fact that ILR with rollback recovery requires physical logging.

Physical logging ensures that the changes to the database are recorded and written to the log file as they occur. This prevents the log record from remaining in memory where they can be lost in the event of a failure.

Despite the time savings that can be realized with Turbo's newer recovery features, neither these or IMAGE recovery procedures are of assistance with another common occurrence that results in logically broken databases - program aborts.

RECOVERING FROM PROGRAM ABORTS

Programming bugs, user errors and datacomm line failures are just a few of the occurrences that can result in a database becoming logically corrupt. To date, HP3000 sites have had to live with the fact that once their databases have become logically corrupt, that they have to endure this inconvenience until a full recovery procedure can be initiated.

PROBLEMS ASSOCIATED WITH ABORT RECOVERY

Again, user downtime is the penalty that must be paid as users have to terminate, partial transactions are deleted or completed, and then users are allowed to access the machine again. However, if strong locking is not in place, the transaction interaction that has been rolled out can inadvertently undo a completed call. The real solution to this dilemma is to have a "net change" rollback. This is currently unavailable, as a "net change" rollback requires a detailed and intimate knowledge of the application.

SOLUTIONS - HOW TO MINIMIZE RECOVERY HEADACHES

The benefit of such facilities as DBRECOV and Rollback recovery can be greatly enhanced if you implement the following safeguards.

1. Turn on logging - despite persistent misconceptions, logging **does not** significantly degrade the performance of your machine. If you are not logging you have precluded yourself from virtually all methods of recovery.

2. Use Begins and Ends - Without DBbegins and DBends, by definition no logical transactions exist. Therefore, database logical integrity is impossible to determine. The best that can be done is to provide for an audit trail of physical transactions.

3. Strong Locking - Some method of strong locking should be implemented. Without strong locking, a transaction can interact with another transaction before it has completed, thereby making the result of a rollback recovery questionable.

4. Turn on ILR - Turning on ILR will ensure that your database will always be physically intact.

Implementation of these four key points is crucial if you are to ensure database integrity and ease of recovery for your company. They can result in tremendous reductions in user downtime and the time spent on recovery procedures. There is however, another alternative method of database recovery, that when implemented with the aforementioned safeguards, will render downtime due to the initiating of recovery, or the existence of logically corrupt databases due to program aborts, a thing of the past.

AN ALTERNATIVE - DYNAMIC "ROLLBACK"

A facility that provides a dynamic rollback, will actually undo an aborted transaction as the abort occurs. This "real time removal" of an aborted transaction will result in your database always being logically intact. Without the existence of incomplete transactions in

your database, it would also be unnecessary to have to take the system down to initiate a cleanup.

Such a utility does exist, and is actually one product for the HP3000 from the Carolian Systems Research and Development group. Known as INTACT, this product provides these major capabilities which have been previously unaddressed and unavailable to HP3000 users.

Additional information on INTACT and other Carolian Products is available from:

Carolian Systems International Inc.
3397 American Drive, Unit 5
Mississauga, Ontario
L4V 1T8
CANADA
or call (416) 673-0400

SYSTEM RESOURCE PLANNING
(TAKING OUT THE GUESSWORK)

Dale L. Folkins
Carolian Systems International
5055 Avenida Encinas, Suite 210
Carlsbad, Ca 92008

Since the first HP3000 system was shipped to an anxiously waiting customer, capacity planning has been a topic of discussion by both Hewlett-Packard and their user base. The primary tools for this "Resource Planning" task have evolved over the years from simple self-analysis tools to very sophisticated modeling products. Some of these tools have been supported by HP and/or vendors; others have shown up on user-contributed library tapes. Some have been rather "friendly and easy to use", while others have caused some very painful side effects. But, for the most part, none of the tools have taken all of the "guesswork" out of planning the future capacity of your HP3000 resources.

Realizing that system performance is a topic that few sane individuals would say deals with absolutes, there are still some basic guidelines or "rules of thumb" that are generally accepted. The mystique of system performance analysis has been inflated over the years by the concept that only highly-trained "specialists" can interpret the internal bits and bytes of MPE. Certainly, the subject requires an indepth knowledge of MPE, but there are also many indicators that an average system manager can use to analyze a system's resources and its future requirements. There is also a new tool that bridges the gap between the average system manager and the "system specialist".

A SHORT HISTORY OF PERFORMANCE TOOLS

About seven or eight years ago, the undocumented "measurement interface" was imbedded within the MPE operating system. Prior to its inception, performance tools needed to gather information resources from various tables of MPE. These tools were strictly of the unsupported variety: overlord, son of overlord, son-in-law of overlord, mother of overlord, etc.. The information was fairly reliable but support was an issue, along with training (usually provided by your SE - off the record). A "system specialist" also had MPEDCP or the data collection program that could be cold-loaded onto your system, log performance information to a tape drive for a few hours and provide data to be analyzed by MPEDRP (data reduction program). Usually, the specialist took several weeks to massage the data and had some home-grown add-on tools for generating graphs and reports. The end result was a report that only the specialist could interpret.

After the measurement interface came along, many "time-delay" oriented performance products came into common use. Surveyor became the most popular to many system managers, mostly, I suspect, since they received it on the latest contributed tape and didn't have to tell anyone they were using it to make conclusions about the future of their systems. Surveyor also required no training and was basically easy to use, where its HP-supported cousin, OPT/3000 was much more complex, requiring a purchase order and two weeks of training (presently reduced to one week).

The investment in OPT/3000 and training, coupled with the new level of expectations set by management, has not consistently paid the anticipated dividends. About three years ago, Carolian Systems introduced another measurement interface-based tool (SYSVIEW). Focusing on ease of use, learn-as-you-use technology and response time statistics, SYSVIEW bridged the gap between the unsupported utilities and OPT/3000, while adding new features and concepts to the performance analysis arena. All of these tools provide a "window" into the current state of system resources with statistics based on a "time delay" or "snapshot" methodology.

TIME DELAY LIMITATIONS

When a tool accesses the measurement interface (MI) for performance information, it must re-access the MI after a time delay period in order to compare readings and generate statistical data. These tools are specifically oriented to displaying the data at the end of each delay period, which makes them excellent "bottleneck" finders. They are used by a system manager at a given moment of time to find out the global resources being used and, in some cases, the load attributed to specific active processes. Under dedicated use, the system manager would need to constantly watch the system through these tools to gather data on resource usage.

Both OPT/3000 and SYSVIEW provide an option to allow more manageable collection of resource data through logging. By setting the log file destination, a user can capture resource data at each time delay. The shorter the time delay, the more frequent a log record is written and visa versa. For SYSVIEW, a user can also log process data at each time delay, thereby greatly increasing the size of their log files. When a performance tool begins logging, it leaves the "bottleneck finding" realm and enters the domain of "System Resource Planning". With logging comes the ability to automate the reporting of historical data and, with historical data available, we are able to derive trends for the future.

WHO DRAWS TRENDS - EXPERTS ONLY?

The concept that only "experts" can draw trends from reliable historic data is really a challenge to the reporting of that

data. For example, last year, I test-drove a 1986 "Indy 500" model Corvette. To say the least, I loved the car. It had all the gadgets that a "teckie" loves -- LED gauges of every kind: speedometer, odometer, tacometer, fuel, oil pressure, miles per gallon and Trip! What really impressed me was the MPG gauge and its counterpart, the Trip gauge. The MPG gauge showed me what my current resource management was for fuel; that is, I got instant feedback on my use of fuel. Now, I don't understand all the workings of the standard internal combustion engine, nor do I understand the technological mysteries of "multi-port fuel injection". However, I do understand driving up a steep grade, pressing a little harder (believe me, just a little harder) on the gas pedal and noticing that my MPG went from 22.5 to 14.1. When I see my fuel gauge getting into the red, I know how much gas I have left. With the Trip gauge, I can even get more help in my planning, because it reports not only how far I've come, historically, but uses its data to let me know how much further I can drive. By taking the saturation point of my resource into mind, that is, how much gas does the car hold, those GM engineers tell me, based on my current speed and average fuel use, when I need to stop for gas. We need only to define the saturation point of the resource that we are trying to measure/forecast, in order to apply the same methodology to the HP3000.

SYSTEM RESOURCE PLANNING

The Measurement Interface gathers two types of data: Global and Process. Global data is defined as the resource utilization measured in terms of the system as a whole or "globally". Collection and reporting of global statistics is relatively straightforward. Most users are satisfied with a one-hour snapshot of global data which provides accuracy to the hour of any given day. Global data is usually provided for these resources: CPU, Memory, Disc I/O, Disc Freespace, Disc Capacity and Terminal I/O. With proper care and feeding, a "time-delay"-based tool should be able to log most global resources without sacrificing a large degree of accuracy.

In order to accurately log process data, a "time-delay" product must be replaced by a "time-relative" product. By "time relative", I mean that the product must be able to gauge the contribution of a particular process to the overhead of the system into the time frame of that contribution. For example, the time line of Figure 1 shows the beginning and ending of Process "A". This process is represented in three "snapshots" - one at 2:00 p.m., another at 3:00 p.m. and the last at 4:00 p.m. If your resource planning tool only relied on "time-delay"-oriented data collection/logging, certain compromises to accuracy should be considered:

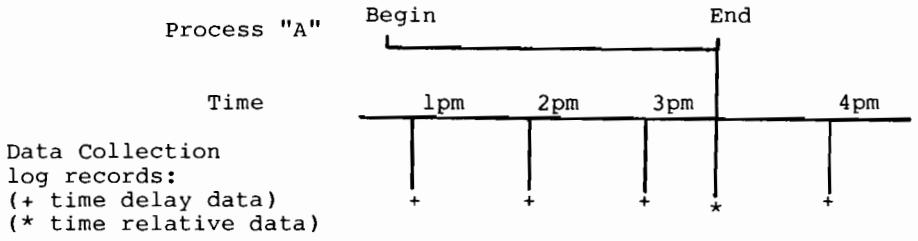


Figure 1

1. MPE reuses PINs; therefore, if a process were to start and stop within the same time delay, its resource load might be missed completely. A short time delay would tighten the window of accuracy but would greatly increase the number of log file records for longer processes. A long time delay would increase the risk of reused PINs.

2. When a user chooses to collect, log and report down to the process level, that indicates a desire to be accurate in all areas. This includes tightening of the above-mentioned "contribution" time frame.

But the choice of a "time-relative" collection/logging product isn't enough to answer both of these concerns. For more accuracy, the product must also be able to capture the process's resource usage at the conclusion of that process. Let's look at Figure 1 again: when process "A" completes between the hours of 3:00 and 4:00, its contribution to that period of time needs to be collected. With end-of-process collection, the data is available for accurately identifying the resource usage of process "A" during each of its active hours. For example: Let's say that process "A" used an average of 12% of the CPU during its life. Without utilizing accurate collection techniques, that average would be applied to all three of the time periods in which it ran. But with accurate collection techniques, we find that process "A" actually used 17%, 5% and 14% of the CPU in the respective hourly periods ending at 2:00 p.m., 3:00 p.m. and 4:00 p.m. When a system manager is trying to track the resource use of a particular program or logical device down to the hours of the day, this level of accuracy is a definite requirement.

TAKING OUT THE GUESSWORK

In any endeavor, the higher the investment, usually, the higher the return. The same is true for System Resource Planning (SRP). In terms of the investment in accuracy, the return of accuracy in reporting is well worth the effort. When accurate data collection takes place hourly over all of a system's active time, the result-

ing data available for historical and statistical analysis yields a greater return. As shown in Figure 2, the tools available have left a large hole in the investment/return ratio of SRP. The more guesswork that you want to eliminate, the larger the investment required. As Stan Freeman wrote in his May 1986 Interact article, "In choosing a capacity planning technique, various tradeoffs have to be made ... Benchmarking is notoriously expensive in terms of equipment and engineering time ..." Of all the techniques that Mr. Freeman's article addressed, Trend Analysis and Modeling seem to be the most cost effective. System Resource Planning, in my opinion, is a superset term for the strategic use of performance and capacity planning tools. In all phases of SRP, the focus is on reducing the guesswork by attempting to receive the most accurate, quantified results from the most reasonable investment in time, products and technical people skills. By using tools that make the most of accurate historical performance data and report the trends that that data indicates, a system manager knows how to best invest his other options in long-range SRP. Therefore, Carolian Systems developed SYSPLAN to fill the hole left in Figure 2.

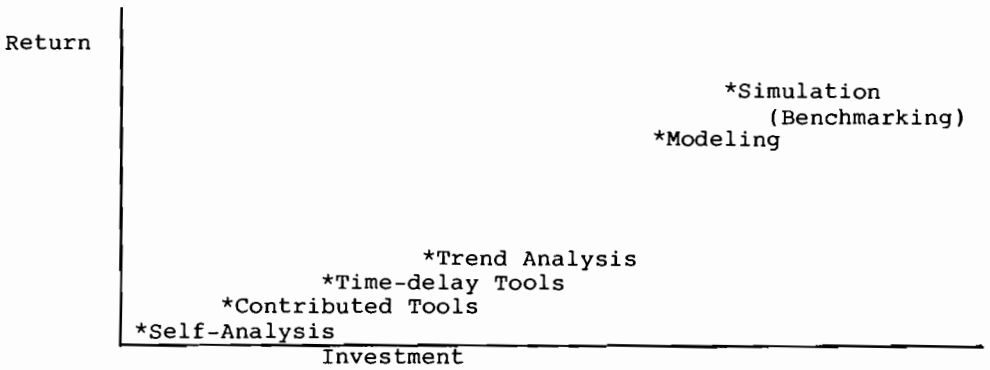


FIGURE 2

SYSPLAN - Long-range System Resource Planning

Utilizing under 1% CPU for its resource collecting and logging, SYSPLAN gives the average systems manager the ability to gain the value of high-range planning tools without the higher investment. Once data is collected on global usage and on each process that has run on a system, users can apply that data to a range of reporting possibilities:

1. On each of the typical Global resources, (CPU, for example)

(Chart 1) for the current month compared to the six previous months.

2. For a given resource, (DISC I/O) the current month compared to the three-month average (Chart 2).

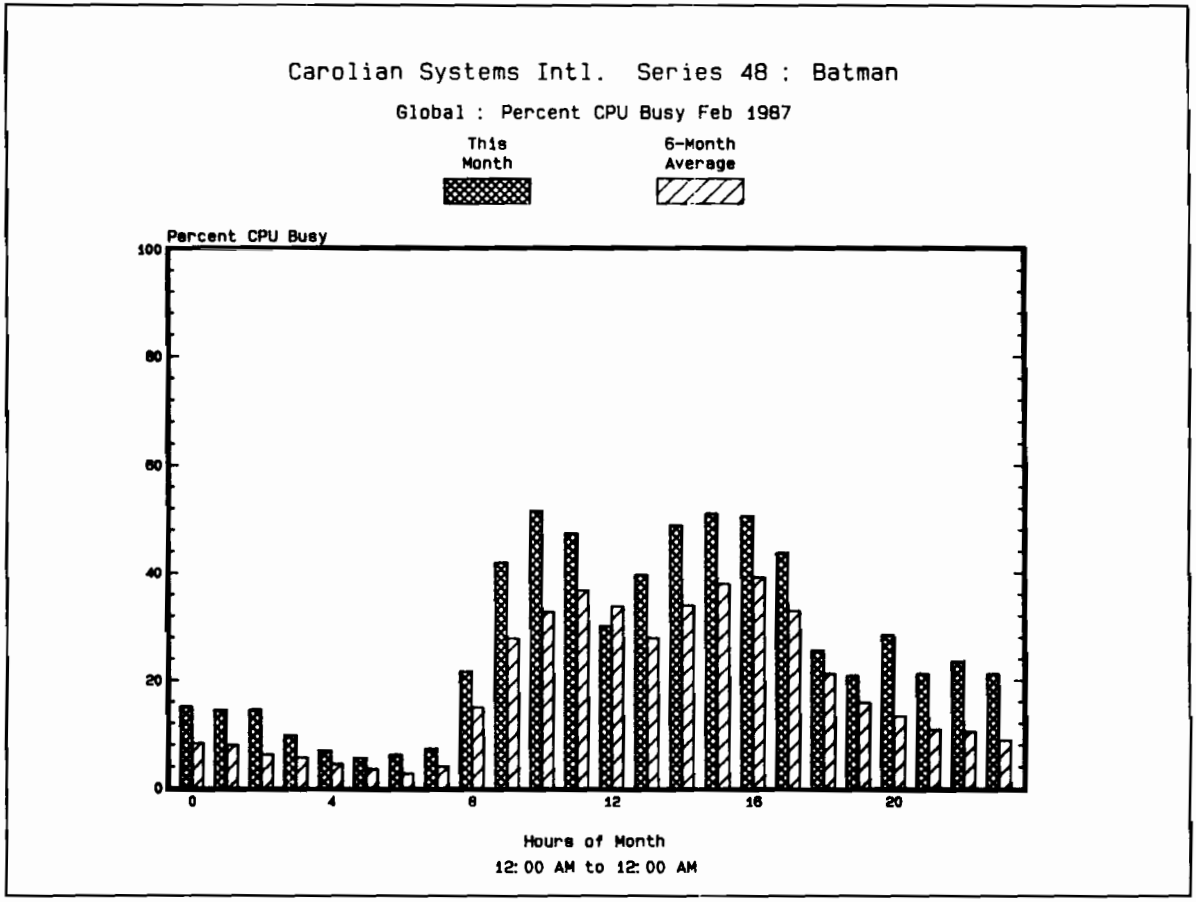
3. For a set of processes (Editors, in this case), Terminal I/O Rate for the current month compared to the average of six previous months (Chart 3).

4. Trend charts, like Charts 4 and 5, showing at least three months of historical data being statistically applied to future months, including a user-supplied "threshold" to show when the resource will become saturated.

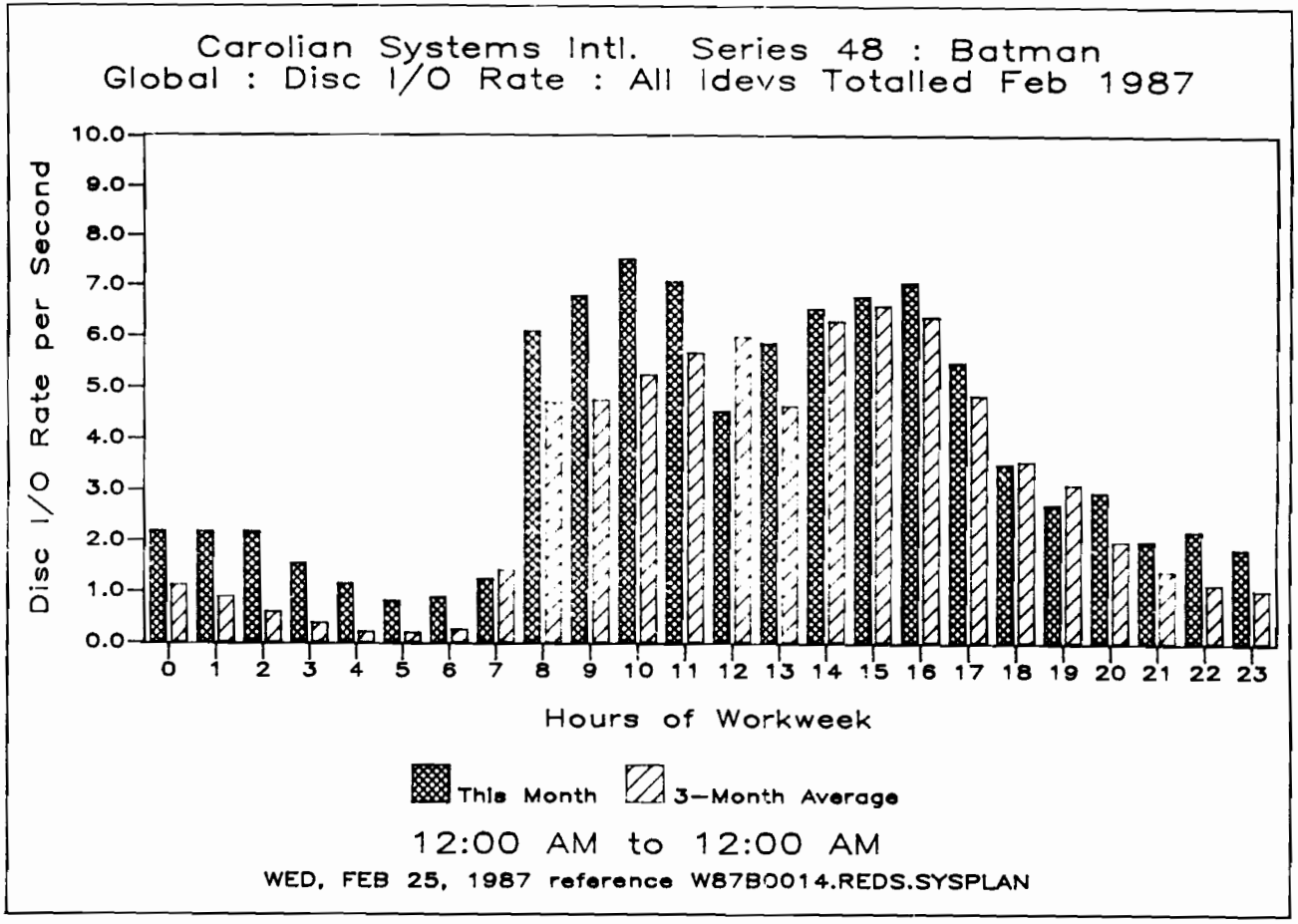
In addition to the standard resources mentioned above, tape and Printer usage is also reported. Through "Filters", several options are given for process data reports, including specific LDEV's, log-on groups, session/job names and session/job differentiation. Since the typical HP3000 installation has different shifts, reporting different hours of the day should be considered a must for any SRP tool.

SYSPLAN allows many flexible reporting methods. For more information, contact your Carolian Systems representative, or stop by our booth for a demonstration of the variety of reports available.

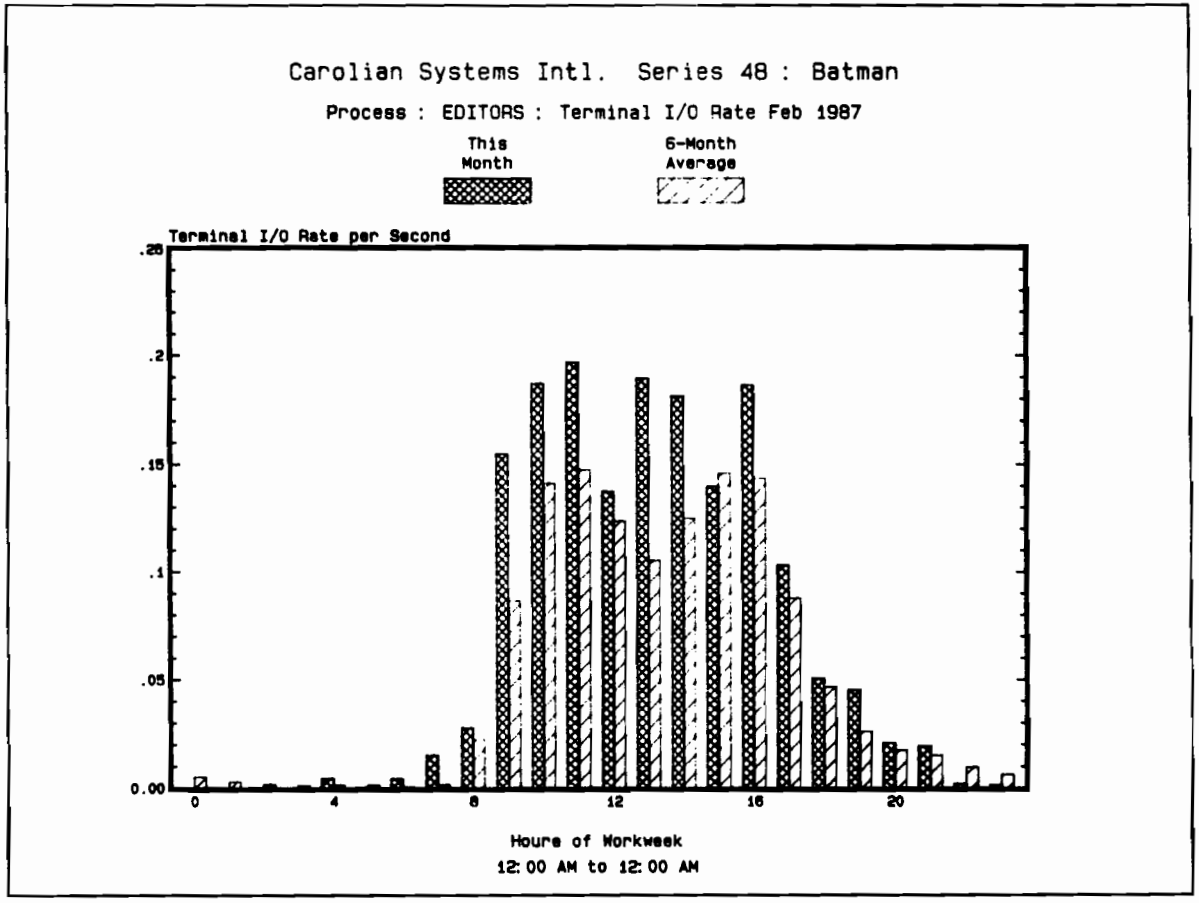
(CHART 1)


Carolian Systems Intl. Series 48 : Batman
Global : Percent CPU Busy Feb 1987
This Month
6-Month Average
Percent CPU Busy
100
80
60
40
20
0
0
4
8
12
16
20
Hours of Month
12:00 AM to 12:00 AM

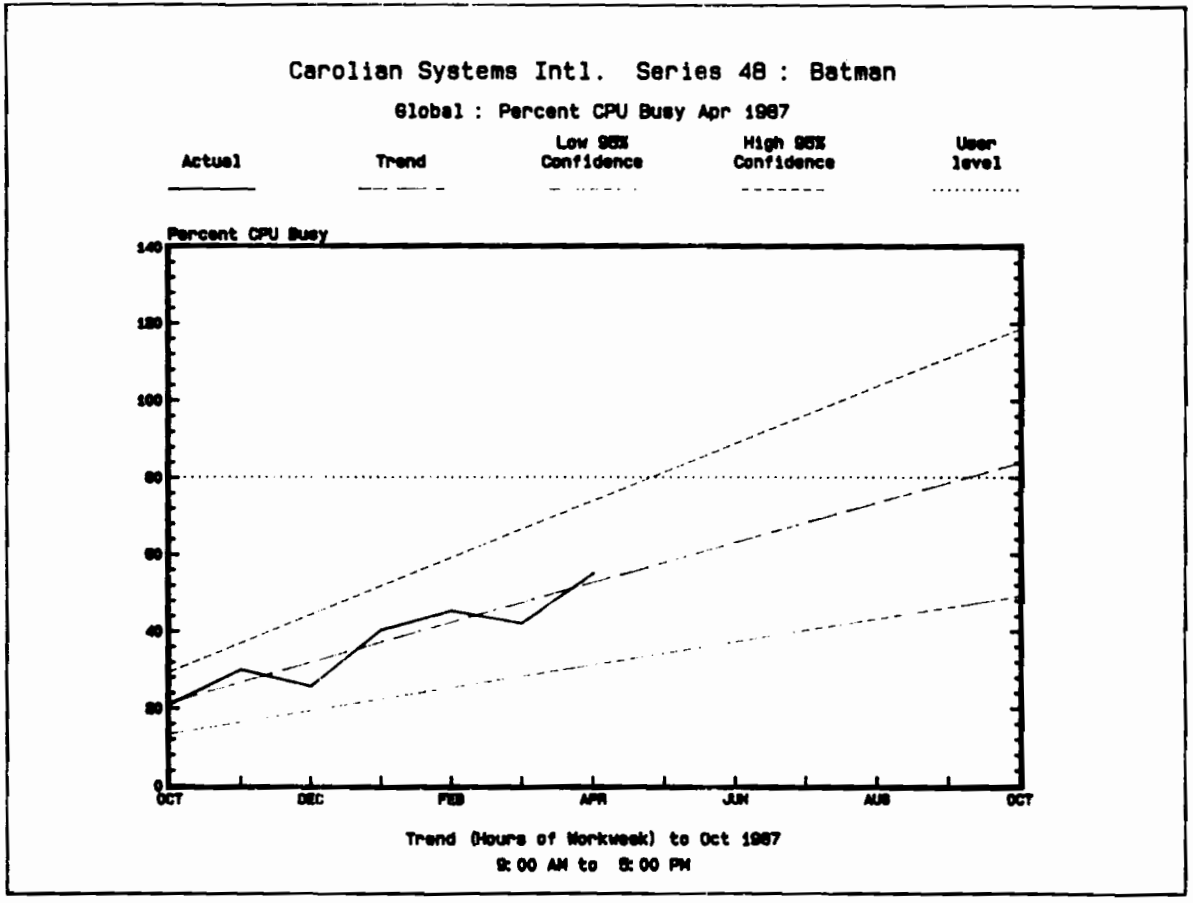
(CHART 2)


Carolian Systems Intl. Series 48 : Batman
Global : Disc I/O Rate : All Idevs Totalled Feb 1987
Disc I/O Rate per Second
10.0
9.0
8.0
7.0
6.0
5.0
4.0
3.0
2.0
1.0
0.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hours of Workweek
This Month
3-Month Average
12:00 AM to 12:00 AM
WED, FEB 25, 1987 reference W87B0014.REDS.SYSPLAN

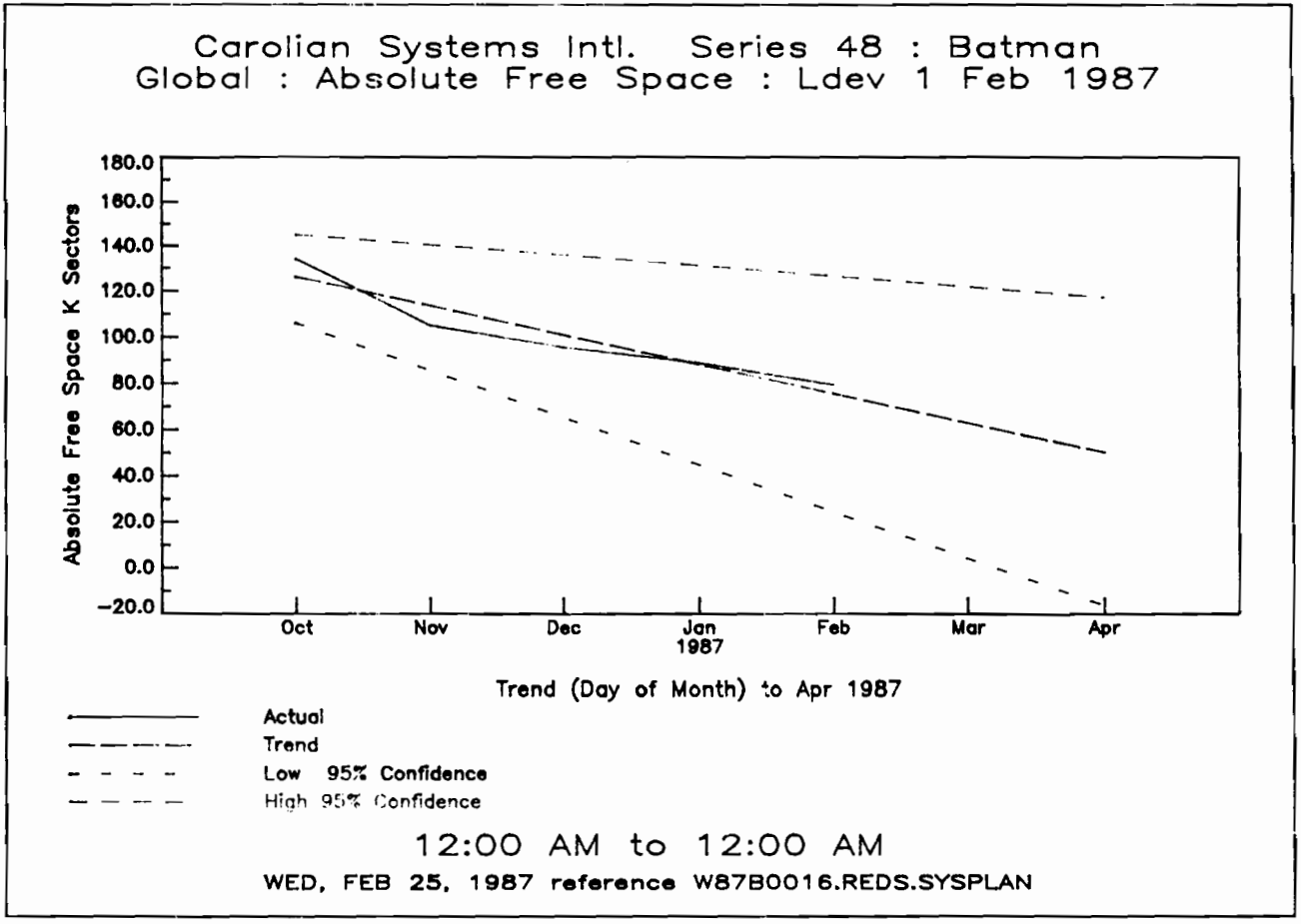
(CHART 3)


Carolian Systems Intl. Series 48 : Batman
Process : EDITORS : Terminal I/O Rate Feb 1987
This Month
6-Month Average
Terminal I/O Rate per Second
.25
.2
.15
.1
.05
0.00
0
4
8
12
16
20
Hours of Workweek
12:00 AM to 12:00 AM

(CHART 4)


Carolian Systems Intl. Series 48 : Batman
Global : Percent CPU Busy Apr 1987
Actual
Trend
Low 95% Confidence
High 95% Confidence
User level
Percent CPU Busy
140
120
100
80
60
40
20
0
OCT
DEC
FEB
APR
JUN
AUG
OCT
Trend (Hours of Workweek) to Oct 1987
8:00 AM to 5:00 PM

(CHART 5)


Carolian Systems Intl. Series 48 : Batman
Global : Absolute Free Space : Ldev 1 Feb 1987
Absolute Free Space K Sectors
180.0
160.0
140.0
120.0
100.0
80.0
60.0
40.0
20.0
0.0
-20.0
Oct
Nov
Dec
Jan
1987
Feb
Mar
Apr
Trend (Day of Month) to Apr 1987
Actual
Trend
Low 95% Confidence
High 95% Confidence
12:00 AM to 12:00 AM
WED, FEB 25, 1987 reference W87B0016.REDS.SYSPLAN

THE EVOLUTION OF INTELLIGENCE

by Richard Forsyth, VRS Consulting Inc.,
(4676 Admiralty Way, Marina del Rey, CA 90292).

"Our only hope therefore lies in a true induction" -- Francis Bacon: First Book of Aphorisms.

"It is highly unlikely that any system we can build will be able to undergo the kind of evolutionary change (or learning) that would enable it to come close to the intelligence of even a small worm, much less that of a person." Winograd & Flores: Understanding Computers and Cognition.

Keywords: Artificial Intelligence, Machine Learning, Rule Induction, Genetic Algorithms.

Abstract: The need for Artificial Intelligence (AI) to grapple with the core problem of learning is argued, and a renaissance of interest in machine learning (especially rule induction) is predicted. In particular, AI workers are urged to conquer their distaste for processes (e.g. Monte Carlo methods) which have a random element and investigate the potential of evolutionary algorithms more seriously than hitherto.

Five years ago, Expert Systems were exciting novelties. Today they are dull. Of course, the last thing users of computers want is excitement: they get too much of that when their systems break down. But programmers seek new challenges; and knowledge-based systems are no longer challenging enough for some adventurous spirits. What, then, is the next frontier -- the place where leading-edge AI programmers can look for excitement?

My personal answer to that question is: Machine Learning. This field shares many of the characteristics which Expert Systems had a decade ago. The crucial proving experiments have already been successfully conducted (e.g. Michalski & Chilausky, 1980; Lenat, 1982; Michie & Johnston, 1985); but laboratory research has not yet filtered out into general computing practice. A rich seam of valuable ore has been discovered, and its exploitation is only a matter of time.

Yet, in the field of Artificial Intelligence (AI), researchers have persisted in trying to build fully-fledged 'adult' systems which cannot learn for themselves at all. Early attempts to devise learning machines, during the cybernetic days of AI, proved disappointing, so the whole idea was dropped. Only recently has it been revived.

From time to time, leading AI practitioners admit the necessity of learning, e.g. Schank (1982):

> "A computer that doesn't learn really cannot make much of a claim on being intelligent."

But the general attitude is that learning is a small and relatively unpromising subfield of AI.

The present paper seeks to advocate a minority view; namely that there can be no genuine machine intelligence without machine learning. Its argument is that the revival of interest in machine learning is overdue, is important, and will continue until systems capable of self-improvement become the norm rather than the exception. Indeed it may well be that many AI problems (e.g. speech understanding) are so difficult that they can only be solved by systems that go through a 'childlike' phase.

To put it simply, learning is the key to intelligent behaviour -- and that is a lesson that the AI community should learn as fast as possible. It is far more than just a short-cut through the 'knowledge acquisition bottleneck' in building expert systems. Machine induction opens up the possibility of synthesizing totally new knowledge -- of automating the process of scientific discovery and creating patterns and concepts that no one has ever thought of.

1. THE ACT OF INDUCTION

In order to discuss machine learning and rule-induction, we need a consistent terminology.

Looked at from sufficiently far away, all systems designed to modify and improve their performance share certain important common features. Figure 1 is a diagram of the four major components of a typical learning system. Essentially this sketches a pattern recognizer which learns to associate input descriptions with output categories; but, as we shall see, many systems that are not overtly concerned with pattern recognition also fit into this general framework.

[Figure 1 -- A Paradigm for Learning.]

Note that the system contains a feedback loop. We can briefly describe its main components in turn, by going round this feedback loop.

The Critic compares the actual with the desired output. In order to do so, there must be an 'ideal system', as we call it, against which the system's behaviour is measured. In practice this may be a human expert, or teacher. For instance, if the task is medical diagnosis, the ideal system may be the diagnosis given by a top consultant when faced with the patient whose history is being presented to the computer as input. The job of the critic is to apportion credit and/or blame for the system's responses. It must assess deviations from correct performance.

This can be simple or complex. In a simple case it might compare

(for example) rainfall as forecast with actual rainfall. If 0.6 mm of rain fell and the system predicted 1.7 mm then it is only a matter of subtracting one number from another and passing the difference on as feedback to the learning module. In other circumstances there may be more work for the critic to do to ascertain what went wrong. For example, after losing a game of chess, it may not be at all obvious where the computer blundered. But however simple or complex the task of the critic, evaluative feedback is absolutely fundamental to the learning process. There are 'unsupervised learning' programs in existence, which do roughly the job that statisticians know as cluster analysis -- i.e. tidiying up a cluttered conceptual space -- but they cannot be said to learn in any meaningful sense: they do not and cannot improve their performance at an assigned task simply by virtue of conceptual clustering. What they do is a useful precursor to the learning process proper.

From this follows a very important point: a machine cannot learn anything unless its performance can be assessed objectively. For learning to take place, there must be an agreed criterion according to which responses or partial solutions can be scored. Most failures of machine induction arise not because the algorithm is inadequate but for one of two other reasons:

> (1) the chosen representation is not capable of expressing the knowledge needed for satisfactory performance;
> (2) the chosen evaluation measure is misleading.

The Learner is the heart of the system. This is the part of the system which has responsibility for amending the knowledge base to correct erroneous performance. A large number of learning strategies have been proposed since work on machine learning got under way in the 1950's, some of which we will examine in detail.

The Rules are the data structures that encode the system's current level of expertise. They guide the activity of the performance module. The crucial point is that they can be amended. Instead of a Read-Only knowledge base (as in most current Expert Systems) the rules constitute a Programmable-Erasable knowledge base. Obviously they must only be modified under strictly defined conditions or chaos will ensue. (Other forms of knowledge representation than condition-action rules have been used successfully, but we use the term 'rules' as a convenient shorthand for the time being.)

Finally, the Performer is the part of the system that carries out the task. This uses the rules in some way to guide its activity. Thus when the rules are updated, the behaviour of the system as whole changes (for the better, if all goes according to plan). Naturally the performer is the part of the system which varies most from one task to another, and is therefore the part of the system about which least can be said in general terms.

Two other terms need to be defined before our examination of practical learning methods -- 'description language' and 'training set'.

The description language (or rule language) is the notation or formalism in which the knowledge of the system is expressed. There are two kinds of description language which are important. The first is the notation used to represent the input examples. One of the simplest of input formats is the feature vector. Each aspect of the input example is measured numerically, and the vector of measurements defines the input situation. This is a representation carried over from the field of statistical analysis which has served well in a number of applications; but it is by no means the most sophisticated, and it is not always appropriate.

The second kind of description language is that chosen to represent the rules themselves. It should be noted that the expressiveness of the description language in which the rules are formulated is crucial to the success of any learning algorithm. It also has a bearing on how readily the knowledge can be understood, and hence on whether it can be transferred to people.

The notion of a training set is important in understanding how a machine learning system is tested. Typically there is a database of examples for which the solutions are known. The system works through these instances and develops a rule or set of rules for associating input descriptions with output decisions (e.g. disease symptoms with diagnoses). But, as Francis Bacon pointed out three and a half centuries ago, rules must be tested on cases other than those from which they were derived. Therefore there should be another database, the test set, of the same kind but containing unseen data. If the rules also apply successfully to these fresh cases, our confidence in them is increased.

This preliminary definition of terms enables us to compare learning systems in a consistent fashion.

Machine learning can be viewed from a bewildering number of perspectives -- as optimization, concept formation, pattern recognition, automatic classification, scientific discovery, programming by example and many more. But the simple common theme of

generate + test

provides a unifying principle.

[Figure 2 -- Generalized Rule Induction Flowchart.]

All learning systems, however clever they are about avoiding brute-force exhaustive search, propose new potential solutions and test those potential solutions. This means that there are two fundamental questions we can ask about an automatic induction system:

(1) how are new knowledge structures generated from old ones?
(2) how are candidate knowledge structures evaluated?

There are many possible taxonomies of the field of rule induction; but the answers to these two simple questions help to organize an area that up till now has been characterized by

somewhat anarchic eclecticism. If we know the answers to those two questions, we have grasped the essence of the learning system.

A more detailed taxonomy divides up learning systems into 128 types based on their distance from either pole of the following seven polarities:

Domain:	general-purpose	vs	specific
Method:	model-driven (= top-down)	vs	data-driven (=bottom-up)
	general-to-specific	vs	specific-to-general
	deterministic search	vs	random search
Critic:	logical evaluation	vs	quantitative evaluation
Representation:	humanly intelligible	vs	black box
	feature vector (= unary attributes)	vs	structural (= relational predicates)

The meanings of these terms will be explained during the talk. To take only the first as an example, there is a huge number of domains of application for learning systems; but the most significant binary feature of a learning system (with respect to its domain) is whether it can handle lots of different problems or is specialized for a single task.

2. THE EVOLUTION OF IDEAS

Granted that machine learning is the wave of the future, where do we look to gain hints on how to ride that wave? I suggest that we look back about 4 billion years into the past and consider the process that gave rise to the blob of grey jelly with which you are now interpreting these words -- the process of evolution. As a natural model for computer-based optimization, it has a number of advantages. It is robust, relatively well understood (at least in broad outline) and it has been extensively field-tested. But AI scientists have tended to overlook it, believing that it does not work, or rather that it could not form the basis of a workable computer algorithm.

They have based their pessimism largely on a book by Fogel, Owens and Walsh (1966) which reported a series of attempts to design learning programs on an evolutionary model. The trouble with the Fogel, Owens and Walsh experiments was that they used mutation as their primary genetic operator. Thus they restricted themselves, in simple terms, to asexual reproduction. As we shall see, mutation is not the primary genetic operator in abstract genetic algorithms, nor in the living world for the vast majority of species which indulge in sexual reproduction.

While the AI community, with few exceptions, dismissed the whole concept of evolutionary systems, workers in other domains got on with the job of building self-improving computer systems based on a Darwinian approach. As early as 1964 (before the supposedly definitive study by Fogel, Owens and Walsh) Ingo Rechenberg had employed an evolutionary algorithm in an engineering context. He used it as a way of finding shapes that minimized drag in a

fluid flow (Rechenberg, 1965). Since then his methods have evolved (to coin a phrase) and been used succesfully in aircraft design and civil engineering. They have also been successfully adopted by the West German automotive industry as a practical, money-saving design technique. (See Rechenberg, 1985.)

More recently, Rechenberg's ideas have had a spin-off in an entirely different application -- computer art. Richard Dawkins (1986) has created the 'Biomorph' system which follows a quasi-evolutionary approch and creates pleasing shapes on a graphical display screen. The Biomorph program is also used as a teaching aid in biology.

But Rechenberg is an engineer and Dawkins a zoologist. Moreover they are Europeans. AI, despite the efforts of the Fifth Generation research group at ICOT in Japan, takes its lead from North America. So most AI scientists are unaware of their work. It was not until 1975, with the publication of Adaptation in Natural and Artificial Systems by John Holland at Michigan University that a reappraisal of the promise of an evolutionary approach to learning began in the AI community; and even Holland's definitive work took about a decade to gain acceptance. Only recently has the subject become respectable again. Holland's theoretical treatment has in recent years led to the development of practical evolutionary learning algorithms (e.g. Smith, 1980).

In any evolutionary learning scheme there is a population of structures which are treated like simulated living organisms. Each of these structures defines a potential (or partial) solution to the problem under consideration. They are also used to generate new structures in ways that are intended to simulate the main attributes of biological reproduction. If the structures are rules, which is the normal case, then we have an evolutionary rule-induction system.

Selection of which rules survive longest and have greatest likelihood of 'breeding' is governed by their performance on the task. This has been called naturalistic selection (Forsyth, 1981). It corresponds to the popular phrase 'survival of the fittest' (due to Herbert Spencer, not Darwin).

A groundplan for a genetic/evolutionary algorithm can be outlined as follows.

> (1) Create an initial population of rules at random.
> (2) Evaluate the rules and if the overall performance level is good enough halt and display the best of them.
> (3) Compute the selection probability of each rule as p = e/E where e is its score and E the sum total of all rule scores.
> (4) Generate the next population by selecting according to the selection probabilities and applying certain genetic operators; then loop back to step 2.

Each pass round this loop corresponds to a single generation in a population of living creatures. Clearly generations are not so well synchronized in the animal kingdom; but despite its many simplifications, the basic genetic algorithm is a robust, general-purpose optimization procedure, applicable to a wide

variety of problem domains.

Most people think they understand biological evolution very well, when they do not, and on that basis regard the genetic algorithm as a crude, brute-force approach to learning. It is of course a highly simplified version of what goes on in nature; but it is by no means synonymous with brute-force trial and error. The reason people underestimate its subtlety is that they think only of mutation; but the key genetic operator is crossover, not mutation (which is a 'background operator' serving only to prevent the system sticking on a local optimum).

Mutation involves making a few haphazard alterations in the genetic information, and is usually (though not always) detrimental. Evolution, in nature and in the computer, is greatly speeded up by sex, because sexual crossing produces change without mutation: genetic information is chopped up and re-spliced. As Holland has proved in the abstract, and as the huge energy expenditure of animals and plants on sex should lead us to expect, the hybridization resulting from applying the crossover operator is a fundamentally more efficient way of searching the space of possible strucures than random mutation. Its power lies in the implicit parallelism of the process (coupled with explicit parallelism in the real world). (See Holland, 1975.)

The inherent parallelism of the evolutionary process should alert us to another important property of the genetic algorithm: it is tailor-made for the new generation of highly parallel computer architectures which almost everyone in the industry is expecting to revolutionize computing by the turn of the century. In other words, when the hardware boys get their acts together, genetic algorithms will be ready to take off.

Genetic algorithms are, in fact, a special class of the 'Monte Carlo' methods employed by statisticians and operational researchers on problems which defy analytical solution. Monte Carlo methods employ controlled randomness to achieve optimal or near-optimal solutions to difficult problems. But AI workers have traditionally frowned upon randomness: somehow it does not fit with the profoundly rationalist world-view in which AI has grown up -- although it was a prominent theme in the early cybernetic days of machine intelligence.

In a pure Monte Carlo method, solutions are generated at random for as long as there is time and the best solution found is kept. Each trial generates a completely fresh potential solution with no reference to what has gone before. If the problem under attack, for example, is the Travelling Salesman Problem (TSP), then each trial will create an entirely new route for the salesman, measure it, and -- if it is the shortest so far -- retain it.

This is indeed a blind seach procedure, and does justify the description brute-force trial and error. A genetic algorithm, however, modifies this basic procedure to make use of information about the distribution of values in the solution space. Sub-patterns that have been found useful are recombined in various ways. Thus the search is guided through the enormous

multi-dimensional space of possible solutions towards regions where good results have been found on previous trials. In the example of the TSP, new routes would be generated by small modifications of existing, shorter-than-average routes.

Looking at evolutionary algorithms as Monte Carlo procedures, we can usefully distinguish four levels of complexity:

(0) pure Monte Carlo methods;
(1) mutation-only methods;
(2) the basic genetic algorithm;
(3) the 'elitist strategy'.

Each level is characterized by a different answer to the question:

how is the next candidate solution to be chosen?

Method (0) simply hops from one random point in the search space to another. It takes no account of any regularity which that search space almost certainly possesses. Method (1) modifies existing structures to produce new ones for testing. It will do better than Method (0) provided that the value of potential solutions varies smoothly with their location in the search space, as is often the case. Method (2) generates new structures by recombining portions from old ones that have proved valuable. It can be shown to be more efficient than Method (1) by virtue of exploiting implicit parallelism. Method (3), the elitist strategy, differs from biological evolution in preserving the best-ever structure until it is displaced by a better one in the breeding population. It confers effective immortality on the best structure so far. Thus it approximates more closely to the brave new world of artificial genetic engineering than the classical Darwinian scheme of nature.

We have been applying something like the elitist strategy to our livestock and crops for a long time. In the near future we may apply it to human beings as well. The ethics of doing so are questionable, but we need have no ethical qualms about cutting up rule-structures in a computer's memory.

3. BEAGLE

To put these remarks into context, we shall briefly consider a package devised by the present author and based on the use of a genetic algorithm. The BEAGLE system (Biologic Evolutionary Algorithm Generating Logical Expressions) is one of the few commercially available software packages employing evolutionary rule-induction (Forsyth & Rada, 1986). It consists of six main modules.

SEED	Selectively Extracts Example Data
ROOT	Rule-Oriented Optimization Tester
HERB	Heuristic Evolutionary Rule Breeder
STEM	Signature Table Evaluation Module
LEAF	Logical Evaluator And Forecaster
PLUM	Procedural Language Utility Maker

A diagram of how they link together is shown as figure 3.

[Figure 3 -- The BEAGLE Software Modules.]

We shall concentrate on HERB, which is the heart of the system -- the program that actually performs the evolutionary induction process. It works by running through the following procedure, expressed below in 'pidgin pascal'.

```
REPEAT
  mainloop := mainloop + 1;
  runs := 0;
  REPEAT
    runs := runs + 1;
    reset(datafile);  { start at beginning of training data }
    FOR nx := 1 to samples do begin
      { for every training instance }
      { get the next sample case }
      { try all rules on it }
      end;
    scoring;  { apply the Critic }
    culling;  { get rid of sub-standard rules }
    eugenics;  { mate the survivors }
    mutation;  { clobber a few rules at random }
    tidying;  { clean up the mess }
    UNTIL (runs >= gens);  { enough done }
  savebest;  { keep the top-scoring rule }
  wipeout;    {  eliminate the variables it used from further
                 use }
  UNTIL (mainloop >= maxloops) OR (varsleft < 2);
{ dump saved rules onto a file }
```

This is in fact an outline of the HERB main program, with comments to explain what is going on. It is also the basis for figure 4.

[Figure 4 -- HERB Main Program Flowchart.]

The mating procedure picks out random subtrees (sub-expressions) from the two 'parents' and simply glues them together with a random connective. Thus if the parents were the two rules

```
((GREEN - ORANGE) > 0.33456)
```

and

```
((LANDMASS > 4) & (RICHNESS < 600.00))
```

a possible descendant could be

```
(LANDMASS > 0.33456)
```

or

```
((GREEN - ORANGE) >= (RICHNESS < 600.00))
```

where the latter involves a comparison between a numeric value and a logical value. This is resolved in BEAGLE's rule-language by treating TRUE as 1.0 and FALSE as 0.0 where context demands.

BEAGLE can be said to use a non-parametric version of the

genetic algorithm: the survival of rules is determined by their rank order of merit. Its original contribution lies in resolving the apparent contradiction between a flexible rule language modelled on that found in procedural programming languages such as C and Pascal, and the demands of the genetic algorithm for fixed-length strings. BEAGLE does occasionally come up with rules that are very hard to interpret (some might say that is part of its charm) but compared to a stretch of DNA they are as clear as crystal. Prior to BEAGLE, most computer scientists had presumed that genetic algorithms could only work with fixed-length, position-independent rule-strings, i.e. with a write-only description language. BEAGLE shows that a hierarchically structured rule language can also accomodate quasi-genetic recombinations.

BEAGLE has already proved itself in a variety of domains, from medical research to sales analysis. One of its more successful applications is reported by Spiehler et al. (1987) in forensic science, where a rule-set developed by BEAGLE on a PC has been found to discriminate among glass fragments better than standard forensic procedures. The problem they set was to distinguish different types of glass fragments on the basis of laboratory tests, including an elemental analysis derived from X-ray fluorescence. The determination of glass type (e.g. after a break-in or automobile accident) is often crucial in the assessment of forensic evidence. Their results are tabulated below. It should be noted that in every case the evolutionary method out-performed a commonly used statistical technique.

[Table 1 -- Performance Comparison (glass data).]

PC/BEAGLE versus Nearest-Neighbour Classification of Glass Fragments based on element composition and refractive index.

Source	Samples	% Correct : N.N.C.	BEAGLE
Building window (float)	70	85	87
Building window (non-float)	76	69	74
Vehicle window	17	73	94
Container glass	13	56	77
Tableware	9	67	78
Headlamps	29	83	86

'Float' and 'non-float' refer to the process by which the glass was made.

On a completely different task, the evolutionary strategy of Rechenberg (1985) has been used in structural engineering to design a minimal weight lattice frame; while the genetic adaptive system of Smith (1980) devised an optimal poker-playing strategy.

Naturalistic Selection is certainly a general-purpose learning technique. Indeed the reason why evolutionary algorithms have attracted so little attention may be that the method is too general. It has such a wide range of applications that no one knows which conceptual pigeon-hole to allocate it.

4. CONCLUSIONS

When the training set of examples is relatively free from 'noise' in the data, there are algorithms (Quinlan, 1979; Mitchell, 1982) which discover rules more efficiently than Naturalistic Selection, outlined above. However, real-world problems are more often characterized by noisy than noise-free data (e.g. weather forecasting, financial decision-making). In such areas, AI researchers should force themselves to overcome their deep-rooted prejudice against Monte Carlo methods. AI workers like logical rigour and dislike randomness. Nevertheless, they should not be blind to the virtues of controlled randomness.

Naturalistic Selection is not a panacea, and BEAGLE is only one way of performing naturalistic selection; but if your problem has

(1) a very large search space;
(2) a clear-cut quantifiable way of evaluating rule structures;
(3) formal methods for chopping up and recombining rule structures; and
(4) no obvious algorithmic/analytic solution

then a Darwinian approach to rule induction will almost certainly pay off.

My message to users is to avail themselves of these powerful but under-utilized techniques, and to programmers it is to learn how to incorporate them into their software.

REFERENCES

DAWKINS, Richard (1986) --
"The Blind Watchmaker": Longmans, London.

FOGEL, OWENS & WALSH (1966) --
"Artificial Intelligence through Simulated Evolution": Wiley.

FORSYTH, Richard (1981) --
"BEAGLE: A Darwinian Approach to Pattern Recognition": Kybernetes, 10.

FORSYTH, Richard & RADA, Roy (1986) --
"Machine Learning: Applications in Expert Systems and Information Retrieval": Ellis Horwood, Chichester.

HOLLAND, John (1975) --
"Adaptation in Natural and Artificial Systems": University of Michigan Press.

LENAT, Douglas (1982) --
"The Nature of Heuristics": Artificial Intelligence, 19.

MICHALSKI, Ryszard & CHILAUSKY, R.L. (1980) --
"Learning by Being Told and Learning by Examples": J. Policy Analysis and Information Systems, 4.

MICHIE, Donald & JOHNSTON, Rory (1985) --
"The Creative Computer": Pelican Books, Harmondsworth.

MITCHELL, Thomas (1982) --
"Generalization as Search": Artificial Intelligence, 18.

QUINLAN, John Ross (1979) --
"Induction over Large Databases": Report HPP-79-14, Stanford University Computer Science Dept.

RECHENBERG, Ingo (1965) --
"Cybernetic Solution Path of An Experimental Problem": R.A.E. Farnborough.

RECHENBERG, Ingo (1985) --
"The Evolutionary Strategy: A Mathematical Model of Darwinian Evolution": Technical University of Berlin Report.

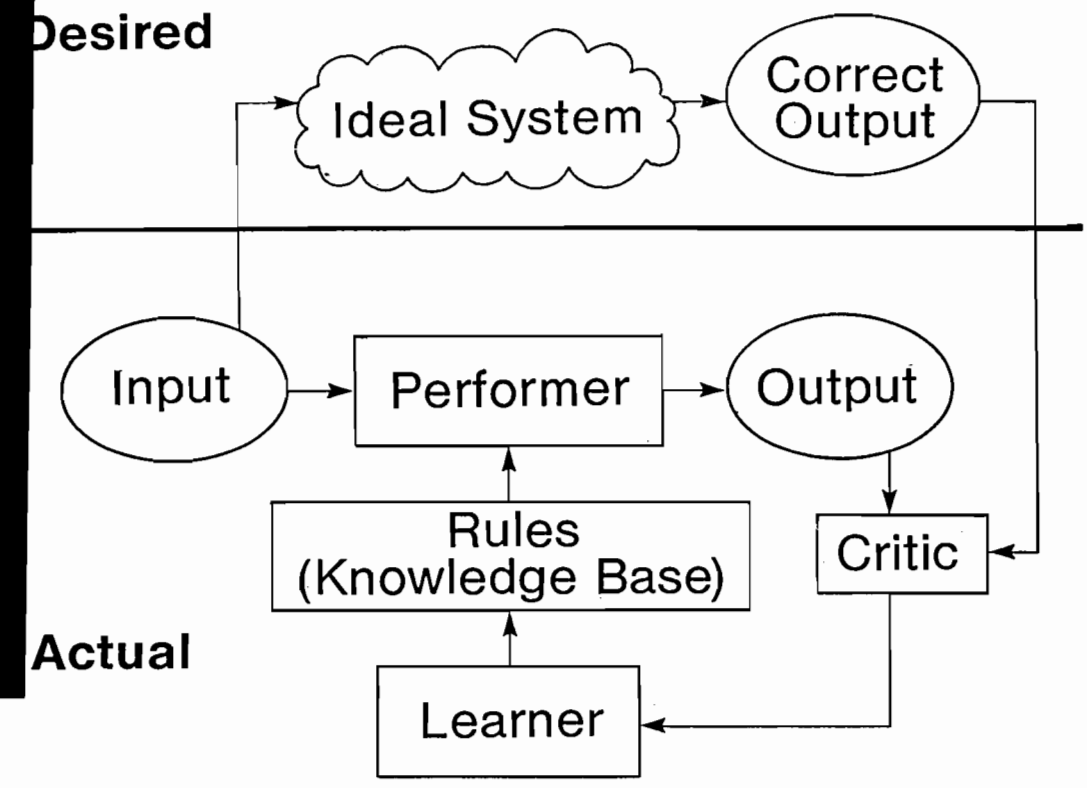
SCHANK, Roger (1982) --
"Looking at Learning": Invited Lecture, Proceedings of ECAI-82, Paris.

SMITH, Stephen (1980) --
"A Learning System Based on Genetic Adaptive Algorithms": PhD Thesis, Computer Science Dept., University of Pittsburg, PA.

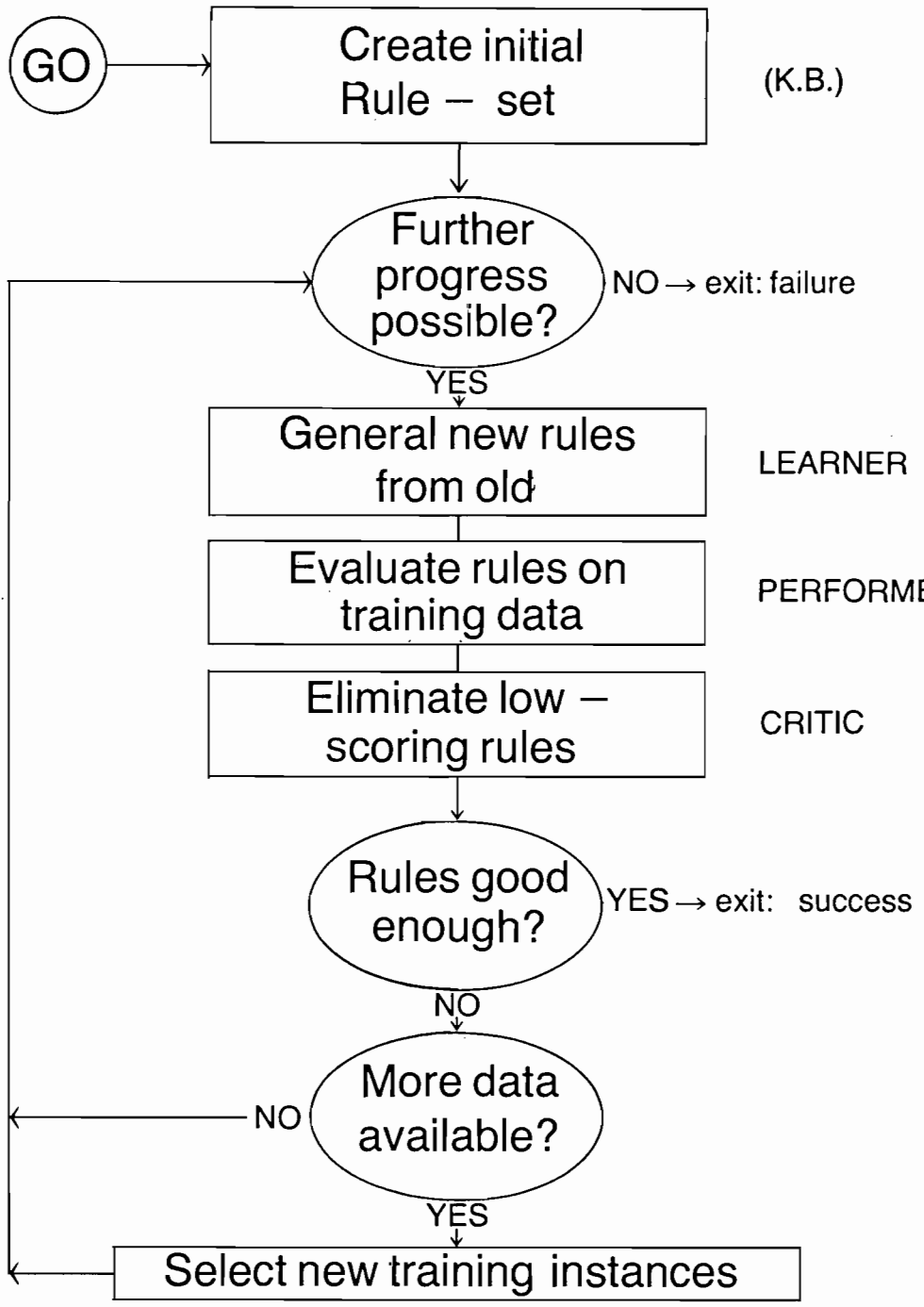
SPIEHLER, Vina (1987) --
"Aplications of Expert Systems Shells in Forensic Science": Proc. International Expert Systems Conf., London.

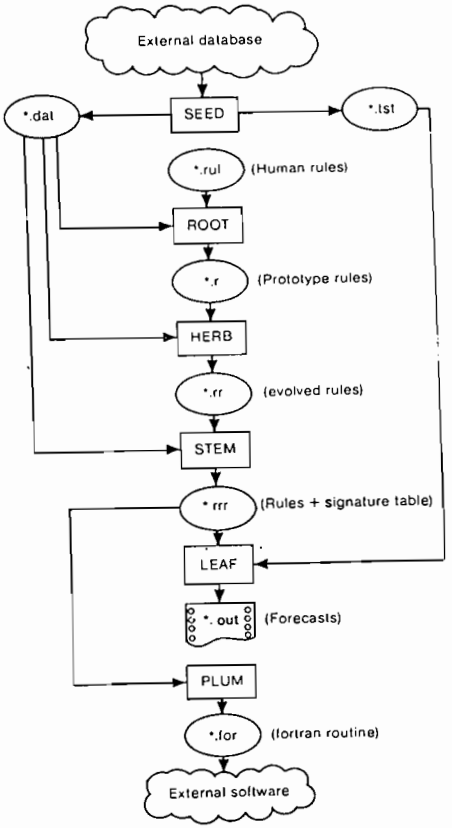
WINOGRAD, Terry & FLORES, Fernando (1986) --
"Understanding Computers and Cognition": Ablex, New Jersey.

Machine Learning Paradigm



General Rule Induction Flowchart

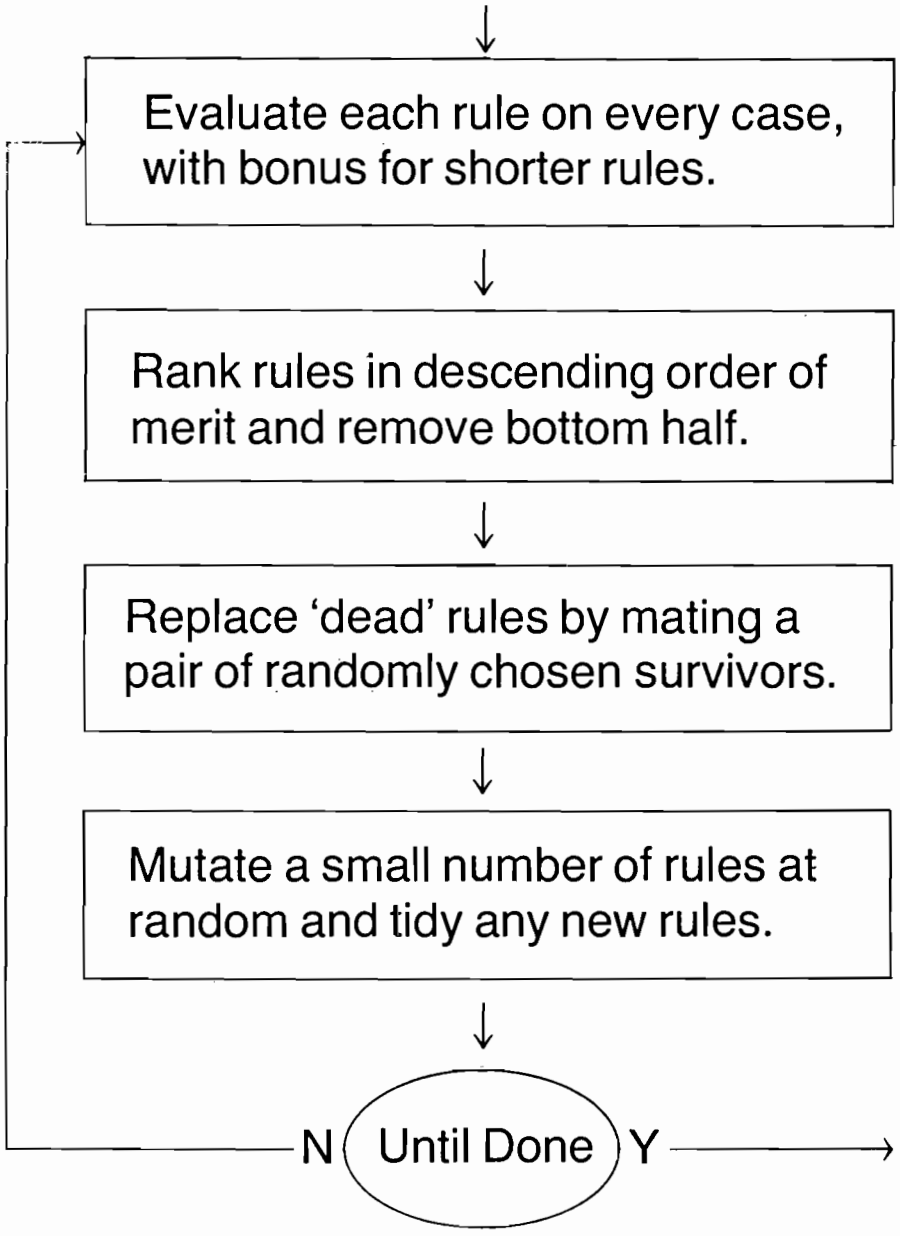




BEAGLE contains six main components which are generally run in sequence. SEED (Selectively Extracts Example Data) puts external data into a suitable format, and may append leading or lagging data-fields also. ROOT (Rule Oriented Optimization Tester) tests an initial batch of user-suggested rules. HERB (Heuristic Evolutionary Rule Breeder) generates decision rules by Naturalistic Selection. STEM (Signature Table Evaluation Module) makes a signature table from the rules produced by HERB. LEAF (Logical Evaluator And Forecaster) use STEM's output to do forecasting or classification.

Finally PLUM (Procedural Language Utility Maker) can be used to convert a BEAGLE rule-file into a language such as Pascal or Fortran. In this form the knowledge gained may be used in other software.

HERB'S Evolutionary Algorithm



[Differences from real genetics:

genotype = phenotype;
male = female;
no recessives;
no chromosomes.

]

ABSTRACT

Network Support Overview: Services and Products

Ron Fountain, Hewlett Packard Company

This session will give an overview of the HP's Network Support Strategy and the products and services that are available that help users plan implement and operate their networks. Four services that were recently introduced are:

Network Planning and Design, Network Prepare, Network Startup, and NetAssure. Network Planning and Design provides users with a complete network strategy that supports their business objectives. An HP network consultant will analyze the user's communications requirements and create a detail network design based on these requirements.

Network Prepare helps users plan for a smooth integration of their new or expanded network into their business environment.

Network Startup provides users with the assistance they need to get the network up and running quickly.

NetAssure maximizes the uptime of the network by isolating problems on a multivendor network. HP will work with other vendors to ensure that the problem is resolved quickly and the network is returned to full operation. In addition to these services, the overview of the network support and diagnostic tools and packages will be given.

Is C Useful For Programming Business Applications?

Bruce Frank
Tim Chase
Corporate Computer Systems, Inc.
33 West Main Street
Holmdel, New Jersey 07733

In order to answer a question such as the one posed by the title of this paper, it's necessary that one understand the question. There is little doubt about what C is. It is a relatively new programming language which is gaining a great deal of attention lately as a general purpose programming tool.

The second part of the title question is a little more difficult to understand. What, exactly, is "business programming?" It is important that we understand what is meant by "business programming" if we ever hope to understand if C (or any other language for that mater) is good to use for business programming.

What is business programming?

Business programming is like a number of things that we really have a hard time of defining but "know when we see it." Everyone has a feel for what business programming is, but it's likely that no one can succinctly put the feeling into words.

One definition might be to characterize all programming done in a given set of languages as being business programming. But "If it's written in RPG, it must be a business program" somehow seems too simple to be really useful. After all, who blessed RPG or COBOL and said that they are good for business? What about those languages make them suitable for "business programming?"

One is tempted to assume HP's view of the world. HP divides the world's uers into "technical" and "commercial." Technical user's purchase HP/1000's or (more currently) UNIX based 9000's. Commercial user's get the HP/3000 with MPE. It would appear that HP clearly understands the differences between technical and business applications and knows what features are needed by each class of user.

Looking a little closer we may find out that this is just not the case at all. HP, at least in the recent past, has tended to be a collection of cottage industries. Each division is like an island unto itself (need we remember the number of incompatible BASICS that used to float

around HP?). It's quite easy to envision that in order to justify the existence of two computer divisions one would have to completely sell management on the concept of two vastly different types of users. In the HP world, much of the hoopla about technical users and business users is a convenient marketing scheme.

Convenient it may have been, but practical it is not. Take a look at DEC. DEC, long ago, realized that there is little difference in the hardware needed to do business or technical data processing. As a result, DEC has put its money in a range of processors which are all derved fom the same basic VAX architecture. HP has recently seen the error in its ways and has introduced the "Precision Architecture." Reminiscent of VAX in marketing concept, the SPECTRUM PA will provide the same basic hardware for both the technical and commercial side of the HP house.

So the hardware needs of the two classes of user are clearly not different. Perhaps the essence of the programming job is different for commercial and technical users. This, however, does not appear to be the case either. Sure there are some superficial differences. It's true, few shop floor data collection systems have to write payroll checks, but if you look below the surface, writing payroll checks employs the same programming techniques as any other data processing application technical or business.

In fact, one of the great realizations of the 80's is that both technical and commercial user's want and need the same tools to work with. Data base packages, file organizations, operating system features and so on and so on are the same now for business and technical data processing. Frankly, we would be hard pressed to tell the difference between a test system interfaced to a number of "cell controllers" (clearly technical) and a bank computer interfaced to a number of automated teller machines (clearly commercial).

In passing, it's interesting to note that HP's tendency to pigeonhole accounts as commercial or technical does a disservice to the end users. A banking account might have an easier time of it getting that automated teller project working on a 1000, but they would probably have a 3000 pitched at them because they are a "commercial" user. DEC, on the other hand, sells them a VAX, regardless, and then selects the interfaces to solve the problem.

About the only real programmatic difference between business and technical users is in the response to "almost." Technical users (perhaps due to their engineering roots) are often satisfied with "close enough." Floating point arithmetic calculating 2 plus 2 and coming up with 3.9999 is often fine with them. Commercial users, however, are dealing with people's money, or worse yet-- thir own. This means that adding 2 dol-

lars to 2 dollars had better result in 4 dollars; not a penny more or less. As a result, commercial users have a need for decimal arithmetic. Because of COBOL's standard, this appears to be limited to 18 digits. After all, even the US national debt can be represented nicely in 18 digits, thank you.

So, decimal string arithmetic aside, it would appear that programmers doing business programming or technical programming, use the same techniques, and have the same types of problems. Further, with "Computer Integrated Manufacturing" rearing its ugly head in the near future it is becoming even more difficult to draw the line between technical and commercial applications. Any one sticking to the notion that business programming techniques are fundamentally different from technical techniques will be left in the dust.

So, if it isn't the hardware or the techniques used in the software where is the difference between commercial and technical applications? There is only one place left to look -- the programmers themselves. Now, how can we delicately put this? Business programmers were often thought of as "less experienced" then their technical siblings. Much of the "wordiness" of COBOL is a result of trying to design a programming language to be written by inexperienced programmers and read by even less experienced management. Maybe, in the final analysis, this is the difference between business programming and technical programming -- the programmers. The techies built the computers to solve their own problems while the business side of the "Great Divide" had computers thrust upon them. Perhaps the schism between the two camps is actually a result of old skill mismatches.

Today, however, one need only look at the level of sophistication in the business application of computers to see that this is no longer true. The business people have made up for lost time and if anyone thinks that business doesn't have technical requirements then they are living back in the 60's.

So what about C?

In discussing C's suitability for business application development we will draw comparisons between C and COBOL. The reasoning is pretty simple. COBOL is considered by most people in the free world as being THE BUSINESS LANGUAGE. If C can stand up to COBOL, then it must have a chance of being useful in business.

Many of the arguments against using C for business applications are mute. This is because people are already using C for what once was thought of as being classic business programming. The question, then, has to be, why have a bold few chosen to ignore the obvious and picked C to develop business applications?

First of all, C is a highly portable language. So is COBOL. But C's portability stems from a different source than COBOL's. C is portable because it contains a small set of well defined features. All of C is contained in about 11 or 12 statements. There are a small number of well defined data types and a small set of rules for generating new data types. Contrast this with COBOL which attempts to get its portability by enumerating all the possibilities. COBOL's designers tried to look at all the ways of doing things on all computers and then designed options into the language to accommodate everything. The result is so poderous, that it becomes difficult in practise to use the full language because different implementations implement only parts.

The COBOL standard calls out various levels of conformance for each of 9 different modules. Nucleus is required, but the report writer is optional. This implementation has level 1 of the table handler while supporting level 2 of sequential file handling. It becomes difficult (if not impossible) to know if a given implementation will accept a given application; and all of this without considering what IBM has done for (to) the language.

C, with the introduction of its ANSI standard is simpler. To have a conforming compiler you have to have all the features. If an implementation does not have the features, then it's non-standard. Even without the standard, history has shown that it's eminently possible to implement complex applications in C and have them port from machine to machine with relative ease. So, reason number one to use C for business has to be C's portability. It matches that of COBOL and, in the opinion of many, exceeds it (this may especially be true for smaller machines where the resources for full COBOL are limited -- C is often available where COBOL is not).

Because of the high cost of software development and the standardization of operating systems (pronounced UNIX or MS-DOS), portability is becoming increasingly more important. Having portable applications frees users from vendor dependence -- a new and very comforting feeling for technical and business oriented consumers alike.

Apart from C's portability, many pick it for business applications because of its features and expressive style. Beusiness applications tend to have complex data definition requirements composed of records and fields and sub-fields and sub-sub-fields. One of COBOL's main charms is its ability to deal with hierarchies of data fields. It is not surprising to discover that C offers the same richness as COBOL, with some of the problems removed and some interesting new features added.

C's structure feature (much like PASCAL's records) is an analog of COBOL's data division but there are several basic differences:

C deals with templates while COBOL deals with storage.

If you define a structure in C, you are telling the compiler about a new data type, you are not actually allocating storage. In COBOL, you are always defining an area of storage. If you want two areas to look the same, you define two areas that look the same. In C, you just use the template twice. One of COBOL's contributions to Computer Science was the separation of data descriptions and algorithm descriptions. C takes this a step further and separates the "definition" of data from the "allocation" of data. This means that the C user can centralize the definition of data structures, yet decentralize their use. This is ideal for modular programming and quite difficult for COBOL even with the COPY statement.

C and COBOL both allow data fields and data movement.

The ANSI standard structure assignment found in C provides much of the power of COBOL's MOVE verb without some of the dangers. One of COBOL's complexities is the CORR option on the MOVE. CORR coupled with format changes implied by MOVES, have resulted in management rules disallowing MOVE corresponding in some programming shops. Because C allows structures to contain sub-structures, record formatting can be clearly and naturally represented. For example, consider the following:

```
typedef struct {
   int month;
   int day;
   int year;
} date;

typedef struct {
   char first[10];
   char middle_initial[1];
   char last[15];
} person;
```

This example shows the C source to define two structure templates. They are given names "date" and "person". These now become like new types. They can be used in single instances like:

```
date current_date;
```

Or in subscripted arrays or as components of more complex records. For example, the definition of a "time card" template might look like this:

```
typedef struct {
   person name;
   date   day;
   int    employee_number;
} time_card;
```

Once the time_card template is defined, it may be used to allocate time card's as needed. For example, an array of 100 time cards called "table" would be written as:

```
time_card table[100];
```

C provides much the same qualified access to sub-components of these structures as does COBOL. To access the name of the tenth time card in table one would write:

```
table[10].name
```

This value may be moved by assigning it to another data object defined to be the same type:

```
person employee;
    . . .
employee = table[10].name;
```

More complex moves can be performed by using C's library functions such as "strcpy" and "memcpy". There are number of these which work equally well with all data types.

C provides pointers and COBOL doesn't

One of the important features missing from COBOL which is present in C is the pointer. Because of pointers, C applications can often pass the address of objects rather than actually moving bytes. For example, a comon C practise is to assign a string to a pointer to characters:

```
error = "This is an error message";
```

This requires two memory references and as a result is very fast. In COBOL the same effect is achieved with:

```
MOVE "This is an error message" TO ERROR.
```

but requires a byte move of a literal constant to a variable and as a result is quite a bit slower. It would be difficult to say that either method is clearly more readable than the other.

In addition, pointers enable the C programmer to easily construct more

complex (and useful) memory-based data structures than those available to the COBOL programmer. The net result is that C applications are more compact and tend to be easer to understand.

But C has no facility for input and output.

Although C has no statements in its definition for I/O, every ANSI standard implementation must support a very extensive I/O library which is the same for all implementations. These functions provide a comprehensive set of procedures which provide for sophisticated I/O and formatting. C provides just about anything a programmer could want to access simple flat files. It is true, that there is a learning curve which must be delt with when using C I/O (there are conceptually no records), but once understood, C's definition of I/O is very flexible and quite powerful.

Applications which require more sophisticated file usage nowadays, invariable, use some form of standard data base package. C provides excellent features to interface into existing packages. This is often difficult for COBOL users and usually requires "special" COBOL interfaces. The best interface into the ORACLE data base package is achieved through C.

But what about decimal string arithmetic?

True. C does not support decimal string arithmetic. C does, however, support a macro pre-processor. This has been used by many to extend the language. At CCS, we have used the macro pre-pass to allow the HP/3000 users to gain direct access to the decimal string instructions provided by the 3000's extended instruction set. Because CCS/C macros can be programmed to provide in-line code rather than PCALs to separate routines, the implementation of decimal string arithmetic is very fast and efficient. Because C's macro pre-pass is a standard feature, it may be used to "abstract" the concept of decimal string arithmetic and provide an element of portability to even such a machine dependent feature.

The point of this is that C is quite versatile at adapting to the problem at hand. COBOL would be hard pressed to allow the programmer to gain access to some direct hardware features of the host computer. If a feature is missing from COBOL one needs to drop into another language. This is seldom the case with C.

Modular structured programming.

C programs tend to be highly modular. The features and organization of the language make this easy to do. COBOL applications tend to be huge monolithic structures which are difficult to understand and maintain.

C's features encourage the development of reusable code and the development of libraries of functions.

Will my programmers be able to learn C?

If in any given programming environment the basic difference between business and technical applications is programmer skill, then C may be in some trouble. C is not as easy to use as COBOL for small simple report type applications. If programmers are inexperienced, a 4GL or COBOL may be the right choice. But inexperience poses a greater problem in that COBOL or a 4GL in the hands of unskilled programmers cannot be "scaled up". The same features which make COBOL attractive for small applications, get in the way for large applications. Complex applications development will require skilled programmers and skilled programmers like C's expressive power and features.

C is clearly not for everyone, nor is it for all applications either technical or commercial. The decision to use C is one which must be made based on staffing and job complexity not on whether or not the job is a "business" or a "technical" job.

And in conclusion....

Business programming is just like any other type of programming. Business programmers need the same tools as technical programmers, they face the same problems and use the same solutions. In the 1980's there is very little to differentiate the two types of users.

C has been shown by many examples to be a successful programming language. It provides a terse, yet understandable style which is often a refreshing change from COBOL when used in business applications. There are not many features found in COBOL which do not have direct analogs in C. Few would argue that C is much more modular and expressive for large complex applications.

C's portability and availability make it an attractive choice for business software package developers who want to carve out as large a market place as they can across as many processors as possible. C has come of age and to not consider it for your next business application could be a serious mistake.

GETTING YOUR SITE SUPPORTED

Jackie Fritts
Westinghouse Electronic Assembly Plant
7807 East ByPass
College Station
Texas 77840

INTRODUCTION:

Hewlett Packard (HP) provides no real customer training in how to get the most customer support for the dollar. Many system managers are left to their own means to learn how to effectively interface with HP support. The purpose of this paper is to present what we have learned about getting support for our three HP 3000/70s and make concrete suggestions for all users.

AUDIENCE:

HP3000 System managers and computer site managers.

SCOPE:

HP3000 software support from Hewlett Packard.

ORGANIZATION:

Section 1. **What you need to know about HP support and how to get the most from it,**

Section 2. **How to Document a Problem Effectively,**

Section 3. **How to Use the Response Center,**

Section 4. **Problem Escalation,**

Section 5. **Using Other HP Support Resources,**

Section 6. **Summary.**

Section 1. **What you need to know about HP support and How to Get the Most From It**

The perfect site for any customer or vendor would contain the following elements:

(a) **Perfect software** developed at low cost to the vendor. This software never fails or behaves in an unexpected manner.

(b) **Perfect hardware** which never wears out or malfunctions.

(c) **Perfect documentation.** Never an omission or an error.

(d) **Perfect customers.** Must be able to interpret all situations correctly and resolve their own problems with applications.

Please let me know if your site is perfect. I want to meet you. In fact, you can be my boss. I'll work free just to find out how you do it!

As we have seen from the round table, such a site does not exist. We all have our problems. This is why HP has to spend many dollars maintaining its products and why customers have to expend resources making sure that applications are working properly. In this section we will study the standard software support HP sells and how we may be able to work together with HP (and each other) to obtain the most support for our money.

In order to obtain good support from HP you must know what services are provided and how to obtain them efficiently. You are in Las Vegas - do you know where your 5954-2767(D) is? How about your Exhibit 18T? These are the part number and title for the HP document which explains the Customer Support Services Agreement between yourself and HP. It makes excellent reading. Here are some things you will find there.

HP offers three types of support for your site: Account Management Support, Category Support and Family Support. Purchase of Account Management Support gets you 13 support services for the operating system only. If you want the same coverage for your software products you will have to purchase Category Support also. Family Support covers specified application software when the level of support differs from the usual 13 Account Management Support services.

The table 1.A below summarizes the 13 standard support services and classifies them into the support categories of product support, application support and information

support. Product support refers to the sale of a product to the customer by HP and the ongoing support of the normal operation of the product. Application support is the support of an HP application when the application has malfunctioned or is viewed by the customer as having malfunctioned. Information support is the distribution in printed form of information relating to HP products, services or applications.

Table 1.A

HP support services and categories

##	HP 3000 support service	product	appl	info
01	Account Support Representative	Y	Y	N
02	Support Management Review	Y	Y	N
03	Software Release Planning	Y	Y	N
04	Telephone Assistance	N	Y	N
05	HP Remote Support Services	N	Y	N
06	On site Assistance	N	Y	N
07	System Performance Analysis	N	Y	N
08	Software Release Installation	Y	Y	N
09	Software/Firmware Releases	Y	Y	N
10	Reference Manual Updates	N	N	Y
11	Software Status Bulletin	N	Y	Y
12	HP Communicator/Newsletter	N	N	Y
13	Software Problem Reporting	N	Y	N

The first support service is the assignment of your account support representative. We usually think in terms of our sales representative, software engineer (SE) and customer engineer (CE). It is up to your local office as to who will be your main support representative. HP tries to select the person they feel will best be able to serve you. Most of the time the SE is selected. Your account support representative is responsible for ongoing support of your account and ensuring that all the other 12 software support services are being provided.

The Support Management Reviews can be anything from yelling matches to sales pitches. It is up to you to be ready for the reviews by knowing the current state of any unresolved issues. HP likes to do annual account reviews. If you are having problems you should request quarterly reviews. HP may be reluctant but the support exhibit says at regular intervals. The regular interval is not defined.

Prior to the support management review you should make sure that the "right" people are attending. For example: If your SE has been unable to resolve a problem then invite the district SE manager or the Area SE manager to the meeting.

Make it extremely easy for these people to attend even if the meeting has to be rescheduled. Send them a written invitation to start with. If they refuse then call them directly and ask them again. If you cannot reach them by telephone then make an appointment and see them in person. Getting the right people there allows you to get acquainted and can be of great assistance in getting your SE the priority needed for your problem. Getting increased priority for a problem is called problem escalation. We will say more about escalation in section 4. Don't be shy. If you have a problem HP wants to fix it and as much (if not more) than you do. They will be glad that you kept after the problem. They are as busy and cost conscious as we are and sometimes need a little push. It is very important not to cry wolf. The first time you report a trivial problem as serious and bring in the "right" people will be the last time they come. This can be very harmful to both you and your SE. Be sure you are right before you go ahead.

A second thing to do before the account management review is to set the agenda. HP will not be offended if you have an efficient meeting agenda ready when the meeting starts. This agenda should be written and distributed when the meeting starts. You could even send HP a copy ahead of time so they can prepare. It is not very professional to "spring" things on your SE in the presence of his bosses. More will be accomplished if both sides know ahead of time what is coming. Also, prior notification to HP sets the meeting subject and depth. Not setting an agenda usually results in a "default" management review meeting usually run by the sales representative. HP will not know what is bothering you unless you tell them (and tell them ...).

Support service 3, Software Release Planning, is assistance by HP in planning the installation of major software releases. This is usually a new version of MPE. Installations of the software releases are covered by support services 8 and 9. These services provide input from your SE before the installation is done. Software Release Installation Assistance (support service 8) is available at your request. If the installation is to be done during standard hours then you have the option of having your SE on site during the installation. This is a good idea. If something goes wrong the SE will be able to contact other HP resources which can usually resolve the problem quickly. Also, if you have a large patch set the SE may want to customize a set of application patch tapes for the new release. He would then be able to assist you should the custom tape fail.

If you cannot do the installation during standard hours then the SE will not come to the site unless you pay for a service call (Usually travel plus $175/hour. See Exhibit 18T Section C paragraph 4). Some SEs are willing to give you a

phone number to call if you have a question or problem. Most of them want to know right away if the installation did not work. They are not obligated to do this but it makes everyone look better if everything works on Monday morning.

As with the Support Management Review there are some things you can do to help software upgrades go more smoothly. During the software release planning sessions with the SE you could ask the following questions:

(1) Has this upgrade been installed at any other sites similar to mine? Do I really want to be the first to try it out?

(2) Why is this upgrade necessary? If this upgrade does not address any of your my problems, then why install it?

(3) What additional system resources will this upgrade require of my CPU? Does it use more memory? Does it use more disc space?

(4) What impact on system performance will this upgrade have? How do you know?

(5) Will any of my existing application software have to be changed or recompiled in order to support the new upgrade?

(6) Which patches specific to my site have been incorporated into the new update? What patches were not incorporated? Will I need to install the patches? If so what procedure should I follow?

(7) Do I need a site specific patch installation tape in addition to the usual tapes?

In addition to these questions you could request that the SE provide you with:

(1) Written instructions for the installation and a review of the procedures in the instructions.

(2) The output from a successful run of all procedures in the instructions.

(3) The time to ask questions about the instructions before using them.

(4) Verified copies of the media needed for the installation. Have the SE verify the tapes. You do not want to have a bad tape stop a lengthy installation procedure.

This may sound like a lot to expect from HP but they will be glad to provide it. They want your installation to go as smoothly as possible. Also, most of these things have to be

done by the SE before he talks to you. If he knows your requirements ahead of time, he will make sure that he knows the answers to all your questions and has the written materials you need. This approach avoids a "hacker" approach to the upgrade which can waste time and yield poor results.

If you are updating the operating system you should also read the copy of Communicator 3000 which corresponds to the version of MPE you are installing. There are sections explaining changes/enhancements to MPE and HP applications you should know about ahead of time. This will allow questions 2/5 above to be product specific.

Support for your HP applications starts with the Telephone Assistance support service from the response center. If you have a problem your system manager (or designated alternate) can report the problem to the response center. A clerk will enter the call into the queue for the product and a specialist for the product will call within two hours. The sections on problem documentation and using the response center are intended to minimize the time required to get the information you need. We could think of the response center as a system. If we provide it with good input then we get high efficiency and valid output. The response center is discussed in some detail in section 3.

The performance analysis guaranteed by support service 7 has never been done at our site. The CPUs have always been so busy that we felt that the analysis would slow response time down to the point that production would be affected. We have used OPT and other means to obtain some measure of system load and performance measurement.

Support services 8 and 9, Software Release Installation Assistance and software/firmware releases guarantee you the services of your SE at your request to install upgrades of major software releases. How to get the most of this service is discussed above with support service 3.

Reference manual updates do you no good if they don't get incorporated into your manuals. This means they have to get to you and have to be inserted in the right place. Your manual should exactly explain the current state of your software. Late updates can cause new features to be unnoticed and early updates can cause programming errors. You can find out about new features by reading the Communicator 3000 and Software Status Bulletin. Then check your manuals for version and see if all the new features are included. Call your SE if you find any missing features. He can usually find out if you need any manual updates and get them to you.

We have already mentioned the Software Status Bulletin (support service 10) and HP Communicator/Newsletter. One

copy of the Software Status Bulletin (SSB) is mailed quarterly to the system manager. Updates to this copy are sent periodically. The SSB contains descriptions of all known problems in the software product lines listed in the Cumulative Product Index. The entries in the SSB are also indexed by Known Problem Report (KPR) number and Cumulative Keyword Index. The SSB is a large document. The 15 April 1987 contained entries for 2510 KPR numbers. If you know the KPR number for your problem then you can find the status of the problem quickly. If you do not know the KPR number then you have to count on the keyword index to find the status of the problem. For example KPR # 470152504 deals with a SF 310 (System failure 310) that was fixed at our site. Knowing this we can look up all the latest information on the problem. For example, the problem has been fixed in U-MIT. If we did not know the KPR number we would have to find the problem by its keyword. In this case the keyword SF310. There are only a few entries. In some cases there are many entries for a single keyword.

An alternative to SSB is Electronic System Status Bulletin (ESSB). This is a set of editor files which you can restore on your system and search electronically. We had ESSB for beta testing on our site and it was much easier to use than SSB.

We use SSB (ESSB) to get a problem report started. It is very nice if you can tell the response center that you are reporting a known problem and give them the KPR number. This helps to focus the problem and give the response center a better starting point for the problem investigation. If you cannot find the problem in the SSB then make sure to ask the response center for the KPR number if they tell you that your problem is known. If they do not know the KPR number then ask that they tell you what it is before you close the call. This way you can track the problem. You should also ask your SE for KPR numbers of problems you are having. If no KPR number has been assigned then your problem may loose all visibility and not be addressed for some time.

The Communicator 3000 is published at each release of the MPE operating system and provides detailed information on significant enhancements made to HP 3000 system software, hardware, and documentation. It is very wise to read the sections in Communicator 3000 which apply to products on your site before you upgrade the operating system or product.

Support service 13, Software Problem Reporting gives you an alternative to the response center. You can report a problem with installed HP software or updates by submitting an Software Service Request (SSR). Usually the SSRs are written by your SE on your behalf but you can do it yourself. HP will acknowledge receipt and inform you of the status of the

SSR. HP reserves the right to determine the final disposition of all SSRs.

There is no real advantage to submitting your own SSR except that you can make sure that all the information on it is correct and you know exactly when it was submitted. Also, if you submit your own SSR you usually get a copy of the receipt acknowledgment. This is not always the case when your SE submits the SSR for you.

You can overcome these three problems by having your SE send you a duplicate of all SSRs submitted on your behalf and of all SSR receipt acknowledgments received. He can usually put you on the distribution list for the SSR.

To qualify for support services there are some things you are required to do. These are:

Q1. You must agree to support the operating system and all applications by whatever software support is available,

Q2. You must supply a System Manager and a designated alternate trained through completion of the appropriate HP training courses or equivalent experience,

Q3. You must purchase your software products from HP and have an appropriate HP software license,

Q4. You must purchase support for the system with the fastest processor speed and largest main memory on which program development work is being done,

Q5. If you want HPTREND you must supply a modem link capable of sending the data to HP,

Q6. All your software must be at the current release or revision level prior to current release.

Once you have qualified for support there are some limitations to the software support services. These are:

L1. HP will not support modified or third party software,

L2. HP will not support those non HP products used with HP Series 100 and non HP personal computers that are not listed by HP as part of an HP supported configuration,

L3. HP will support the current supported version and the immediately preceding version of HP software products.

L4. HP will charge you time and materials for investigation and repair time for the resolution of problems resulting from the use to privileged mode code on the HP3000.

L5. HP support is limited to problems which can be duplicated on the standard version of the object code of the particular software product. Any other assistance will be billed on a time and material basis.

L6. Data recovery services are not included in the support services regardless of the cause of data loss. Data recovery assistance will be billed on a time and materials basis.

Other support limitations pertaining to hardware/firmware have been omitted from this list.

You also have some responsibilities. These are:

R1. You have to provide HP with access to and use of all customer information and facilities determined necessary by HP to provide Account Management Support,

R2. You are responsible for reconstruction of lost or altered files, data or programs,

R3. You must follow routine operator procedures as specified in the HP operating manuals for your products,

R4. You must provide a representative who will be present at the site at all times HP is performing service (on site or by phone),

R5. You are responsible for the safeguarding of your proprietary, confidential, and classified information,

R6. You must allow HP to maintain system diagnostic programs on your system,

R7. HP can require you to return your software on its original media to be eligible to receive software updates,

R8. You are responsible for the execution of all HPTREND software and transmission of HPTREND data prior to the creation of an HPTREND report.

These qualifications, limitations and responsibilies mean that you must guard your data, don't fool around with priv mode, make sure you watch what HP does to your stuff and don't tell HP anything they shouldn't know.

Section 2. **How to Document a Problem Effectively**

Problem documentation is effective if it helps get the problem resolved. This section explains the following elements of effective documentation:

E1. Problem Identification

E2. Problem Description

E3. Problem Reproduction

E4. Problem Tracking

The first element of effective documentation is a complete investigation the problem. The investigation should be done before calling the response center. There are several phases in the problem investigation. First the system manager should talk directly with the user who experienced the problem. Some questions to ask the user are:

UQ1. How were you logged on?

UQ2. What statement or command were you using when the problem occurred?

UQ3. What did you expect to happen when you executed the statement or command?

UQ4. What actually happened? Be Specific.

UQ5. Has anyone else had the same problem? (ask several users)

UQ6. Has this happened before? When? What did you do?

UQ7. Has the statement or command ever worked before? When?

UQ8. Did you do anything this time that you do not usually do?

UQ9. Can you make the problem happen again? Show me.

UQ10. Did you notice anything else unusual?

Write down exactly what the user says. This information can be used later in problem tracking.

Armed with the answers to these questions you should isolate the environment in which the problem occurred. This includes the version of the operating system, a list of all HP software (compilers, interpreters, editors, database managers, etc.), and all other application code (third party

and in-house) involved. Also include a list of any peripherals involved (printers, terminals and whatever). For each peripheral include its model and serial number.

A good description of the problem is one of the keys to ensuring that your response center call will be properly routed. Include the HP application in use, what the user was trying to do, and the results. Be sure to include specific error messages. Be as concise as you can.

Now try logging on the same as the user and repeating the steps they gave you for reproduction of the problem. This is necessary verify which application was causing the problem and will provide HP with a means to recreate the problem. It will simplify matters if you write a short program specifically intended to recreate the problem. HP may also be able to use your program in their investigation of the problem.

Problem tracking is the process of keeping up with the current status of a given problem and detecting trends in problems over a period of time. This is both tedious and time consuming. We have used two methods for problem tracking. The first is a HPLISTKEEPER list which allows quick entry of outstanding problems. The second is a database which allows us to print more involved reports.

An entry in the HPLISTKEEPER system contains the fields:

HPLISTKEEPER FIELD NAME	FIELD CONTENT DESCRIPTION	ENTERED BY
Date Entered	When the problem was put into HPLISTKEEPER	reporter
Entered By	Who put the problem into HPLIST	reporter
Identified By	Who found or experienced the problem	reporter
product name	Official HP name of the product	sys man
description	Description of problem	sys man
date RC was called.	Date call placed to the response center call was placed.	sys man
time RC was placed.	Time the response center call was placed.	sys man
PICS ID	The identifier assigned by the response center	sys man
RC analyst	Specialist who will solve the problem	sys man
RC solution	The workaround or solution to the problem.	sys man
Comment	Open comment field. Good for KPR results of fix or call cross ref	sys man
Problem Status	A flag for the current state of the problem. We use open and closed.	sys man

This LISTKEEPER system has worked fairly well but does not allow enough space for complicated problem descriptions. We are implementing a simple database for problem tracking to overcome this problem.

The structure of the database is too complicated to discuss here. The purpose of either of these two tracking systems to find out about the current status of problems. Also, we generate some reports which can be used in the reviews with HP. A key ingredient of the tracking system is the CALL ID assigned by the response center. We will say more about how to use this in the next section.

Section 3. **How to Use the Response Center**

The response center is HPs main software support effort. It is possible to obtain excellent service from the response center provided one knows what to expect. The purpose of this section is to explain what the response center does, what is expected of you and how to overcome some of the problems most users experience.

The response center provides the following services:

RCS1. The response center can answer questions pertaining to the normal functioning of your software.

RCS2. The response center can assist with discrepancies in documentation or software functioning that make it difficult for you to use the software package effectively.

RCS3. The response center can help you determine which HP engineer can best resolve the problem.

RCS4. The response center can help you restart your system after a system interruption. You should report the interruption even if you do not wish assistance. The response center tracks such failures.

RCS5. The response center can advise you on the availability of a patch should you find that a patch is available from the SSB. They will also assist if you have trouble installing the patch.

To initiate these services you have to place a response center call. You should be very prepared before the call is placed. The information to have at hand is your system handle, your system and model number, your problem description and the severity of the problem.

Problem severity levels are:
4 - general usage;
3 - production affected;
2 - system/application failure but recovered, and
1 - system down.

Be sure to tell the call coordinator what the problem severity of your problem is. If they don't ask and you don't tell then they will assign the severity themselves.

When the call has been placed the call coordinator will route the call to the appropriate team at the response center. The teams of the response center are: personal computers, office automation, system interrupt logging, manufacturing, finance, distribution, data communications, design and manufacturing, instrumentation and test. A member

of the team will call back within two hours. Team members are SEs. When they call you back the problem will be resolved immediately or will be submitted for further investigation.

Immediate solutions occur when the SE already knows the solution from experience, can find the answer while you are on the telephone or the problem is being investigated. In the last case the SE will let you know the current status of the problem.

It's not always this easy. Some problems are based on complex usage misunderstandings or nonreported software discrepancies. The SE usually has to perform several tests to tell more about the problem before he can tell where to start.

If the problem is judged a complex usage problem then the SE will decide whether what you are trying to do with HPs software is feasible. You may be asked to reduce the size of the problem or to provide information as to how to reproduce the problem.

If the problem is a software discrepancy then the response center SE will determine if the problem has been reported. If so it is a reported problem and a KPR number will have been assigned. A workaround or a patch may also be available. The SE will let you know.

If the problem has not been reported things are more complicated. The SE will want more information. You may be asked to reduce the size of the problem and/or reproduce the problem. The response center SE will try to devise a workaround for you. If he is not successful then the response center team will submit an SR for the problem. If you have extensive documentation then they may ask you to submit the SR.

If the problem involves a software problem that requires local assistance then the response center will refer the problem to your local office. It is then up to your SE to interface with you and the response center in the resolution of the problem.

When the problem has been solved, or an SR has been filed of the ownership of the problem has been transferred to your local SE then response center would like to close the call.

The response center sends reports to your local SE on calls that have been open for an extended period.

If you have a recurring problem you can reopen the call by referencing the old call id number. You will be issued a new call number and the process will start all over again.

Now all this sounds very good but in actual practice there are some problems.

One of most difficult problem for the response center is keeping track of multiple instances of the same problem. Consider the following scenario:

Three users (say A, B, and C) report a problem with the security feature of VPLUS 3000. Say that the echo for a secure field is not being turned off. User A using a correct COBOL program, user B is using a correct TRANSACT program and user C is using a correct PASCAL program. The problem is with VPLUS. They all call the response center. User A reports a COBOL problem, user B reports a TRANSACT problem, and user C reports a PASCAL problem. The call coordinator will route the three callers to the appropriate teams. All three users will get different call ids for their problems. Each of the users is not getting the proper terminal id from VPLUS and none of them will know that the other exists. Now if this is a new problem there is a good chance that the response center SE will think it is a problem with the language and will send in an SR to this effect. The lab will assign a KPR to the problem on behalf of each of the users. Now suppose the severity is 4 for user A, and 1 for users B and C. The lab will eventually discover that the problem is with VPLUS and not any specific language.

When the lab discovers this they will assign one of the KPR numbers to the problem with VPLUS. Suppose the KPR the lab chooses belongs to user A. User A now finds from the SSB that his problem is being worked on and sees that there were two other users with the same problem. The severity of the problem is 4. What happens to users B and C and their severe problem - you ask? Their KPR vanishes from the face of the earth. It is no longer in the SSB and the response center SE has no way track it since it was switched around in the lab from the language under which it was submitted to VPLUS.

How can we prevent this? That is a very good question. I am not sure that the current system can prevent it. One suggestion would be that the response center send a copy of any KPR they submit to the lab to all the customers who had any input into the generation of the KPR. This can be done by generating a cross reference between the call ids and the resulting KPR numbers. This would let everyone know that the KPR had been submitted. Next, the lab could collect the mailing lists for all the KPRs involved in the problem definition. All the users could then be notified of the new KPR number. After that the users can track the problem themselves and the response center would be able to tell the user when a KPR had been replaced.

Until HP corrects this situation you can ask for copies of all KPRs for known problems and copies of any SR's submitted

on your behalf by the response center or your local SE. By getting these in writing you can sometimes bridge the KPR gap from the problem description. Also, written copies of the SR's and KPR's help you make sure the KPR was really opened and the information is accurate. It is amazing what some of the information looks like after passing through the call coordinator and several SEs. Also, you can leave the response center calls open until the problem is resolved at your site. This will cause the response center to track it longer. There may be less chance of the KPR vanishing if the response center SE is trying to find its latest status.

A second problem with the response center is the "can't duplicate the problem dead end". Recall from section 1 that HP is not required to fix anything they cannot duplicate. If it will not happen in the lab then they don't have to try to fix it. Some problems cannot be reproduced because the SE is trying to use incorrect or incomplete information from the call report or SR or because the problem is site specific. On many occasions my calls have stopped at this point. The call was closed, without my permission. No SR was generated. The call was not forwared to my local SE.

The reason for this was that the response center SE stays at the response center for a week at a time. When the problem could not be duplicated at the lab, the lab never responded to the response center's inquiry about the problem. The replacement response center SE was never told to call back since no call came back from the lab.

Since HP is perfectly within their contractual rights to drop problems they cannot reproduce and it it up to HP how hard they try to reproduce the problems, you have to speak up for yourself.

The way to handle the situation is to never close a response center call unless you have made sure that your local SE knows about the problem. It is up to you to call the response center and check the status of all your open calls once a week. This is especially true if you never get an SR or KPR or your KPR number vanishes from the SSB. If the replacement response center SE does not know about your call then ask that they find out. If the call was closed for any reason the reopen it. Be sure to cross reference the call ids of the two calls and any KPRs you can find from the old calls.

Another thing to watch for is the "workaround". Sometimes you will be told that a workaround has been found. These sometimes sound good at first. However when you investigate the workaround you discover that it does not work at your site. Some other workarounds are ridiculous. You find such things as "don't use the feature until ZZ-MIT".

If you do not make it perfectly clear that the workaround is not acceptable you may find your call closed. Remember that HP is only obligated to provide a workaround. If you accept it then the obligation of the response center is ended.

Never accept a workaround that affects your operations adversely. Tell the response center that you will evaluate the workaround for your site and call them back. Explicitly tell them to leave the call open until you call back. Then check the workaround carefully.

Another reason to not accept poor workarounds is that they tend to become fixes. If they were not good as workarounds then they will not be good as fixes.

The vanishing KPR, the workaround, and the can't reproduce problems are the most serious drawbacks of the current response center system. There are some other minor annoyances. One of them is that the response center is sometimes used as a training (or proving) grounds for new SEs. It is tough to have to tell the response center SE that TRANSACT is a programming language when he is supposed to be fixing your problem. If you get a response center SE who does not know what is going on, don't try to train them yourself. Ask for another SE.

One last swat at the response center is that currently they are only available during HP local site hours. Stuff also breaks at night and on weekends. I hear that the hours will be expanded soon.

If you do not feel that your problems are being dealt with properly you should call response center and ask for the customer service desk. They will help you. If that doesn't work then ask to speak to the director of the response center. Remember in all this that HP wants to fix the problem and will welcome your help in getting the situation cleared up to your satisfaction.

What happens when the response center cannot resolve a problem is covered in the next section on problem escalation.

Section 4. **Problem Escalation**

Problem escalation is the process of getting more resources assigned to a problem. HP has certain escalation rules built into the software support system. The purpose of this section is to point out the built in escalation strategies and suggest methods of obtaining escalation when these may not exactly apply.

The normal start of a problem report is the response center. The call coordinator there routes your call the appropriate software engineer who calls you back. You should get your call back within two hours.

If after a reasonable period of time the response center SE cannot resolve the problem, he will hand the problem off to your local SE who will contact you. There is no set time for this part of the resolution but the response center should tell you when the hand off has occurred and you can contact your local SE.

If your local SE is not able to resolve the problem by dialing into your system he will come on site and attempt to fix the problem. If he is not successful then your site will be escalated to a problem site. This is the first step of the normal escalation process. Escalation to problem and hot site statii are the normal escalation levels built into the HP escalation procedures. You are a regular site until HP escalates you.

You will be escalated to a problem site if a problem is not resolved within 4 hours of your SE arriving on site. This escalation must occur after 4 hours even though the problem is felt to be diagnosed at that time.

For recurring problems the rule for escalation to problem site is based on the 5-4-3-2-1 rule. This says that your site will become a problem site if you have any one of the following:

(a) downtime greater than 5% in one month,

(b) four unresolved failures, halts, hangs,

(c) during the three month warranty period a system site has not stabilized,

(d) a recurring intermittent problem is unresolved after 2 weeks elapsed time, or

(e) You are a guaranteed up time service (GUS) customer with downtime approaching 1% per month.

The third way you can become a problem site is when common sense dictates that a problem is critical.

Of these three the only one you can control is the third. It is up to you to make it clear to HP that a problem is critical. This should be done with information such as the cost of downtime to your corporation, the critical nature of your business, etc. For example, down time on our shop floor processor rises to $3000.00 each minute after 15 minutes, gradually decreases to a fixed value dependent on paper data tracking manual material movement then stays there until the problem is fixed. You should analyze the cost to you and make sure it is known to HP and to your management.

Make sure to keep track of downtime, failures, halts, hangs and intermittent problems. You could log them in a database (OCS provides one) or just write them down in a logbook. If you request escalation based on cost or whatever and HP refuses (and they can) then you have the data to know when you have met one of the other criteria. HP cannot refuse the escalation when you have the data to indicate that the escalation is required.

By the way, 5% of a month is calculated as: 24 hrs/day * 30 days/month * 5 percent / 100 = 36 hours! That's a long time. We would have to close the plant waiting for this to happen. If you are a GUS customer then be aware that 1% is 7.2 hours/month. Still a long time!

Should you feel that your problem should be escalated for any reason the persons to contact are your account SE, your senior SE, your sales representative, your district manager and area manager. You should work your way up if refused. Better yet you might have someone in higher management contact someone in HP higher management. This sometimes works wonders.

Once you have reached problem site your SE will report the problem status of your site to the district manager and participate with the district manager in the development of an action plan to resolve the problem. Your SE will continue to function as your first level technical resource and owner of the technical solution of the problem. Your sales representative will communicate the problem status to customer management.

The district manager will function as problem site manager for your site. He will mobilize field technical, logistic and operations resources, inform you that your site is being escalated, and conducts meetings with the field marketing team members to review the status of the escalated site. The frequency of the meeting should be specified in the action plan.

In addition communications channels within HP will be opened between your local SE and the district sales manager, district customer support manager, district software support manager and system specialist. The district manager also makes sure that communications are open to you, the major account team, the service dispatcher/response center, the field marketing team and the area management team. You can how much the problem is escalated by the length of the distribution list on the memoranda you receive. The longer the better for you.

The conditions under which you are escalated to a hot site are that the problem was not resolved within 8 hours of the arrival of your SE on site. Escalation must occur even though the problem may have been diagnosed at that time.

For recurring problems you become a hot site if you have had eight unresolved system or product failures, hangs or halts in any 60 day time period. You also become a hot site if you experience four identical system or product failures, halts or hangs in any 30 day period.

You will be escalated from a problem site to a hot site if your action plan deadline has been reached or the field technical resources are insufficient to resolve the technical problem or your anxiety level is escalating faster than the problem resolution. Again, it is up to you to make your anxiety level politely but firmly known to the right people.

When you reach hot site status the HP district manager will notify the appropriate factory SE manager of your escalation to hot site status, modify the action plan to include input from the factory, present the action plan to the customer for approval, conduct meetings with the field marketing support team members, ensure solid lines of communications have been established between field and factory technical, logistics, and operations resources, and integrate factory level action plan into the overall problem resolution action plan.

The factory SE manager becomes a resource to the local SE. The factory SE manager is responsible for coordinating all factory/product division resources to facilitate the resolution of the problem.

In addition to the factory SE manager a factory problem manager will review your problem situation and action plan with the site problem manager and appropriate technical resources. The factory problem manager will recommend changes to action plan as appropriate.

The factory problem manager will also consult with the site problem manager and field technical resources to facilitate the diagnosis of the technical problem.

The factory problem manager will develop a factory level action plan. This is the plan that the district manager will have to integrate into the existing action plan. This factory level action plan will take no longer than 48 clock hours for development and distribution. The action plan must define the dedicated factory resources (personnel, equipment, and material) and timetable to be used to solve the technical problem. This factory level action plan will be distributed to the site problem manager and the appropriate individuals within the factory.

The problem site manager will maintain communications with the customer, major account team, service dispatcher/response center, field marketing team, area management team, region management team, factory problem manager, computer support division manager, and field marketing team.

Your account district manager will maintain communications with customer upper management.

Your factory problem manager will maintain communications with the site problem manager, technical support manager, system specialist and appropriate factory management.

From all this information it is clear that HP has resources necessary to resolve your problem once the problem has been escalated. If a problem is not solved to your satisfaction, do not accept the solution. You do not have to agree to live with a workaround if it does not work for you. This is important for yourself and for others. The users set the standards for software support. If we accept less than what we need then everyone will suffer. Don't be afraid to hold out for a solution acceptable to you. Everyone will benefit in the long run.

If you have statistics which require HP to escalate then HP cannot refuse to escalate your problem. If the escalation is at your request then it becomes a point of negotiation with HP. Get all your information together, decide what you want then go after it. Try not to be unreasonable but never give up.

A few last suggestions. Ask for a copy in writing of all action plans. HP usually does them verbally when they decide to tell you about them at all. Ask for details and deadlines. Some of the action items are also negotiable. Also, you should know your system better than HP and some of the action items may not be feasible.

Find out who the factory manager and the factory SEs are who are assigned to your escalated problem. Call them if you run into the workaround or can't duplicate dead ends. If nothing else you can tell your local SE or the response center to call the factory representatives with additional information or questions. This way you can help open communications channels and keep the problem moving.

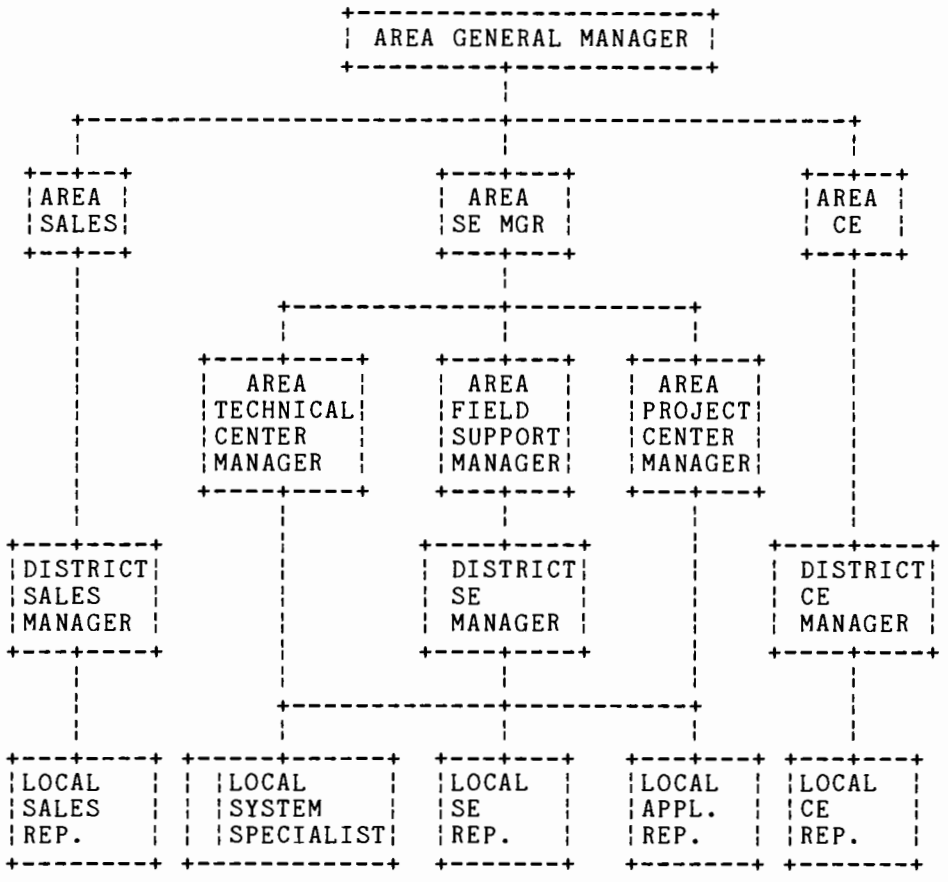
Section 5. **Using Other HP Support Resources**

This section is intended to summarize specific support suggestions from the previous and add a few additional suggestions.

One thing we have not mentioned is who to call for service. This depends on the problem or service needed. According to HP you should call the response center for software usage assistance and problem resolution., system interrupt assistance, and questionable hardware/software problems. According to HP you should call your local SE for training, consulting, project management, implementation assistance and information about support contracts. According to HP you should call your sales representative for new product information and new orders or new order tracking.

This is fine if you can find who you need and HP takes some action based on your call. If not you need to work your way up the HP support structure until something happens. In order to work your way up the structure you have to know the structure.

At the local level you have an account team. The structure of a typical team is based on area, field, district and local members. A diagram is shown below.



You should know who all these people are and how to contact them. Your local SE can give a list of their names and phone numbers.

A resource we have not mentioned is the Software Release Bulletin (SRB). This is published prior to a new release of the operating system. The SRB documents all fixes and enhancements that are incorporated into a new release of the operating system. If you have been lucky enough to trace your KPR through the entire process then you can verify that the KPR is really resolved in the release. It is a very good idea to check. When you are told by the response center that your KPR will be fixed in a future release then ask which release and what the KPR number will be. Then check the SRB for the fix. It it is not there call the response center with the original call id and the KPR number. Either the KPR has changed or the fix was not ready when the release

deadline was reached. You should be able to find out what happened and correct it. Also contained in the SRB is the release of the software product which contains the fix.

Another publication of use is the Communicator 3000. This is also published just prior to a new release of the operating system. It contains descriptions of all the new enhancements to the products under the new release. Also included is a MPE PRODUCT RELEASE TABLE. This lists the product and under which version of the operating system it was introduced.

There is also a catalog of user documentation in the Communicator 3000 which lists the latest version of all the manuals, when they came out and what subscription service gets them. The manuals are grouped into data communications manuals, system manuals, subsystem manuals, language manuals, software product manuals, transaction processing manuals, maintenance management manuals, materials management manuals, production management manuals, production cost management manuals, generic manuals and JIT manuals.

Another resource available is the HP3000 Application Notes. These are published biweekly by the response center and contain information the response center would like you to have. For example Application Note #25 is about TurboImage Transaction Logging. It is a summary of the sections in the manual on logging and answers some of the most common question encountered at the response center.

The response center also publishes a list of the most common questions asked them. These are helpful in avoiding unnecessary calls to the response center. If you find the answer in the questions and answers a call can be avoided.

Section 6. **Summary**

The details of support are a little overwhelming (for me anyway). This section summarizes the previous sections and is intended to be a guide should you have problems.

Section 1:

There are 13 support services available under AMS support.

Prepare for any type of support meeting by knowing problems, preparing a written agenda, and inviting the right people.

Prepare for any type of upgrade by defining what you expect ahead of time, asking questions and checking all references available.

Make sure you meet all your responsibilies and know the limitations of your support contract

Section 2:

Documenting a problem consists of identification, description, reproduction and tracking.

Log all communications between yourself and HP

Section 3:

Make sure all your descriptions are concise and accurate

The response center has some problems: Workaround, vanishing KPRs, and can't duplicate.

Never close a RC call until you are satisfied with it

Section 4:

Never attempt to escalate a problem unless you absolutely have to.

Keep statistics needed to prove the escalation is necessary

Get all the information you can during the escalation

Know the status of your KPR all the time

If possible contact the people working on the problem directly.

Section 5.

Know who to call.

HP supplies SRB, Communicator 3000, and response center publications.

References Used in This Paper:

Response Centers Users Guide, 1986, Hewlett Packard Company, part number 5958-7386 E0886

HPTREND: An Installation and Problem Solving Guide, 1987, HP 3000 Application Note #27, North American Response Centers, Document P/N 5958-5824R2718

Exhibit 18T Account Management Support, 1986, Hewlett Packard Company, part number 5954-2767(D)

HP 3000 Off-Hours Emergency Assistance (System Engineering), 1982, Hewlett Packard Company, part number 5953-5204

Communicator 3000 Version G.02.01 of MPE V/E (UB-Delta-1), 1986, Hewlett Packard Company, part number 5958-3153 R 2642

Software Status Bulletin, 1987, Hewlett Packard Company, part number 32002-90007 E 2716

Software Release Bulletin Version G.02.01 of MPE V/E (UB-Delta-1), 1987, Hewlett Packard Company, part number 32002-90006 R 2642

System Security: A Hacker's Perspective

Jerry Felix
Chris Hauck

Hewlett-Packard Company
Cincinnati Sales Office
4501 Erskine Road
Cincinnati, OH 45242

Disclaimer

This paper documents activities and personality traits which neither the authors nor Hewlett-Packard condone. They are described here solely for the purpose of discussing appropriate techniques for their prevention. This paper does not attempt to present an exhaustive list of vulnerabilities or prevention techniques. Rather, it highlights the most common methods of breaching security.

Introduction

Hi. You don't know me, but I know where you live, how much you make, and what you paid in taxes last year. I could bring your company to its knees, force you to lose your job or order merchandise from your company without ever receiving a bill. I am your worst nightmare - I am a HACKER.

Who am I, you ask? Am I a disgruntled employee eager to "get even" with the company that I feel has mistreated me, by misusing some of the information available to me? You shouldn't have trusted me as much as you did. I have access to all the information kept on the system because I do the nightly backups. I can use this knowledge for personal gain (modifying payroll records) or to achieve a feeling of "power" over the organization (crashing system or corrupting data).

Or, am I just an innocent data entry clerk? How was I supposed to know that I should not access the payroll information, when the payroll department has the password taped to their terminals?

Or, am I your largest competitor, searching for information on your clients and

production rates? With your on-line, dial-up ordering system, you were asking for trouble. You published the procedure for accessing your system, and I took advantage of the opportunity by circumventing your feeble attempts to secure your system.

Or, am I someone without any direct interest in your information? I don't care what happens to your data or your company; I am just a teenager with a PC and modem at home, eager to prove my computer prowess to my peers. I'm not looking for anything in particular, but I just enjoy the challenge of trying to break a system's security. I can't resist the feeling of power achieved by knowing that I have some control over your multi-million dollar company.

No, you don't know who I am, or when I may penetrate your system security, but by understanding my motivation and techniques, you may be able to stop me. But then again, maybe not!

Tactics

My efforts can be categorized into four main steps:

- Gain access to your system.
- Achieve necessary capabilities.
- Acquire and/or change the information.
- Leave a "backdoor", an alternate method of access, for next time.

Gaining access to your system may not be a difficult task. If I am an employee, you have probably already given me a password (if not, there is probably one written near a terminal somewhere). Sometimes I don't even need to sign on, since someone has already done it for me, and left their desk. Of course, there are always many generic user names and passwords that I will try. TELESUP is a great target, since by default, every system has the same passwords, and new system managers are intimidated about changing them, since it is used by HP. And there is always the other "standbys" - HPOFFICE, SUPPORT, ITF3000, HPPLxx, etc., that end up on systems but may not even be used. As a last resort, I can use any of your backup tapes to find out any passwords (the entire system directory, including passwords, are located at the beginning of EVERY backup set). Another method, slightly more obscure, is to =SHUTDOWN the system, then examine the contents of the system directory using either HP provided utilities (DUS tape) or low-level debuggers built into the system startup routines.

If I am not an employee, your system may be connected to a public data network, such as TELENET or TYMNET. But more likely, your system is connected to the

"most public" of all networks - the telephone system. With my PC at home, I can find your system by having it randomly dial thousands of numbers. Eventually, I'll connect with YOUR system.

Once I am able to sign onto your system, I'm probably more than halfway finished. All I need to do now is to get the capabilities and information that I'm looking for. On nearly every system, this is a trivial task, as most system managers fail to adequately protect their valuable data.

The ultimate hacking feat is to acquire PM (privileged mode) or SM (system manager) capability - with either, the system becomes an open book. I can then access any file - payroll information, manufacturing production data, financial information, or whatever else happens to reside on your system. Acquiring and effectively using PM capability typically requires a little understanding of the internal workings of MPE - but by now procedures for using it have been widely published. Acquiring SM capability may simply involve a search for other logon passwords. Most installations still have them conveniently located in the first line of stream files. Typically, files starting or ending with 'J' or 'JOB' are good candidates.

If this doesn't work, I may be forced to use more devious methods. One program I use is a simple BASIC program which prompts the terminal with a colon. When an unsuspecting user tries to enter a command, my program prints a message indicating they must first logon (EXPECTED :HELLO,...). Naturally, they enter their HELLO command. I ask them for the passwords and record all the information to a file - my personal password directory!

Another method is to search certain groups for any program files which have been :RELEASEd (or to which I have write access). If the group has PM capability (such as PUB.SYS or PRV.TELESUP), the system is history. All I need to do is an FCOPY, replacing the contents of the program file with a simple program which, when executed, grants me all capabilities (a trivial, well documented procedure, using PM).

If I can logon to a user which has OP capability, I can use the tape management privileges granted to this type of user to either :RESTORE a program into PUB.SYS which will give me capabilities, or create a SYSDUMP tape which contains all of the passwords on the system (then a simple FCOPY from the tape, and MANAGER.SYS is mine).

My personal favorite is to use standard system utilities which have been :PREPared with PM. If I can access DEBUG (privileged or not) from within these programs, I can alter the program's logic to provide me with the capabilities I'm looking for.

After achieving the necessary capabilities, I can roam the system freely, searching for any valuable information. My choices here are nearly endless. Depending on what I want to accomplish, I can purge files, databases or entire accounts. Discovering database passwords and altering their contents becomes easy (changing my salary, for example). Just as easily, I could put your system out of commission by trashing the directory, halting the CPU, forcing you to RELOAD all of your data (you may start begging for mercy now).

A true hacker never even lets his or her victims realize that they've "been had". My next step is to cover my tracks. Much of my activity so far may have been recorded into the system LOG files - a :SWITCHLOG and purge will do the trick. Or, FCOPYing another LOG file to take it's place numerically, may cause less suspicion (so that there does not appear to be a missing file).

I don't want to forget about the system console - I may want to erase any attempted logon messages I may have generated when trying to find an entry to the system. If the console output is not being printed, I will clear the console's screen from my terminal. A number of utilities exist which will allow me to send any string of characters to another terminal (the proper escape sequence will home the cursor and clear the screen).

Now that I have erased my tracks, I will usually leave behind some type of "backdoor" - a method to allow easier access to the system for next time. This may be a privileged program file which I store in an inconspicuous place. I often hide it amongst the drivers in PUB.SYS - who knows what all those things are for anyway? A name like HIOTERM3.PUB.SYS usually suffices.

Of course I will take the opportunity to generate and record a list of all the account, group and user passwords on the system - just in case the logon I used this time is gone next time. Altering an account's capabilities to include PM may also come in handy and make running a PM program very easy in the future. To hide this, I may programmically alter the directory (with DISKED5) to give a group PM without altering the account, thus making the change less conspicuous. If the console is

constantly monitored, I may create a new account with PM capability. Typically, I will take an account that is used heavily (say, MGR.PROD, or USER.PAYROLL), and create an account with the letter 'O' changed to a zero. Logons to this account could then go virtually unnoticed.

:RELEASEing a program file in a PM group, or altering the security to allow Write access is one more "hole" I may create. Finally, one "backdoor" that is relatively short-term, but nearly impossible to detect, is to add a privileged segment into the system SL. This procedure could then be called (from something as simple as EDITOR), to grant all capabilities to the user. This is only short-term, since an UPDATE or COLDLOAD from a tape which does not have this segment will remove the routine.

Detecting and Recovering From Violations

There you have the confessions of a hacker. As was mentioned, it may be difficult to determine if an unauthorized user was even accessing your system. But as you can see, appropriate procedures must be in place at all times to both prevent illegal accesses (as best you can) and attempt to determine if one occurred.

Periodically examining your system LOG files, whether a suspected violation occurred or not, is always a good idea. Particular attention should be paid to logons from any modem devices, off-hours logons, and any invalid logon attempts. The same type of log file analysis applies to database log files - off-hours database modifications should be investigated. If, through examining the system log files, a logon occurred by a suspected violator, the recorded session number should be cross-checked with other logged events (file closes, etc).

Also, consider using a hardcopy device as the system console. This will give you an audit trail of all logon, logoffs and invalid attempts. The system message catalog can be modified to highlight these messages when printed.

Establishing Security Controls

Regularly review each account and group on the system to insure that no accounts have been added or their capabilities altered. Remember to examine group capabilities as well, since a privileged user could force group capabilities to be greater than the account capabilities. In addition, insure the accounts and groups have the proper security access and that for any privileged group, no files

have been :RELEASEd. Account for every file in PUB.SYS - any PM programs which may call DEBUG should be restricted (ALTSEC or lockwords).

If you have modems on the system, consider using call-back devices (although not fool-proof if call forwarding is used). A logon UDC and a relatively simple program can be set up to add an additional layer of access security for these devices.

ALL accounts on the system should have a password assigned. These passwords should be a minimum length and not be any default (i.e. TELESUP). Passwords for "sensitive" accounts should be changed frequently, and possibly have the users of the account assign the password, so the need to write it down is minimized. Consider purchasing a security package (such as HP SECURITY) which gives you greater control over access to various features of the system. Use of private volumes is an excellent method of removing sensitive data from the system, when it is no longer required to be on-line.

Control access to the system backup tapes - remember, they contain the same data you are trying to protect. Use a :STREAMing utility which does not require you to place the passwords in the stream file. They can be automatically substituted at :STREAM time, or may prompt the user. Prevent users from inadvertently leaving their terminals signed-on by using any number of available idle logoff utilities. Whenever possible, automatically place the user into a menu interface which will control what functions he may perform - it is always easier to limit what someone can do rather than list everything that can't do.

Change the system message catalog, replacing the current messages (such as "ACCT EXISTS, USER NAME DOESN'T") which effectively "lead" a person attempting to logon, by a generic "INVALID LOGON" message. For highly sensitive information, encryption techniques are available which may prevent a user from retrieving "sensible" information, even if they do somehow gain access to a file.

While it may seem a difficult task to restrict unauthorized users from gaining access to your system, it is not impossible. The most important point to keep in mind is that system security does not just "happen", it requires a continuous effort on behalf of the system manager.

Office Automation:
Managing Your PC-based Word Processing Documents

Rick McCahan
Jerry Johnson
Sandi Fruehling
Harris & Paulson, Inc.
6251 Greenwood Plaza Boulevard
Suite 122
Englewood, CO 80111

You walked into the office one morning and found that 200 PCs had been installed while you were on vacation. That's the good news. The bad news is that the Vice President's secretary lost a 40 page document overnight and the data processing department is being held responsible because you didn't back up the PC.

Sound scary? Although this may be an extreme case, similar situations are happening everywhere. Traditional data processing departments are being called on to insure the integrity of PCs out in the world that are being used by naive, untrained users who have never heard the terms "backup", "head crash" or "oops, I purged all my files!". Data processing managers have two possible plans of action; ignore the problem and hope the users know that they are responsible for everything or merge the PCs into the host-computing environment that now exists so that data integrity throughout the company is assured. Those managers who take the first course of action had better have their resumes in order! This paper will present a possible scenario that allows the DP manager to not only improve data integrity, but also to incorporate PCs into the environment so that profitability is increased due to better sharing of work and leveraging (pardon the marketing term) of existing work.

In the host-based shop, managing and sharing word processing information is technically easy because everything is located on the same computer. Control is easy because there are established procedures for getting access to the host, backing up the host, and recovering the host in the event of a failure. Hardware is rapidly repaired because the vendor's customer engineers understand that a host computer or host peripheral device failure leads to loss of productivity for a large number of

people -- you have their attention! Those are the good points. The problems are: performance, everyone wants to use the computer to do everything on the 29th of the month and response time suffers; redundancy, when the host goes down everything stops; file management, "I was in the word processing section last month and I created a document that dealt with marshmallows, fiber optics and data bits but now I can't remember what I called it or even how I logged on."

Like magic the PCs come in and the performance and redundancy problems are solved. The PCs have other benefits too, otherwise there wouldn't be so many of them. But what are those benefits? Okay, here they are. Low cost, MIPS are real cheap in a PC; "friendly" user interfaces, face it folks, PCs defined the term; personal flexibility, you like LOTUS 123, but the VP of marketing likes VISICALC; performance, no one is competing for the resources; no requirement for redundancy, if a PC is down it affects only one person not the entire company. Now for the bad points: control, if the user doesn't back up the PC it doesn't happen; data sharing, only if someone happens to mention it over coffee.

Alright, we've told you things you probably already knew. So why did we write this paper? First, to discuss how you can solve the control and management issues. Second, to discuss how you can personally become a hero be increasing everyone's profit sharing slice.

The PCs must be integrated into the overall strategy for integrity which is currently in place for the host computers. Ideally, the PCs should automatically be backed up to the host computer's tape drive during off hours so that an inadvertent file deletion or hard disc failure does not mean the total loss of all the work that a user has done for the past several months. This backup should be taken out of the hands of the end user as much as possible. The best case would involve a menu choice for the users marked "GO HOME" to be selected as they are searching for bus fare. The host system polls the PCs during the night and backs up all files that were updated during the day. These files are dumped to tape and a listing is generated showing which files from what PCs are loaded on a specific tape. A report is also generated detailing which PCs did not respond to

the poll so that when a user complains because the PC failed and "the computer guys can't recover it," you can show management that the user failed to perform that one small step.

There are other useful parts of this strategy. Management can determine how much work is produced by the PC user during a given day based on files created or edited. The backup tapes can become a company archive for PC documents so that PC disk drives are not constantly being added. Instead, users are secure in knowing that they may periodically clean up their disks because "those wonderful computer people" can always get back a document that is stored on tape. If a PC fails, the "computer people" can also get the user's PC files back so that the last three months don't seem like a dream.

The PCs should be integrated into the entire company strategy. PC users must be able to create electronic mail messages in their favorite PC-based word processor. That message should then be easily uploaded to the company's electronic mail system to be transported anywhere in the network where other users can read the message on their terminal or download it to their PC for further revision and retransmission. The user should not need to be aware whether a process is being performed on the local CPU or a remote host. Make the interface as "seamless" as possible while maintaining indicators for your in-house troubleshooters so that they can tell from naive descriptions of problems whether the user is having trouble with the message creation function in their word processor or with the mail interface to the host computer.

Okay, now for the stuff heroes are made of. Have you noticed how one of your programmers writes this great subroutine and that subroutine starts appearing in programs all over the system. How do you think management would look at you if you spread the capability of using existing work all over the company? Here's the scenario. A PC user in the San Francisco office spends two weeks writing a proposal to a large customer in that area dealing with the installation of XYZ Company's new widget analyzer. The proposal is accepted and 200 analyzers are sold. Another PC user in Miami is dealing with a new customer who might have a need for widget analyzers. This user remembers some comments about the San Francisco proposal made

during a recent sales meeting. The things he remembers include the cost saving to the customer, the proposal was presented in the first fiscal quarter, the customer used stainless steel buckets. The Miami user logs on his PC, selects "Proposal Archive" from the menu and fills in a screen with the information he remembers about the subject of the San Francisco proposal. The company-wide archives are searched and the user in Miami finds a copy of the proposal given in San Francisco. Instead of having a week's worth of work to do, this user can now use pertinent portions of work performed elsewhere in the company and completes his task in one day.

How does all this work? The user in San Francisco wrote the proposal on the PC. Before saving the document the user was prompted for summary information regarding the document: the author's name and company branch office, the type of document (proposal, letter, contract, etc.) and a brief summary of the contents. The PC was backed up during the night to the local host computer in San Francisco and the text and summary information of all the documents was loaded into a centralized text management database in Los Angeles. The Miami user accesses the central database in LA and performs a text search for the phrases "cost saving", "stainless steel bucket" in all documents created between 1/1/87 and 3/31/87. The text management software presents the user with three possible documents. The authors' names are reported along with their office locations. The documents themselves are no longer available on-line but have been archived to tape in LA. The text management software asks the user if the documents should be restored from archive and mailed to Miami via electronic mail. While this is in process, the user in Miami calls the authors of the other documents and gets more specific information on the accounts which can then be used to more effectively market the product to the client in Miami.

So we have used a host-based solution to solve the control and data sharing problems that PCs don't handle. But what about the text management problem. Neither the PC nor most host-based systems have an effective text management system that can do the kinds of searching and document retrieval that the Miami user needs to do to prepare the second proposal.

Just as the adage in real estate advises that the three most important considerations when buying are "location, location and location", the three most important considerations in designing a good software system are "data structures, data structures and data structures". In both cases, accessibility is paramount. Neither the right location nor the ideal data structures in themselves guarantee success, but without them results will be mediocre, at best. What then would be an ideal solution for storing, accessing and retrieving textual data? If you look at a typical office, the work product might include anything from a one or two line memo to a contract or user's manual of hundreds or even thousands of pages. In addition, it may be necessary to maintain various revisions of each document or associate multiple comments or notes as part of the same "logical" record. In short, the basic requirements of a text management system are:

- Fields may be of any length
- Fields may be repeated zero or more times
- The fields that comprise a particular database must be definable by the user to model the actual application as accurately as possible
- Adequate support structures must exist to allow for data retrieval

In addition, certain practical requirements must exist or users will be very reluctant to take advantage of such a system. Some of these requirements are:

- Ease of use, both in database management and in accessing data
- Access to data should be permitted by a variety of convenient forms such as looking for all or part of a word, looking for a phrase, looking for words within a certain proximity of each other and looking for synonyms or words that sound alike. Keyword in context searches showing words with surrounding text are also very useful
- Access must be reasonably fast even on very large documents
- Changes to not only the contents of the database but also the structure should be allowed in an easy and timely fashion.

Textual databases are unfamiliar to many data processing people. Most computer applications involve some type of transaction processing and well defined, fixed-length records. In a text tracking system, a single word of a document is a value just as an invoice amount is a value in a billing application. Therefore, the problem of managing text involves the document as well as every word in that document. It is important to distinguish document management applications from more traditional non-textual applications. This is a case when you need to hope for the best, program for the expected and provide for the worst. It is also a case where the rigid formats of traditional data structures (hierarchical, network and relational) are not well suited to the unpredictability of the data. This unpredictability also creates far more complex problems in dealing with real-time updates by multiple users. Several years ago during a presentation of a document retrieval system for medical information, the question was asked "when are updates made to the database?". The answer: Christmas. For those of us for whom "tonight" is usually the wrong answer, that is a luxury we'll never know.

Looking at the advantages of PC-based word processing, it is easy to see why this solution has become so popular and why that popularity will continue to increase. The task of improving the management of documents in a PC environment involves solving the disadvantages by utilizing the strengths of a host system to develop a complete and efficient integrated solution.

Protecting Your Investment in Your Staff

by
Donald P. Gholson
Innovative Software Solutions, Inc.
10705 Colton Street
Fairfax, VA 22032 USA

One of the major resources in computing is the people who have contact with the computing environment. We often tend to concentrate our efforts on the hardware and software and ignore the major investment we have in people. The "people cost" in our field is rising.

The quality and knowledge of our staff determines the quality of services to our users. The knowledge level of our users determines how well they can perform their jobs and the level of satisfaction they will have with the services we provide.

We have a major investment in people and in order to protect the investment we must properly train the staff and users to effectively and efficiently use the computer resources we provide.

This paper will discuss several training methods and provide insight based on past experience. It will concentrate on methods that can use readily available resources within your own organization and inexpensive alternatives.

According to Terry Lundgren and Norman Garrett in the June, 1987, issue of Modern Office Technology, the major reason for office automation failures has been a lack of adequate training in new skills and procedures. This applies to all facets of computing as well. They continued, "true training is a process which familiarizes the system's users and makes them proficient with the hardware and software. All modern ... technologies will benefit from training in increased user productivity, reduced initialization time, or both, but few offices consider the psychological and sociological elements involved in making the computer a working partner."

"User participation in the implementation process should always involve a large component of training, ranging from a modest introduction to the equipment and procedures to learning the full capabilities of the system, but the uses of the system beyond the obvious capabilities should always be considered."

Anthony Carnevale, vice president and chief economist of the American Society of Training and Development, stated in the same issue of Modern Office Technology, "Most people tend to learn on their own. They go to the courses and then learn to use the machine so they can respond to their own needs. That's why learning on your own is so effective." The need has to be there to drive the person and the person must be given the resources necessary in order to have a learning environment.

Training has historically been treated as an afterthought by the computing profession. In the early years of computing, training was an "on the job" activity. Many people entering the field were doing so through graduate school experience, where they had worked in the labs on computer development projects.

When computers came out of the labs and into the commercial world the pool of workers was the engineering, mathematics and science graduates. Training continued to be "on the job" where experience was gained by working with experienced team leaders.

By the 1960's, some of the major computer manufacturers were offering classes in the more technical and vendor-specific topics. These classes supplemented those few courses being offered in colleges and universities. The difficulty with the vendor classes was that many were so technically intensive that one needed a wealth of knowledge and experience to gain anything from the class.

As the sale of computer systems increased in the 1960's and 1970's, a wider variety of academic disciplines was represented in the computing field. Computing was no longer limited to engineers, mathematicians and scientists. Business majors, liberal arts majors and other disciplines began to enter the field.

Attempts were made to adapt training programs to handle this diversity. In-house courses were offered and programmed instruction packages were used. The basic problems were that the classes were primarily lecture classes and the programmed instruction packages still retained a heavy technical content.

This was still the period of heavy batch processing. Multiuser, multiprogramming computer systems were in their infancy. Computing was still restricted to a small, elite group within each organization. The availability of

adequate training opportunities tended to restrict the field even further.

Robert Townsend, former president of the AVIS car rental company, wrote a chapter in his book Up the Organization about dealing with computers and the computer center staff. He aptly titled the chapter "Computers and Their Priests". In his book, he described how the computer center staff tended to be a breed apart from the mainstream of the organization. They tended to treat their equipment the same way religious zealots treat idols. The computer center staff even invented a new vocabulary in order to make their little world appear mysterious and forbidding to "outsiders".

Townsend's observations quite accurately reflected the thinking of both management and organizations in general regarding how they viewed computing. The people in computing were considered aloof, secretive and somewhat anti-social. They usually spoke computerese (technobabble) and were difficult to communicate with. While the same can be said of today's computer people, the degree of alienation is much lower.

I remember attending a meeting where the president of a large state university delivered the address. The audience was composed of computer center directors from colleges and universities. The president described the typical university's organization, not in the traditional terms of faculty versus staff but in terms of data processing people versus everyone else. There were many nods of agreement from the audience. He went on to say that it was nearly impossible for anyone to cross the line between the two groups. This event occurred only a few years ago.

These observations are made for a reason. While we may have tried to train ourselves and our personnel to be very proficient technically, we have not directed out training to deal with the "outside" world very effectively. One of the main reasons is that we rarely speak to non-computer people in plain, everyday English.

The above observations lead one to look at the causes. One of these goes back to Townsend's observations - computer people try to be different, to build some mystery into their work. This is not limited to the computer field alone. Many other professions also do this. Another major cause is that most computer people have never been trained to deal effectively with person-to-person relationships.

Studies made during the 1970's and early 1980's indicated the changing personality traits of the computer professional. They indicated that the professionals were becoming less introverted and more people-oriented.

As interactive systems became more common in the 1970's and 1980's a new phenomenon arose, the non-dp user. The advent of the terminal in the office or work place, moving computing out of the computer center and to the users, brought new challenges to the computer center staff. One of the major challenges was to educate/train/teach the new users about computing. The need was to provide sufficient information for the new users to do their jobs using the computer. What occurred was usually something else.

Quite often the computer center sent some of its programmers or analysts who were familiar with a specific application out to teach the users. The results were usually disastrous. The technical professionals knew their topics, but couldn't communicate. They tended to use highly technical jargon and lacked the patience necessary to work with their students. They also lacked an understanding of their students' problems and fears.

The training offered by the computer center usually turned more people off than helped them. The technical staff wanted to talk about binary and octal number systems, while their students wanted to know what key to press to enter a name change to the payroll system. The technical staff wanted to talk bits and bytes, discs and tapes. The students wanted to know if the computer would explode if they pressed the wrong key.

Some organizations got desperate and hired training coordinators and training staffs. They hired outside consultants and training organizations. They spent large sums to send their personnel off to classes. But they still continued to be frustrated. The results were costly and not always satisfactory.

As a computing services director, I faced the same problems. The executives in the organization wanted to rapidly expand the computing network, installing terminals in every office. They wanted to hire a training coordinator, but did not want to spend the money necessary to do the job properly. They failed to realize that if the training coordinator was not familiar with the hardware and software and if the training coordinator lacked experience with the computing environment of the organization, more damage than good would be done.

Since the computing center was to be left out of the training decisions, some of us acted on our own to assist the users. The computer center had been directed to install a large number of terminals in offices as a result of the development of a broadband local area network system. Many of the offices had no other application than using HPMAIL. Most of the users were totally unfamiliar with the computer and were afraid to use the terminals.

As Dorothy Neal, president of the National Business Education Association put it, "the first thing you teach them is not to be afraid of the computer. Teach them basic functions, then the most common applications. The best way is to demonstrate how easy it is to use; then put them at the computer with a helper, who is the teacher. I believe that when you have two people working together as partners at the computer, the student gains new confidence." While we didn't have the people to assist as helpers, we did get the users to the computer.

Two of us decided to use the computer itself to train the users. We developed a modular, interactive package which gave these new users some basic information necessary to better use their terminals. At the same time, it made the them use their terminals and become familiar with them. HP already had a good training package for HPMAIL. We developed our own driver and modules to introduce the users to the HP 3000 and to introduce them to the fundamentals of using TDP.

The software was well received. Fortunately, the users felt free to comment on its good and bad points. As we revised it over a period of months we were able to answer most of the problems. The key was to write the modules in a language that the user could understand - non-technical English. By not trying to overwhelm the users with technical terms, we were able to get our information to them in a much more acceptable form.

The original version of the software covered such topics as keyboard layouts, standard utilities like LISTEQ and LISTDIR and commonly used commands. The software was expanded to show users how to set up softkeys on their terminals and how to use some of our home-grown utilities.

The results were satisfying. There was a marked decrease in the number of phone calls for assistance on basic operating procedures. There was better acceptance of the terminals in the offices, leading to increased use of the computer. This

acceptance generated more requests for a variety of applications. The users developed a better understanding of how the computer center operated, its abilities and its limitations, and the computer center developed a better understanding of the needs of the users.

With the normal turnover in the computer center, we needed some introductory training for our new employees. The two introductory modules, one on the HP 3000 in general and one on TDP, were very useful. Even though they were written in as non-technical a manner as possible, the computer center staff did not feel that the material had been diluted. They quickly realized that it was possible to describe many of their tasks in "plain English".

More modules were developed. One was a module on converting from COBOL (1968) to COBOL II (1974). It was felt that this would be useful for programmers who knew COBOL and had worked on non-HP systems to convert to COBOL II. The change from COBOL to COBOL II involved more than just changing EXAMINE to INSPECT and recompiling programs. There were over one hundred new reserved words in COBOL II, though not all of them were implemented by Hewlett-Packard. Once we got into it, we were quite surprised at the changes in the language.

Another module was developed to introduce IMAGE to new programming staff members. While not intended as an in-depth discussion of IMAGE, the module continues to grow and its scope has expanded to include TurboIMAGE.

The intent of the system was not to be the sole source of training for users and staff. The system was intended to supplement other means of training. Not all people learn well using an interactive system, just as not all learn well with lectures or reading manuals. Many administrators and executives were hesitant to sit in classes with their subordinates. Many couldn't schedule the time. The training software allowed all users at all levels to learn about the computer at their own speed and in whatever environment they wanted - in their office, in a lab, etc.

Many used the training software as a preparation for more formal lecture classes. We found that it provided a basic foundation from which they could progress. It prepared them to ask better questions in class and they made better use of class time.

We added a work book for the TDP lesson material. At one time we had provided printed copies of the interactive TDP lessons, but the users wanted some examples they could work with.

We have found that there is no single best method to train users and staff. Some prefer one specific method, others require a variety of methods, such as lectures supplemented by interactive lessons and work books. In cases where a large number of users must be reached and consistency in lesson material is desired, the terminal based lessons seemed to work very well.

Recent studies have shown that responsiveness to computer training may vary depending on the type of intelligence exhibited by the individual. This can greatly affect how a person learns and which methods are the most effective.

Users whose intelligence is predominantly verbal do well by reading documentation. Those with a logical-mathematical pattern and those with a musical sense respond well to programmed instruction methods. Users with strong spatial or artistic intelligence do best when first presented with an overall concept, while those whose focus is on their own or other's feelings learn best with individual or small group instruction.

Terminal based training worked well for new users at all levels in the organization. It was a non-intimidating experience that could be handled at a personal level. For executives, it meant that they could schedule the time to fit into their busy schedules. They could progress at their own speed and not be embarassed in front of subordinates if they did not progress as fast as everyone else. We made sure that the users had a member of the computer center staff whom they could call to answer their questions. Usually this contact person was high enough in the computing center staff to work effectively with all levels of the organization.

In situations such as the establishment of an information center, computer-based training software can provide an efficient means of bringing users up to speed in a short amount of time. It also can provide a means for a small staff to assist many more users than can normally be done in such a situation. An advantage is that all users receive the same information and the information can be updated in one place, instead of having to deal with multiple copies.

Since much of what is covered is constantly changing (i.e., changes in MPE or in software systems, new procedures, etc.), updates to training material must be scheduled on a regular basis. This also permits new topics to be introduced and new features to be added as a result of user requests.

One of the keys to training is flexibility. Training should not be limited to a single medium or method. Within a medium or method there must be room for change as the users' needs change. Even in a stable staffing environment, where there is less and less need for training new users, there is still the need to expand the knowledge base for current users.

Whatever system is used must be able to adapt to meet the needs of the users as they become more sophisticated. It is not necessary that the individual learn everything by rote and commit it all to memory to be regurgitated at a later time. The key is to be exposed to topics, concepts, and ideas and be able to remember the exposure to the degree that you can find the proper reference material for it.

As an example, it is not necessary to remember all of the verbs and their exact syntax in COBOL. It is sufficient to remember that when given a specific task to be done, such as appending to strings of characters, to know that the language is capable of handling it and then to be able to find the proper reference for it, in this case the STRING command.

After the initial contact with the computer based system, some users later used the material as an on-line reference system. If they were not familiar with a command or concept from their day-to-day use of the computer, they would often use the training software to review the topic. Sometimes this would lead them to reference manuals, which they found easier to understand after using the computer-based system.

In conclusion, we found terminal based training to be a valuable complement in a comprehensive training environment. It provided a solid base from which users could operate. Furthermore, it made the staff, both within the computer center and in the user community, aware that their employer was willing to invest in them in order to enhance their knowledge. This led to higher morale and better cooperation between the computer center and the offices it served.

HEY! How do I get this thing to print?

Craig Gilmore
Hewlett-Packard Response Center
3300 Scott Boulevard
Santa Clara, CA 95054

The decrease in the price of Laser Printers, and the growth in knowledge of the user community, has created a desire to learn more about the functionality of these printers. This paper will attempt to give the reader a more in-depth view of printed output using a Laser.

This discussion is intended for the person already familiar with a Laser Printer. However, a general user who wants to explore the possibilities and uses of a non-impact printer should have no trouble following along.

We will discuss how the HP3000 interacts with a System Laser printer, and how the 2680 and 2688 work with the Spooler and associated Intrinsics to produce output from an Environment file.

The LaserJet family, and how they work with the 3000, will also be covered. Printer Control Language will be discussed, as well as the various escape sequences used by these printers. Finally, there will be a comparison and contrast of the large and small laser printers, turning to a discussion of current questions.

FOPEN

The discussion of getting image to paper will start where the system starts -- FOPEN. Each and every file operation on the 3000 starts with an FOPEN. The spooling system is no different. When a spoolfile is created, the first control placed in it is an FOPEN command. Starting in this direction will help show the differences in each of the HP Laser Printers.

Part of the FOPEN statement, or FILE equate, is the **ENV=** parameter. This parameter is used for a variety of controls and commands. The option is discussed in four paragraphs with the FOPEN section of the Intrinsics manual, and it is also mentioned in the Commands manual with the FILE command. These short discussions do not explain the power of this option.

The first thing to remember is that an ENV= parameter on a :FILE command overrides the ENV= statement in a FOPEN call. When the parameter is present in either case, several different sections of the Operating system are called. The file specified is checked to make sure that it has the proper file code. The file referenced in the ENV= parameter, must have a file code of PENV or TTYPE. An FSERR 155 will result if this is not the case. Then the record size is compared, to further check the validity of the file as an Environment file.

Environment Files

What is an Environment file? Put basically, the Environment file contains all the information and parameters needed to make calls to FDEVICECONTROL. This includes a bit map of each character font, each form, and the parameters to describe the logical and physical pages.

The format of exactly how an Environment file is set up is not going to be discussed in this paper. This is because the format may change as the operating system is enhanced. Some sections and tables will be covered to help in explaining how the Laser works with the system.

One of the questions asked most often at the Response Center is: Can I use the Environment files in the HPENV group of the SYS account with my LaserJet printer? The answer to this is NO. There are many reasons for this, but the main one is that the LaserJet printers do not have a CPU built into them. This allows for their lower cost, and wide use range.

You can use a LaserJet with almost any computer that communicates via RS-232C. The 2680A and 2688 are restricted to use with the HP3000.

The function of the CPU, in the big lasers, is to parse and break down the 512W blocks the printer receives from the spooling system. See figure 1-1 for the layout of a spooler block. The CPU organizes this information into several tables. The first of these tables or blocks are the I/O status block and the Environmental status block. The I/O status block does not contain much useful user information. See figure 1-2 for the layout of the Environmental status block. These two blocks, used together, tell the system exactly what is going on with the printer. The LaserJet family simply uses an error code displayed on the front of the printer, to inform the user of an error.

The Status blocks are set up by the printer. What is to be printed is sent by the Spooler. The HP3000 operating system converts the information contained in an Environment file to several calls of the intrinsic FDEVICECONTROL. This intrinsic was first designed for the 2680A Laser Printer. It has now been expanded to work with Terminal Type files, and NRJE spoolfiles.

FDEVICECONTROL

FDEVICECONTROL is documented in the MPE V Intrinsics Reference Manual. Richard Oxford, of MCI DISC, wrote an excellent paper on the Intrinsic in the Interex Detroit Conference proceedings. I would suggest that anyone wanting to print more directly to the PSP Laser printer, read his paper. He goes into detail of how a print image is set up, and the basics of printing to the laser are also covered. The control codes used with FDEVICECONTROL are documented both in the Intrinsics manual and in Application Note #26. They will be briefly discussed here to aid in explaining how the big lasers use tables to print spoolfile data.

There are 4 descriptor blocks used, besides the 2 status blocks that have already been discussed. Each descriptor block is part of an associated Load Record. These Load Records are used to create the Logical Page Table, the Character Set table, the VFC table and the Forms/Picture storage area.

Spoolfile Block Format

WORDS	MSB 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 LSB
0	Logical Record Length in bytes +8
1	Number of Data bytes including blanks
2	FDEVICECONTROL Code
3	Parameter 1 of FDEVICECONTROL
4	Parameter 2 of FDEVICECONTROL
5	First Data byte | Second Data byte
6	Third Data byte | Fourth Data byte
n	Last Data byte | Null byte
	Next Logical Record above format is repeated for each logical record
	End of Valid Data Indicator (−1) −− all bits set to 1
509	Option reserved for Spooler
510	Record number of
511	first record in this block

Fig. 1-1

Environmental Status Block

0	Number of Data Blocks		Size of Data Buffer	
1	Number of Memory Buckets			
2	Maximum number of Data Buckets used in this job			
3	Memory size	Number of VFC's	Number of Forms	
4	HP−IB address	No. of active pages	No. of Character Sets	
5	(No. char words + 15)/16 + (prospace words + 15)/16			
6	(No. form words + 15)/16 + (No. Triplets + 15)/16			
7	Number of VFC Words			
8	Page Length		Page Width	
9	Number of Pictures	Not Used	Prt Type	PI ES FM DA
10	(Number of User Area Memory Words Used + 15) / 16			
11	Data Control System Date Code in HP format			
12	Number of Characters Clipped in this job			
13	(Num. of Picture words + 15)/16 + (Triplet Words + 15)/16			
14				
15	Current sheet number in process or Silent run Sheet Number if in Silent Run Mode.			

Fig 1-2

When a new print job is opened or a physical page eject occurs, the big lasers moves the printing pen to a new physical page. Then the logical page table is entered and searched sequentially until an active logical page is found. All data and forms for this logical page is processed until a page eject is found. The printer then searches the logical page table for the next active logical page. This cycle is repeated until the end of the logical page table is found. At this point, the printer performs a physical page eject and the process is repeated until the end of the print job.

This is a very simple explanation of how the big lasers print using an Environment file. I have not gone into too much detail because this paper is intended to show the differences between the large and small lasers. The printers are referred to as Large and small because of 2 main factors. The first being size and the second is price.

Small Lasers

Turning the discussion to the smaller laser printers, HP changed the face of Non-Impact printing with the introduction of the LaserJet in 1984. This small printer brought laser printing into the grasp of the large PC market. The printer was not designed to work with the 3000, however, the user community quickly adapted it to this environment as a remote spooled printer. Now, with the LaserJet Series II and the LaserJet 2000, the face of printing is changing again. These 2 new printers are now officially supported on the HP3000.

The LaserJet, and printers like it, caused a new buzz word to be created: Desktop Publishing. Optical scanners also contributed to this new industry. How do these new printers now effect the market? What can I do to get the most use out of my LaserJet now? Let's deal with the second question first.

Control Language

The smaller laser printers do not create the status blocks and, therefore, cannot use the IFS (Interactive Formatting System) Environment files. Changing the printers to include the ability to create these blocks would probably increase the price of the printers. The size of the printers would also be effected. The fix is to convert the PSP Intrinsic calls to the PCL Escape codes. There are a couple of software packages available that will

translate the PSP (Printer Support Package) commands into Printer Control Language. PCL is now used as the standard for all Hewlett-Packard Printers.

Since the new printers do not work with the PSP Intrinsics, you may feel that their full effectiveness is limited. However, PCL is a very powerful and flexible language. True, it is not as convenient as PSP package, but it can be used across all HP printers. It is not limited to the LaserJets. The 293x and 256x families both work with a subset of PCL. The old 2608 and 2631's also recognize some of the Escape sequences. This allows for wide use of applications and ease of migration.

FOPEN

When an FOPEN is placed against a port that has a LaserJet connected to it, several files are opened. The main file opened besides the data file is the Terminal Type file. MPE V/E T Mit showed a large change in how I/O is handled with devices connected to an ATP or ADCC. Soft Terminal type files were created. Previously, the characteristics of how I/O was handled for a port was hard coded into the operating system. The reformating of the system tables allowed for a new data segment to be used, the Terminal Data Segment. This new segment, and a product called the Workstation Configurator, made it much easier to connect almost any device to the HP3000. It has also caused many headaches.

Workstation Configurator

The Workstation Configurator is almost a "Must have" product for use with a LaserJet printer off of a 3000. This product will create a file that can make the calls to FDEVICECONTROL like a PSP Environment file. The wide and varied uses of this product have only recently come to light. Besides the simple part of parity and data bits, a VFC file is also associated with each terminal type file. This VFC file contains an initialization area. This area allows the user to create a string of escape sequences that will be sent to the device when it is first opened. The area can be used to create a pseudo Environment file.

You can include all the necessary escape sequences to define the logical and physical page, the Character font used, and where you want to start printing. The VFC can be downloaded from this area, or you can send a short Macro which allows for forms downloading.

There are some disadvantages. Part of the default initialization string is a Hard reset of the device. The hard reset will usually reset anything that has already been set up for the device. On MITs prior to UB Delta 3, the information from the Term Type file was stored in the Terminal Data Segment, and not overwritten until the system was Cool or Cold started. This deficiency was corrected in the UB Delta 3 release.

Macros

The Macro is a powerful tool. With it you can down load 32 different series of commands to the LaserJet Plus family. You can use it to print forms, Logo's or do automatic pen positioning. Now, with the LaserJet Series II memory, you can have close to the printing power of the large laser printers.

Summation

The large laser printers, and how the HP3000 sends controls to it, were discussed and we covered the LaserJet flexibility. Status blocks and the control languages also played a part. Hopefully, this paper has given you some ideas of the differences in the Hewlett-Packard laser printers. It is intended to start the thought process on how to print with a non-impact printer. The technology of the printers allows for almost anything to be put to paper. The only major limitation is your imagination.

Compiler and Optimization Technology on HP Precision Architecture

Jason M. Goertz

Mattedor Computer Services
Bellevue, Washington

Introduction

A facet of the RISC paradigm and therefore HP Precision Architecture is that only the most basic of instructions be implemented in the machine, and that complex operations be created by stringing together these simple instructions. Because the instructions are implemented directly in hardware, they tend to be relatively simple when compared with the types of functions available in a microcoded Complex Instruction Set Computing (CISC) machine. The lack of microcode in RISC computers makes the hardware much cheaper to build and much faster, but necessitates that complex operations be implemented by combining a series of simple instructions in a particular sequence. In HPPA, this job falls upon the compilers.

This paper will examine this very important aspect of Hewlett-Packard Precision Architecture. A discussion of some of the software conventions will be followed by a general discussion of compilers and compiler technology. After this, code optimization will be discussed, with an emphasis on the specific techniques utilized by the HPPA compilers.

Software Conventions

A good example of a complex operation implemented in CISC machines is the procedure call, executed via the PCAL instruction on an MPE V-based HP3000. A corresponding instruction, EXIT, exists when the procedure must return to the caller. The Hewlett-Packard Precision Architecture does not define any instruction pair analogous to PCAL/EXIT. However, it is still necessary to provide the ability to call procedures and exit from them. Rather than provide hardcoded functionality in the hardware (and thus create expensive and difficult to build designs), this is accomplished by establishing standards in the way the registers and data spaces are manipulated by the compilers.

The software standards being discussed are considered to be part of the architecture just like the instruction set and other facets of HPPA. Some of the standards have written documentation.[1] We think of these standards as being as much a part of the machine architecture as much as the PCAL and EXIT instructions which interact with the MPE V-based HP 3000 data stack and code segments.

Actually, the standards are stringently enforced conventions which are implemented by the compilers. Just as PCAL and EXIT deal with both the code and data portions of the machine, so too does the software conventions deal with code and data. The data stack in the user's data space is the center for a large number of

conventions. Additionally, the code segments have certain formats and standards which must be adhered to and are enforced by software modules.

Software Register Conventions

Before discussing the specific hardware and software conventions, the use of the machine registers must be examined. Certain of the general registers, referred to as Rn (where n is a number from zero to 31) have specific purposes and even have names associated with them. Some of the more important ones will be discussed here. In all cases, unless otherwise noted, the registers contain short pointers. Short pointers are 32 bits wide, and contain an implicit reference to a space pointer, delimiting an addressing area up to 2**32 bytes long. Since the space register to be used is implicit, short pointers depend upon a space register being pre-loaded by the system. Typically, this will be SR5 register for delimiting user data areas.

Register R2 is referred to as the Return Pointer, or RP. This is the address to which the IA register must be set when returning from the procedure call, and is analogous to the P-relative return value in the stack marker on the MPE V-based HP 3000. This return addresses is formed by the "Link" portion of a branching instruction with a link function. An example of this is:

```
BL   $LABEL_0,2          ;Branch to $LABEL_0, place the
                         ;return location in R2
NEXT_INSTRUCTION         ;<-Address loaded into R2
```

The Branch and Link instruction is used only for branching within the same code space. Thus, the return address represents only the offset within the current space. Although the instruction set allows any register to be used as the object of a BL instruction, the software conventions of HPPA dictate that R2 be used. If the procedure being called does not call any other procedure, then the return address loaded by the BL instruction is untouched, and used for the return. If the procedure does call others, then R2 is saved and restored as necessary. The entire procedure calling mechanism will be examined later.

Other registers are used as pointers to the various data areas. Register R27 is called the DP, and serves as a base address for global data. The equivalent on the MPE V-based HP 3000 is the DB register. Using DP as an address to access data can be illustrated by the following PASCAL addition statement:

```
VAR
   i,j    : integer; (assume i at DP+80,j at DP+84)

BEGIN
  i := i + j;
END;
```

The assembler for this construct could be:

```
LDW  80(0,27),10         ;Load DP+80 into R10
LDW  84(0,27),11         ;Load DP+84 into R11
ADDT10,11,12             ;Add R10 and R11, trap if overflow
```

```
STW  12,80(0,27)          ;Store the result to DP+80
```

Register R30 is called the Stack Pointer, or SP. This is roughly equivalent to the S register in the MPE V-based HP 3000, and is used to delimit the top of the current stack. The prime difference between the stack used in HPPA and that on MPE V-based HP 3000's is that, as we have seen, calculations are not performed on the stack, but directly in the registers. The stack in HPPA is used primarily as a parameter and state saving mechanism. This will be illustrated fully when the procedure calling convention is discussed.

Philosophically, these registers are like their MPE V-based HP 3000 counterparts. They are given meaning by a series of fundamental instructions that execute in one or two clock cycles. The only difference is that these instructions reside in RAM (both main memory and cache), not in ROM, as is the case with microcode. Access to the RAM in this case is as fast as the ROM used in most CISC machines because of the intervention of CPU I-cache memory. Indeed, the Writable Control Store (WCS) on the Series 68/70 uses this type of static RAM memory. The main difference between the CISC microcode and HPPA machine code is that the microcode tends to be thought of as part of the hardware. In HPPA, philosophically speaking, the microcode is now visible to the programmer.

Other registers serve a variety of purposes at different times. Four registers are defined as argument registers (ARG0-ARG3) are GR23-GR26. These are used to store the first four parameters when calling a procedure. RET0 and RET1 (GR27 and 28) are used to contain the functional return value from a procedure. There are areas within the stack frame to accommodate these values; the registers serve as an optimization method, since the values in do not have to be moved to and from memory if the register already contains the correct data.

The Procedure Calling Sequence

The actual calling sequence that has been alluded to before is relatively sophisticated. As with the PCAL and EXIT instructions, the sequence has distinct events which happen within the code, and events that occur in the data area on the stack. The data stack convention will be discussed first, then the code.

The techniques illustrated below are not new. Many machines in the past have not had a microcoded calling sequence, such as PCAL/EXIT. In fact, the HP 3000 was a bit revolutionary at the time of its inception because of this procedure calling capability.

For purposes of the discussion, the code (procedure) performing the call will be referred to as the caller, and the procedure being called as the callee. The sequence has four parts: steps performed by the caller on exit, the callee on entrance, the callee on exit, and the caller on re-entry. The first two are roughly equivalent to the PCAL instruction, and the last two to the EXIT instruction. The following discussion examines these in detail.

Refer to Figure 1 for a diagram of the stack. the first task of the caller is to preserve any registers that it has used, will use again, and will not be saved by the callee. This is done by loading them into the stack in the call save area. Since the compiler knows what registers it uses for certain operations, it only

has to save these registers. Upon branching to the callee, the caller must not only pass parameters to the callee, but must save the state of the machine. If the callee has four parameters or less, then they are loaded into R23-R26. These are referred to as ARG0-ARG3. If any more are required, the caller loads the remaining parameters into the argument list area. The caller then sets RP (R2) and branches to the callee.

Upon entry, the callee first allocates the area of the stack that it will need. This area is called a stack frame, and is reserved by simply incrementing R30, the Stack Pointer (or SP). This can be seen in Figure 1. When FOO was called, it incremented SP to the point labeled PSP. When FOO called BAR, BAR incremented SP to the point labeled SP. FOO's SP then became the Previous Stack Pointer, or PSP. After SP is incremented, the callee saves the entry save registers in the entry save area of the stack. If any local variables are to be accessed within a nested procedure, the value of SP at entry is saved at SP-4. The value of RP is saved at SP-20, if this procedure calls any others. The Native Mode stack marker format is shown in Figure 2.

Upon exit, the callee must return the stack to the state that existed before the call. The callee restores the registers it saved in the entry save partition, sets the functional return (if any) in R28 and R29 (RET0 and RET1). It then resets the SP to the value at entry.

The caller, upon re-entry, restores the call-save registers. This completes the data stack portion of the calling convention. The two different save and restore operations (caller and callee) are separated to optimize the overhead of the entire calling operations. Each entity only saves and restores those registers that are necessary. This is possible because the compiler knows at compile time what registers are necessary and other facets of the calling procedure environment, such as what other procedures are called and thus how many words of parameters are needed.

The code execution path is a very interesting one. First, it must be emphasized again that the only transfer of control instructions defined by HP Precision Architecture are branch instructions. Therefore, any transfer that is done must be within the scope of these branch instructions. The code path taken between the caller and callee differs depending upon whether the callee resides within the same space or a different one. An intraspace call would be performed within the same program, or between two procedures that were bound at link time. If the called procedure resides in a different space, as is the case when calling an intrinsic, then an interspace branch is done. The convention is designed such that the branch instruction generated at the call point is always an intraspace branch, as is the branch executed for the return.

Refer to Figure 3 for a diagram of an intraspace call. For this case, the compiler generates a BL instruction to an address within the space. The "link" part of the BL instruction causes the return address (the next instruction after the "slot" instruction) to be loaded into R2. Upon return, the callee loads R2 and performs a BV (Branch Vectored) instruction, which uses the value in RP as its target address.

Refer to Figure 4 for a flow of the interspace call. As previously mentioned,

branch address resolution for interspace calls between the caller and callee is accomplished by a stub. In this case, the external stub performs the linkage of the caller and callee. There is one stub each for the caller and the callee, the calling stub and called stub. When calling, the caller actually performs an intraspace branch to its calling stub. The code contained in the stub acquires the address of the callee from XRT, initialized at load time by the Loader. The XRT contains the SID and address of the called stub for the callee. The calling stub loads a space register with the SID, then executes a Branch External (BE) instruction, which facilitates the switching of code spaces. The stub is logically divided into an entry part and an exit part. After the caller executes the BE instruction the machine enters the entry portion of the called stub. The entry portion of the called stub executes a BL instruction to transfer to the actual entry point of the called procedure and store the return address in R2. Note that this is an intraspace branch. When the called procedure is finished, it must return to the caller. To accomplish this, the called procedure executes a BV instruction. This uses the return address stored in R2 by the BL instruction. This transfers to the exit portion of the called stub, which performs another BE instruction. The target for the BE is the return point of caller, which continues its execution.

It is interesting to note that the code in the body of the procedures themselves only perform intraspace branches. Only the stubs ever perform an interspace branch. This greatly simplifies the code required for the compiler to generate, and isolates the more complicated BE instruction sequences external to the mainline of the code.

The above sequence can be made a bit more complex out of necessity. For example, the PASCAL language supports nested procedures, or procedures defined within another procedure. The nested procedures are only known to the outer procedure, and all nested procedures can access data defined locally in a procedure of higher level. To accomplish the addressing of data within a higher procedure, the PSP must be saved in the stack marker. This adds another step to the callee entry procedure. When the data is accessed, an indirect reference is created by the compiler by loading the appropriate SP from the stack marker and calculating the address of the data relative to that SP. This adds a bit of overhead for data access, but is still faster than a typical indirect access on a CISC machine.

The data state sequence optimizations are possible with this type of scheme because the compiler knows a great deal about the environment of the program being compiled. It can dynamically determine whether registers need to be saved, thus eliminating unnecessary instructions. In essence, the calling sequence can be longer or shorter if need be. With a microcoded machine, the same microcode sequence is performed every time, and the microcode has to determine certain things dynamically. For example, the HP 3000 PCAL instruction has to determine whether or not an external or internal PLABEL is being called, and then go to another level of indirection if external. In HP Precision Architecture this is performed at compile and link time, and the proper exit and entrance stubs are generated, thus hardcoding the decision. XRT lookup is not hardcoded, but this is an extremely simple sequence of only eight instructions. The PCAL must make many slow memory accesses to calculate the address of the object code segment (if external).

Again, it is apparent that the different software pieces must work together. The linker must generate the stubs, and the loader generate the XRT. One component of the software puzzle that has not been addressed well is the compilers, which clearly have the most complex part of the calling convention above. It is worthwhile to examine in depth the compilers used for HP Precision Architecture.

HPPA Compilers

The integration of the compilers into the total scheme is crucial for any RISC architecture to succeed. A great deal of what makes a RISC-based computer system really work must come from the compiler. Additionally, a great deal of the technological progress that has occurred in the past few years that make RISC a viable computing alternative has been in compiler technology. To totally understand the ramifications of this, it will be helpful to examine compilers in general and the HPPA compilers specifically.

A Short Course in Compilers

How compilers work has been the object of much research in the past 25 years in computer science. When the first Fortran compiler was produced at IBM in the mid-1950's, the state-of-the-art at the time was assemblers. At that time, writing a compiler was almost totally an art. Little, if any, systematic study had been done on languages, and therefore the Fortran that was produced was crude by today's standards. Starting in about 1960, many of the eminent computer scientists began investigations of the many phases of computer languages, and created a sub-field that was much more a science than was the case when the first Fortran was written. What follows is a description of the various parts of the compilation process.

The purpose of a compiler is simple: translation. Symbols corresponding to a (hopefully) well defined language are fed in, and the compiler produces output that conforms to different languages. In the compilers that most people deal with, the input language takes the form of words that exist in human languages and symbols drawn from mathematics. Different computer languages draw different amounts from the human and mathematical sides of the fence. The language COBOL, for example, was purposely patterned after English. The statements all begin with a verb, have objects, and end with a period. The statement "MOVE TRASH TO GARBAGE-CAN." is a valid COBOL statement. Mathematical formulae take on a much lesser role in COBOL, as the language was designed to handle applications which do not require a great deal of mathematical manipulations. The Fortran language, on the other hand, was designed for engineering and scientific applications which deal heavily with formulae. Fortran uses a few English words, such as WRITE, READ, and IF, but shines when arithmetic expressions are needed. Other language elements can be used as well. Special characters such as parenthesis, semi-colons, etc, are used heavily by some languages. The language LISP, for example, makes heavy use of parenthesis, and C uses the left and right braces ({}) to delimit compound statements.

A computer language is defined by specifying a series of formats that constitute valid statements in the language. These formats are called **productions**, and the set of productions together form a **grammar**. English defines a correct sentence

as having a subject (which is a noun) followed by a verb and optionally an object. Thus, the sentence "Jane sews." is a valid English sentence, as is the sentence "Jane sews the blouse.". Several notational schemes have been devised to rigorously define grammars. By far the most common notational method is called **Backus-Naur Form**, or **BNF**. In BNF, the symbol ::= signifies a definition. The symbol to the left is called a **non-terminal**, and is defined by the symbols to the right of the definition symbol, which can be a combination of non-terminals, or **terminals**. Non-terminals are usually inclosed in inequality symbols (<>). In English, the first part of the sentence, the subject and the noun, can be defined as the words "Jane" and "sews" respectively. They can also be defined as "Joe" and "drives" or any other sequence of nouns and verbs. The object is optional, and can thus either be an object or null. From this verbal description, the BNF for a basic english sentence would be:

```
<sentence>      ::= <first_part><second_part>;
<first_part>    ::= <subject><verb>;
<subject>       ::= Jane|Joe;
<verb>          ::= sews|drives;
<second_part>   ::= the <object>|(null);
<object>        ::= blouse|car;
```

The symbol | indicates an or state, so the subject can be "Jane" or Joe". The (null) indicates that the second part can be omitted. Thus, the sentence "Joe drives" is valid according to this grammar, and the sentence "Joe drives the car" is valid. Notice, however, that the sentence "Joe drives the blouse" and "Jane sews the car" are also valid sentences. We have not violated any of the rules of how to construct sentences, so the **syntax** of our english subset has been obeyed. What has been violated is the **semantics** of the language. The sentence "Jane drives the blouse" has all of the nouns, verbs and objects in the correct places, the words are from the allowed combinations, and the spelling is correct, but our minds tell us the sentence is meaningless. Semantics of a language translate to the meaning or intent of the language. The following Pascal is valid syntactically, semantically incorrect:

```
VAR
   R1, R2 : REAL;
   I1     : INTEGER;
BEGIN
   I1 := R1 * R2 * 100.433;
END;
```

The assignment statement is a valid Pascal statement, as it conforms to the syntactic rules for assignment statements. However, the real result produced on the right side of the assignment operator is not compatible with the integer on the left. Another rule, external to the grammar, must be specified to remove this ambiguity. Three solutions are that the real result can be rounded and stored into I1, the real result can be truncated and stored, or an error can be generated. Most implementations of Pascal will do the latter.

The output language is also well defined. In this case, the term language might be somewhat confusing to a human user. In most cases, the output language of the compiler is the machine language instructions of the computer upon which the

program is to run, which compiler writers refer to as the **target machine**. This language is defined by the hardware designers, and must be adhered to rigidly if the computer is to do useful work.

The steps the compiler goes through to perform this translation are numerous. When discussing these steps, it is important to not try to correlate the logical steps with what the compiler actually does. The physical processes that occur within the compiler many times combine several of the logical steps in single procedure or operation. Refer to Figure 5 for an overall flow of the compilation process.

The first step performed by the compiler is called **lexical analysis or scanning**. Conceptually, this step is the simplest, but in practice it can be among the most complicated. Scanning is the process by which the "words" of the input string, called **tokens**, are separated. Scanners usually examine the input text one character at a time and determine the start and stop of the token. This token is then available to other parts of the compiler when necessary. For example, in the COBOL statement:

```
MOVE ADB-ACCOUNT-NO TO OUTPUT-ACCOUNT-NO.
```

the tokens are MOVE, ADB-ACCOUNT-NO, TO, and OUTPUT-ACCOUNT-NO. In this example, the tokens are easy to discern by the scanner, since they are all delimited by either spaces or a period. A more difficult example in the SPL statement below:

```
WHILE xxyy(a, b:=4, IF a THEN 1 ELSE 2*43, aaa(i*3+4)
    DO a := 3 + j;
```

For clarity, the reserved words are upshifted. Since SPL is not case sensitive (unlike C), the compiler upshifts the input string. In this example, the symbol xxyy represents a logical procedure, and the scanner must determine its end by seeing the left parenthesis. In the assignment of b:=4, the symbol b is delimited by the colon, and the colon equals (:=) must be deciphered as an assignment statement. The symbol aaa is again delimited by a left parenthesis, and by inspection we cannot tell if this is an array and i*3+4 is the index, or aaa is a procedure with one integer parameter, passed by value. It is sometimes the job of the lexical analyzer to determine what the token is, sometimes just to figure out where they are. In the latter case, determination of the symbol type is done by the **parser**.

There are many algorithms for parsing. Essentially, parsing is performed in order to figure out how the different tokens are to be treated. It is usually in this step that semantic analysis is done. For example, the statement:

```
i1 := aaa(i*3+4);
```

could be semantically correct or incorrect, depending upon the type of symbol aaa. If aaa is an integer array, or an integer procedure with one integer parameter passed by value, then the statement is syntactically and semantically correct. If aaa is an untyped procedure, then the statement is semantically incorrect.

Another very important part of the parsing process is deciding how to deal correctly with algebraic expressions. The different algebraic operators must be given their proper precedence, in order for the computation to be correct. The statement I := 4+8*3 is correctly handled if the 8 is multiplied by the 3 and 4 is added to the product, and store the result of 28 in I. If the parser just performed the calculation from left to right, the result would be 36, which is incorrect. Typically, the output of the parsing step is to generate an intermediate code structure used by the next step.

One other step that is implicit in the lexical analysis/parsing phase is **symbol table management**. The symbol table is a data structure that is used to store data about the different variables, labels, and procedures that are encountered. The table can then be used to look up the symbol later. Such items as the variable data type, the address in memory, and initial value can be contained in the symbol table.

After parsing, the compiler performs **code generation**. This step is the final one where the target machine code (or whatever output language is desired) is produced. This step can be done in one of two ways, with optimizations being done either before or after code generation. The difference here is whether or not the code optimization algorithms use the intermediate form, optimize it, then generate code, or the optimizations are done on the object code itself. Code generation is perhaps the most difficult step of compilation. A great deal of knowledge about the target machine is necessary to generate good code. This raises an important point. It is not good enough to just generate code; the code generated must be **good** code, and ideally it will be the **best** code possible. Best here usually translates to being fast. Nowhere is this more necessary than in a RISC environment like HPPA. Since there is a tendency to need many machine instructions to perform even simple tasks, it is critical the compiler not only try to generate as few instructions as possible, but generate the **least** number possible. This is where optimizations come into play.

Code optimization is the last (or next to last, depending upon the compiler) step in compilation. The term optimization is perhaps a misnomer, since it is not always possible to know the **optimal** code sequence. A better name for this part of the compiler could be code reducer. The optimizer is such an important part of the compilers for HPPA that optimizations and the optimizer will be discussed separately, after the overall design philosophy of the HP Precision Architecture is examined.

Compiler Structure

As mentioned before, the compilers produced for the Spectrum Program show a high degree of integration and leverage of other parts of the program. The reasoning behind this statement will now be examined.

The compilation process consists of a set of distinct steps. The first three steps (lexical analysis, parsing intermediate code generation) are usually called the **front end** of the compiler. The code generation and optimization steps are called the **back end**. Since the steps are conceptually distinct, it is possible to make them physically distinct as well. This is exactly what has been done in the language

lab in HP.

Refer to Figure 6.[2] The top part shows the front ends of the various languages. These front ends consist of the scanner, parser and UCODE generators for each language. Given the profound differences in language grammars and philosophies, the front ends of the different languages will vary a great deal in the way they handle the input string, and the way symbol table management is done. For example, since COBOL supports multi-level data items, the symbol table management must be of a significantly different nature than Fortran, which does not support record structures. Another example of the types of difference dictating different front ends is the language philosophy. Fortran statements are positional in nature. Additionally, Fortran ignores all intermediate blanks, and allows reserved words to be used as variables. For example, the symbols "DO 100 I = 1,200" and "DO100I = 1,200" are equivalent. The scanner of most Fortran compilers must scan ahead and determine from the context whether the symbols are reserved words or variables. Pascal, on the other hand, does not allow redefinition of reserved words, and has a virtually unambiguous grammar. The Pascal compiler's scanner and parser is therefore very different from Fortran.

Most of the front ends (except C) generate a common intermediate code called UCODE. UCODE is a language that represents a "generic" 32 bit stack based machine. UCODE operators resemble assembler language, but is not a real assembler for a real machine. This UCODE interface represents a clean common language that all compilers can generate.

The UCODE package then generates a very low level intermediate language called Spectrum Low Level Intermediate Code. While UCODE consists of the assembler for a mythical 32 bit machine, SLLIC represents an interface to describe the individual HP PA instructions in a consistent manner, and in a form that can be subsequently used by the optimizer. SLLIC is the package that actually builds the object file, and manipulates the various header records in the file.

It is possible for a compiler to generate SLLIC code directly. This is done by the HP C/XL compiler. The UCODE is conceptually at a higher level and is therefore easier for the compiler to generate. For this reason, most of the compilers generate this type of intermediate code. However, it is possible that future compilers may generate either.

Code Optimization

A great deal of the viability of RISC and HPPA stems from the fact that the compilers can now be made to optimize the code output from the code generator. This section will examine some of the philosophies and techniques available,[3] and talk specifically about how the HPPA optimizer works.[4,5]

Essentially, the optimizer works by examining the code that is generated, performing different types of flow analysis, and then decide what code can be eliminated, changed, or rearranged in the code flow for faster performance.

Before looking at specific cases, a small example will be presented. The statements are SPL, and the assembler code is that of the MPE V-based HP 3000.

Consider the following code:

```
 PROCEDURE ADDEM(M);
      INTEGER M;
BEGIN

 INTEGER J,K,L;

 J := 4;
 K := J + 3;
 L := K + 10;
 M := L + 22;
END;
```

This sequence of SPL statements would generate the following MPE V-based HP 3000 instructions:

```
 LDI 4;  << LOAD 4 ONTO STACK >>
 STOR J; << STOR INTO J       >>
 LOAD J; << LOAD THE 4        >>
 ADDI 3; << ADD 3 TO J        >>
 STOR K; << STORE INTO K      >>
 LOAD K; << RELOAD K          >>
 ADDI 10;<< ADD 10 TO K       >>
 STOR L; << STORE INTO L      >>
 LOAD L; << RELOAD L          >>
 ADDI 22;<< ADD 22            >>
 STOR M; << STOR INTO M       >>
```

Assuming that M is the only variable that is used, we can treat J, K and L as temporaries that will never be reused. An optimizer uses **data flow analysis** to determine this property within a program. The declaration and usage of each variable is charted throughout the program and it is determined where a variable must be saved and where it is not longer needed. In the above example, we see that there are several sequences where a number is stored, only to be reloaded in the next instruction. If we eliminate these sequences (and leave the result on the stack) we can eliminate a number of instructions:

```
 LDI 4;
 ADDI 3;
 ADDI 10;
 ADDI 22;
 STOR M;
```

The ADDI instructions can further be combined into one:

```
 LDI 4;
 ADDI 25;
 STOR M;
```

Finally, the LDI and ADDI both deal with constants, so they could be combined into:

```
LDI 29;
STOR M;
```

The code optimizations just performed could be shown at the source level as:

```
M := 29;
```

A good human programmer (organic code optimizer) would program the last statement.

This is a very simple example, purposely made that way to show how optimizations can be done and why. Two situations occur that require optimizations: inefficient programing by humans, and code generation schemes that are easy to implement, but do not generate the simplest ones possible. The above example shows both of these. The whole source sequence could be considered bad programming. Then, when run through the SPL compiler, the code is generated by looking at each statement without regard to others. This is a common and straightforward way to generate code. A code optimizer looks at the code sequence in a holistic manner and considers the whole block, rather than doing this at code generation time, which is difficult to do.

With this as an example to illustrate the principles of code optimization, lets look at the specific types of optimization that is done by the HPPA optimizer.

Code Optimization Algorithms

The HPPA code optimizer performs numerous types of optimizations. All of the compilers have options to specify three levels of optimizations: none, Level 1 or Level 2. Level 1 optimizations are very minor in nature, and can be done with some change of the object code. Level 2 optimizations go all out, and code can be eliminated or moved at will. The length of the compilation is increased accordingly. Level 2 optimization adds some time to the compiler. The larger the program, the more time is added. Optimizations are generally applied just before code is placed in production.

Two types of analysis is performed by the optimizer on the SLLIC code generated by the UCODE module. The first is data flow analysis, alluded to earlier. In this, the optimizer charts the declaration and usage of each variable. This data can be used to determine when a variable is no longer needed, and how it is used subsequent to any point in the program. The other type of analysis is called **control flow analysis**. This charts out different code sequences and places them in a hierarchy. At the top is the program, followed by procedures, then loops and if-then-else constructs, then **basic blocks**, which are defined as sequences of code with no branches. From these analyses, the optimizer can then examine the code for many types of code sequences, and modify or remove them if necessary. The following are descriptions of each type of optimization.

The following describes algorithms performed when level 1 optimization is enabled.

Branch Optimization

Branch optimizations take many forms. The main idea of these is to eliminate branches whenever possible. If this is not possible, then the branches are made more efficient. A couple of examples are shown below:

a) Branches in which the target is the default:

```
CONST
  a  =  1;

  .
  IF a < 2 THEN GO TO x;
  x: write ('howdee')
```

This becomes:

```
x: write('howdee');
```

b) Branch target is unconditional branch

```
IF b1 THEN
   IF a < 3 THEN p(5);
```

This becomes:

```
IF b1 THEN p(5);
```

Dead Code Elimination

These optimizations detect code which will never be executed and eliminates it, making code sizes smaller:

```
IF a < 4 THEN c := 1 ELSE c := 2;
```

This becomes

```
c := 2;
```

Instruction Scheduler

This reorders instructions in the output stream to avoid register interlocks and improve memory pipelining. This cannot be shown with a source code example. The following assembler code shows an example of this.

```
LDO  -88(0,27),4
LDW   22(0,4),5
LDO  -92(0,27),6
LDW   22(0,6),7
```

Both LDO and LDW pairs will cause register interlocks, as the register being

loaded in the LDO instruction is needed as the address base in the LDW instruction following. The instruction scheduler will reorder these to eliminate the interlocks:

```
LDO  -88(0,27),4
LDO  -92(0,27),6
LDW   22(0,4),5
LDW   22(0,6),7
```

This module also schedules floating point instructions, and reorders branch instructions so that a useful instruction follows, if possible.

Peephole Optimization

Peephole optimizations are conceptionally the simplest optimizations to implement. The peephole optimizer examines sequences of instructions and replaces them with other more efficient sequences. This type of optimizations are done by being intimately familiar with the hardware. The sequences that are examined are from a table, and are built into the optimizer. The following sequence shows an example:

```
LDI          32,r3      ; Load 32 into r3
AND          R1,R3,r2   ; Boolean AND
                        ; r1 with r3, result in r2
COMIB,=      0,R2,L1    ; Branch to L1 if r2=zero
```

Since loading 32 into r3 only turns on bit 26 (counting from the left), r2 can only have 2 possible value, 0 or 32. Thus, the COMIB instruction branches if r2 equals 0, and drops through otherwise.

Since only one bit is being tested in this sequence, a single instruction, Branch on Bit (BB) can be used for the same purpose:

```
BB,>=     r1,26,L1   ; Branch to L1
                     ; if bit 26 not on
```

Level 2 Optimizations

Coloring Register Allocation

This optimization uses a very complex algorithm to trace the use of all registers that have been used. The idea is to prolong the use of a register as long as possible, and eliminate as many load and store operations as possible. An algorithm called graph coloring is performed, hence the name.

This type of algorithm is quite resource intensive, and must build a large number of intermediate data structures to be successful. It is largely because of this optimization that level 2 optimization of code takes so long.

Induction Variables and Strength Reduction

This is really two similar types of optimizations. Induction variables are variables

within loops that have a linear relationship value with the loop counter. Strength reduction uses the property that, on most machines, addition is faster than multiplication, and an addition is substituted for multiplication when possible. Further, if the multiplication is by an even power of 2, then a shift or combination of shifts can be done instead of the addition. (Shifting 1 bit to the left is the same as multiplying by 2).

```
FOR i := 1 to 10000 DO BEGIN
   r[i] := i * k;
END;
```

This becomes:

```
t1 := k;    { temp variable (poss. register) set to 1*k}
FOR i := 1 to 10000 DO BEGIN
    r[i] := t1;
    t1 := t1 + k;  { add substituted for mult }
END;
```

Common Subexpression Elimination

This optimization examines the code for duplicate calculations, and rearranges the code to eliminate as many occurrences as possible. For example:

```
VAR
  I,K,J : INTEGER;

BEGIN
   J := (I+4*K)/12;
   .
   .
   J := 24+3*(I+4*K);
END;
```

The sequence of I+4*K is detected as being duplicated, and is only calculated once. The optimizer will then store the result for later use. If possible, it will store it in a register for the fastest possible access.

Local Constant Propagation (Constant Folding)

This technique locates constants and combines them. The example at the beginning of this discussion shows a form of constant folding. Another example would be:

```
VAR
  I,J,K: INTEGER;
BEGIN
   I := J*4+3+K+5-1;
END;
```

With constant folding, the assignment statement would become:

```
I := J*4+7+K;
```

Another type of constant folding is performed after data flow analysis is done:

```
a := 1;
b := 2;
c := a + b;
```

Becomes

```
a := 1;
b := 2;
c := 3;
```

Loop Invariant Code Motion

This type of optimization is needed when code exists within a loop that does not change over the course of the loop. The code is moved by the optimizer out of the loop so that it is performed only once instead of every iteration of the loop:

```
FOR i := 1 TO 100 DO BEGIN
  a[i] := (3*y+4)*i;
END;
```

This becomes:

```
t1 := 3*y+4;   {t1 is a temp storage location set up by the
                optimizer}
FOR i := 1 TO 100 DO BEGIN
  a[i] := t1*i;
END;
```

Store-Copy Optimization

This eliminates stores to memory whenever possible, and stores the data in a register when possible. The following statement shows this type of optimization. Remember that the calling convention requires the functional return to be placed in r24 and also in the current stack frame at sp-52.

```
Procedure a: integer;  {Pascal procedure with return

BEGIN
      a := x + 23;  {a is at SP-60}
END;
```

Assembler:

```
LDO  23(r26),r1      ;x is already loaded
                     ; in r26. Adds 23 to r26
                     ;places result in r1
```

```
STW 1,-52(0,r30)      ;store result in function return
LDO -52(0,sp),r24     ;store result in return reg
```

Optimized code:

```
LDO 23(r26),r24       ;skips storing to
                      ;the stack frame, since
                      ;this will be peeled
                      ;off on exit anyway
```

Unused Definition Elimination

This optimization eliminates references and declarations of variables that are not used.

```
FUNCTION y(x:integer):integer;
VAR
 a,b,c:integer;
BEGIN
   a := 3;
   b := 8;
   c := x * b;
   y := c;
END;
```

This becomes

```
FUNCTION y(x:integer):integer;
VAR
  b,c:integer;  {a is eliminated}
BEGIN
  b := 8;
  c := x * b;
  y := c;
END;
```

This example can be used to show many optimizations. When all Level 1 and Level 2 optimizations are applied, the above procedure becomes:

```
FUNCTION y(x:integer):integer;
BEGIN
  y := x * 8;
END;
```

Further, if function y is never used, then the code is eliminated as dead code.

Side Effects of Optimization

As has been shown by the above examples, code optimization actually changes the program in some severe ways. Statements are changed or eliminated totally from the code. Data that is stored into variables may not be there when the optimized program actually executes. All of these put together create an object program

which can have little correlation (at the machine code level) to the source program. This has grave implications when the program is to be debugged with a symbolic debugger. Statements that show up in the source code may simply not exists in any form in the final program.

To illustrate this point, let's look again at the function in the previous example. It is listed again here with statement numbers for easy reference. Remember, what is listed is what the user specified for the source code:

```
      FUNCTION y(x:integer):integer;
      VAR
       a,b,c:integer;
      BEGIN
1:     a := 3;
2:     b := 8;
3:     c := x * b;
4:     y := c;
      END;
```

Statements 1-4 become y := x*8 in the final output. If the user wanted to debug this code with a symbolic debugger and set a breakpoint at statement 1, this would be impossible since statement 1 (and 2 and 3) no longer exists in the object code. Another type of anomaly that optimizations cause is the inability to examine data locations. If a breakpoint were set at statement 1 of the optimized version (statement y := x*8) and the user wished to examine a, b, or c, this would be impossible because these variables were not even allocated, much less contain any data.

This whole situation points out one of the biggest problems with code optimization. Debugging code that is optimized is very difficult, although far from impossible. Symbolic debugging is out of the question, and even system level debugging is extremely difficult. The person doing the debugging must be a virtual expert on the code sequences generated by the compiler and optimizer. In fact, this area is one of the hottest areas of research in computer science today. About the only literature on the subject are a few doctoral dissertations. The projects that have been done thus far have had limited success. The reigning methodology is to build a debugger that is honest. If a variable is not defined and the user wants it displayed, the debugger displays a message to the effect that the variable does not exist.

The finest minds in the computer science field do not have a complete and final answer to this problem. Research continues to invent ways to allow a user to intelligently debug code altered by optimization techniques.

Conclusion

Software is an extremely important part of Hewlett-Packard Precision Architecture. Parts of the system that have traditionally been done in hardware or by microcode are now done in the software. Further, because of the nature of RISC instruction sets, the technique of code optimization becomes crucial to realizing the full performance benefits of the hardware. It is by the careful co-design of the hardware and software performed by Hewlett-Packard that this union comes

to fruition in the Spectrum Program.

1. Hewlett-Packard Precision Architecture Procedure Calling Conventions Manual, HP PN 09740-90014.

2. Compilers For the New Generation of Hewlett-Packard Computers, Deborah S. Coutant, Carol Hammond, John W. Kelley. Hewlett-Packard Journal, January 1986.

3. Compilers: Principals, Techniques, and Tools. Aho, Sethi, Ullman. Addison-Wesley, 1986.

4. Compilers For the New Generation of Hewlett-Packard Computers, Deborah S. Coutant, Carol Hammond, John W. Kelley. Hewlett-Packard Journal, January 1986.

5. HP 9000 Series 840 HP/C Programmer's Guide, HP PN 92453-64019.

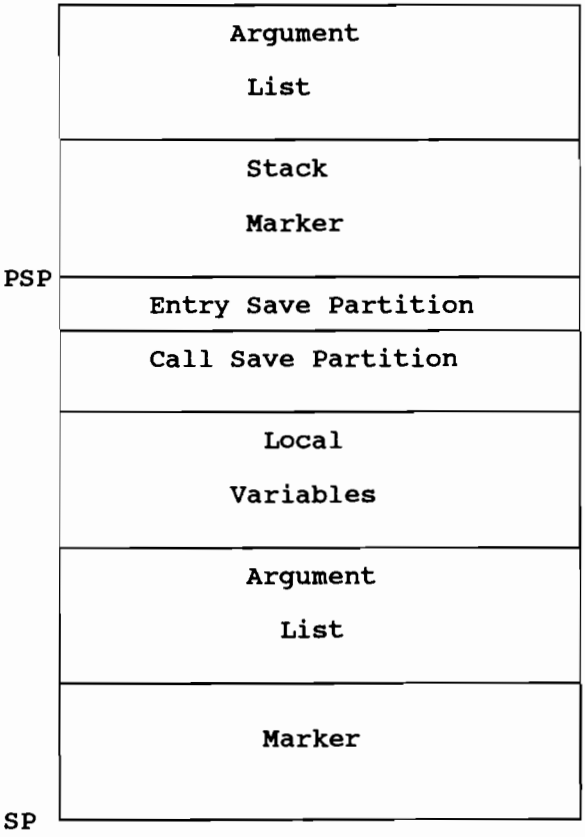


Figure 1 - Native Mode Stack Frame

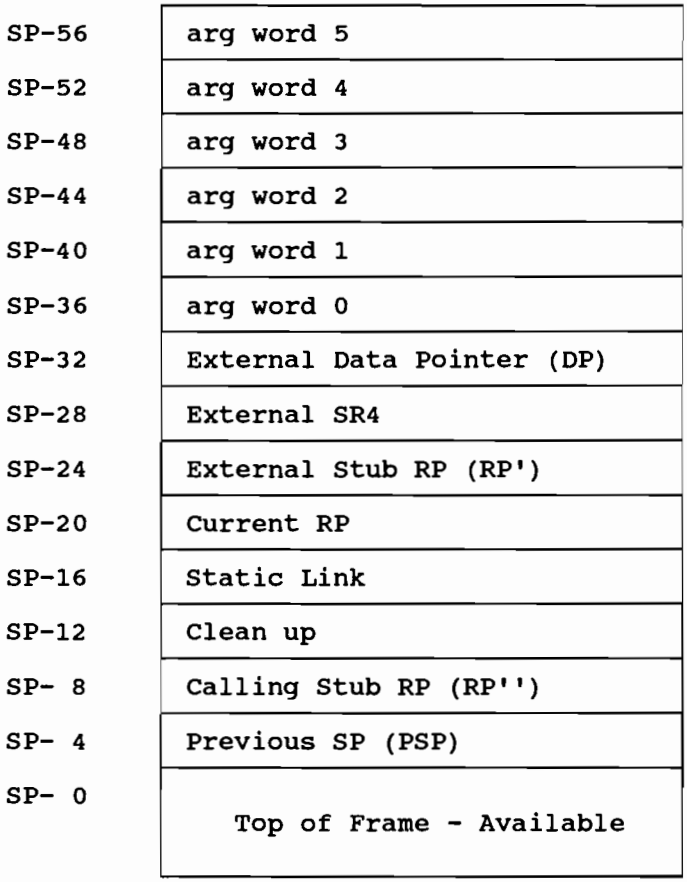


Figure 2 Native Mode Stack Marker

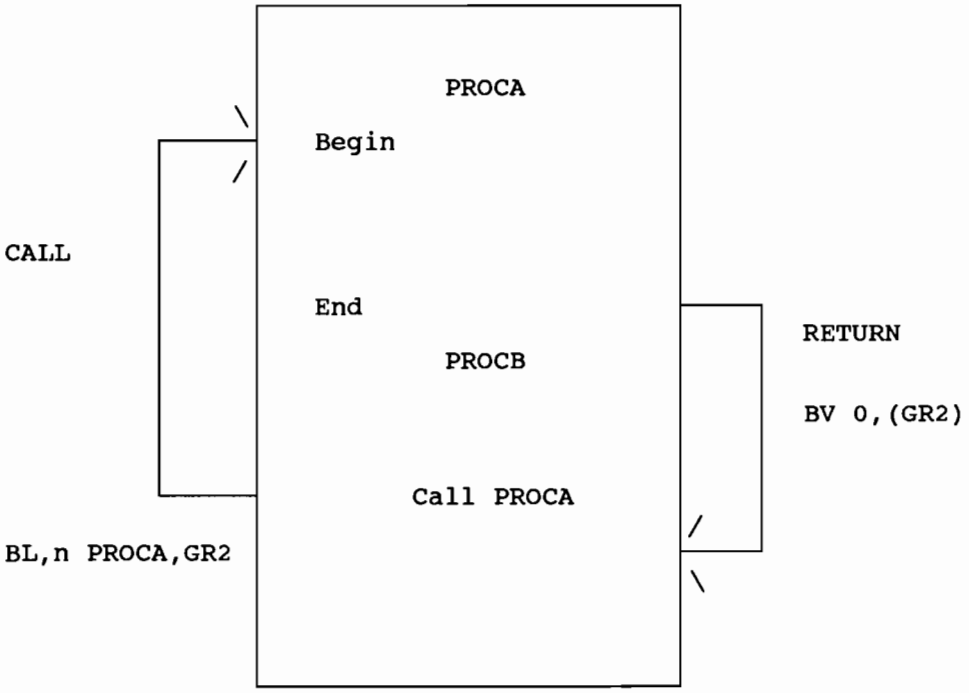


Figure 3 Intraspace Procedure Call

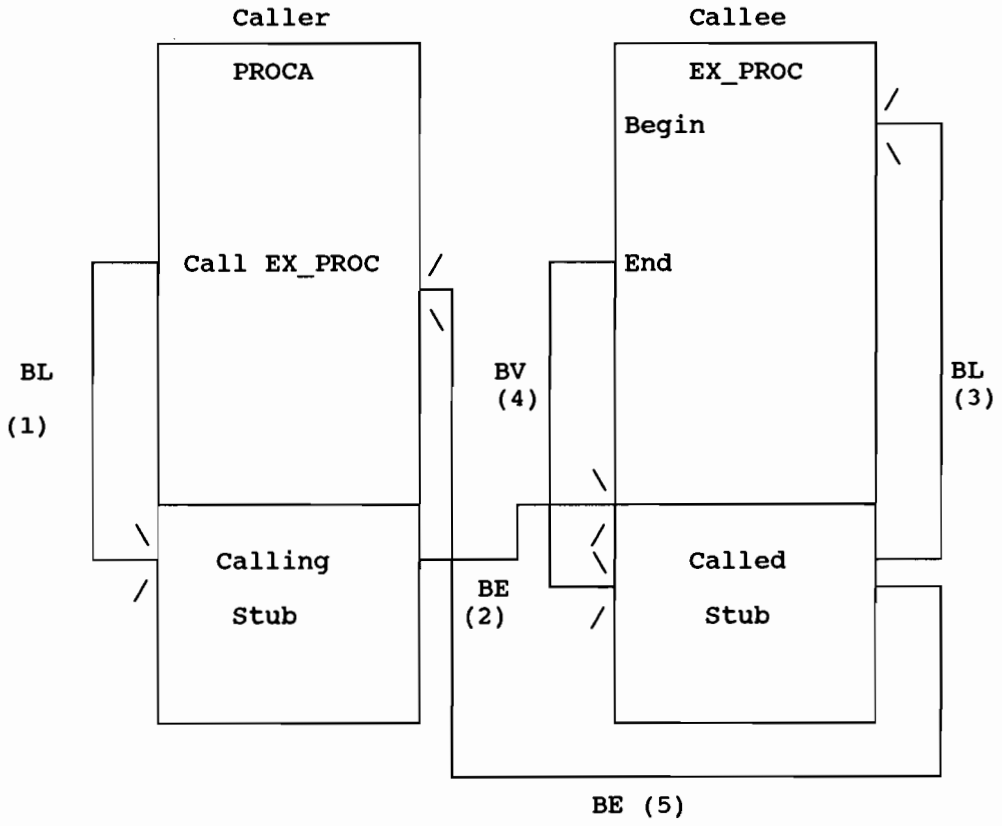


Figure 4 Interspace Procedure Call

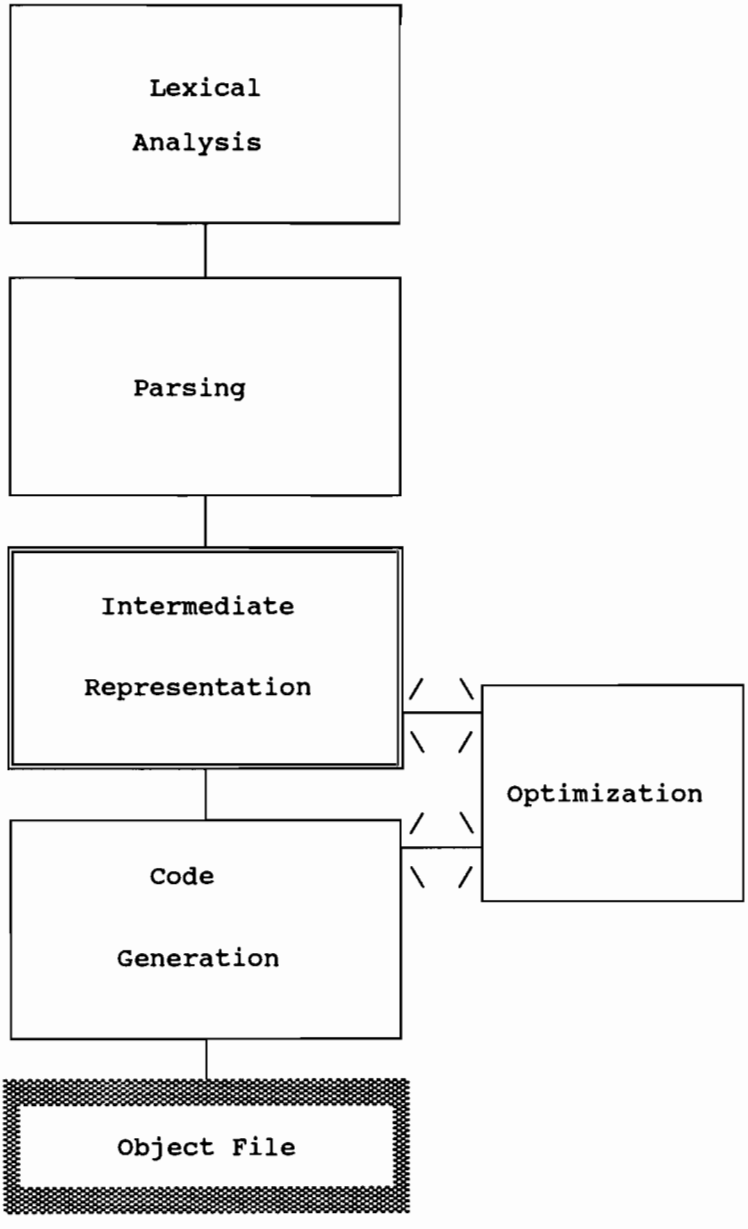


Figure 5 Compilation Flow

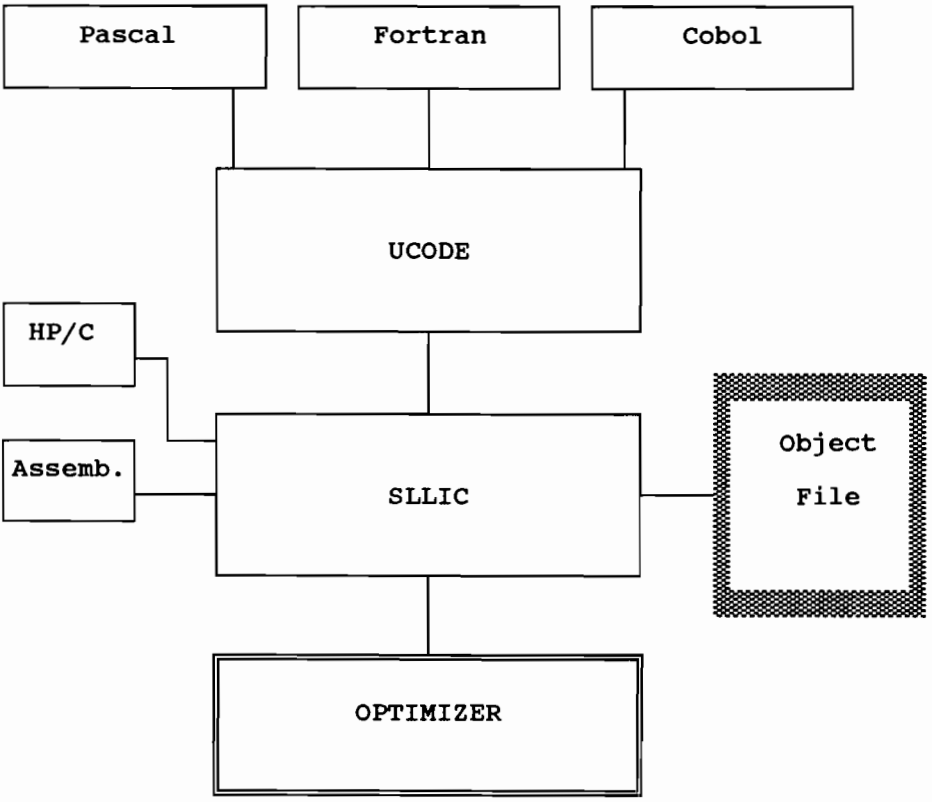


Figure 6 Structure of HPPA Compilers