

Web Services

.NET J2EE XML JOURNAL

September 2004 Volume 4 Issue 9

Toward SOA with ESB

A real-world view from design to implementation
pg.18

Where Web Services Meet Mobile Devices

The challenges to be met
pg.26

Service-Oriented Architecture

Components and modeling can make the difference
pg.34

Integrating XML

The executive initiative
pg.46

ESB Integration Patterns

An insider's look into SOA's implementation backbone

pg 12

RETAILERS PLEASE DISPLAY
UNTIL OCTOBER 31, 2004

\$9.99US \$9.99CAN



09>

0 09281 03420 9

The cost-effective route to the Real-Time Enterprise just became obvious !



Being able to galvanize Real-Time Enterprise aspirations, without “ripping and replacing” your existing investments in programming languages, operating systems, and database management systems, is of utmost importance to any organization today.

To achieve this you need to look no further than Virtuoso from OpenLink Software.

Virtuoso is an innovative Business Integration Platform that unravels the promise of the Real-Time Enterprise vision by alleviating the infrastructural challenges associated with: Universal Data Access, Enterprise Data Integration, Web Services Composition, Business Process Management, and Business Activity Management.

Virtuoso's industry acclaimed product architecture is standards compliant and devoid of any vendor lock-in at the Database, Programming Language, or Operating System levels. It provides the most cost-effective route for realization of the Real-Time Enterprise vision and related technology paths such as the next generation Web of Executable Endpoints and Semantic Content (aka Web 2.0).

Empower your organization today – download a FREE evaluation copy from our Website or call us at 1 800 495 6322.

Database Support

Oracle, SQL Server, DB2, Sybase, Informix, CA-Ingres, Progress, MySQL, PostgreSQL, Virtuoso, and other ODBC or JDBC accessible databases

Standards Support

SQL, ODBC, JDBC, XML, XSL-T, XQuery, XPath, XML Schema, HTTP, WebDAV, SOAP, WSDL, WS-Security, UDDI, NNTP, POP3, SMTP, and SyncML

Platform Support

Windows, UNIX (all major flavors), Linux, Free BSD, and Mac OS X

Runtime Environment Hosting

Microsoft .NET, Mono, Java, Perl, Python, PHP, and others

Programming Languages

Any .NET bound language (C#, VB.NET, others), Java, C/C++ , and others

Web Application Runtime Hosting

ASP.NET, Java Server Pages, PHP, Perl, and Python

www.openlinksw.com/virtuoso/





ARE YOUR DISTRIBUTED APPLICATIONS BROKEN?

Fix them fast.

The SIFT™ solution framework from Service Integrity™ is the leading monitoring and analysis software for XML and Web services applications. SIFT will allow you to rapidly identify and resolve performance issues in your distributed applications. SIFT installs in minutes and automatically discovers your Web services and provides complete visibility into HTTP and XML data flows — all without a single line of custom coding. SIFT will untangle your problems fast.



Service Integrity and SIFT are trademarks of Service Integrity, Inc.



IT Applications of the SIFT™ Software:

- Intelligent Logging to save Time/Expense of Custom Coding
- Deep Visibility into Application Health and Security Status
- Improved Policy and SLA Compliance
- Fully configurable alerting
- Support for Billing and Metering
- And more...

"SIFT has been an invaluable tool and I don't know how we'd live without it. We've been able to monitor system performance and isolate a few bugs that had gone undetected in the past."

Director of Operations at one of the largest commercial banks in the United States



Visit www.serviceintegrity.com for more information and enter to win a **FREE COPY** of *Securing Web Services with WS-Security*, by Dr. Jothy Rosenberg, CTO, Service Integrity, Inc. Fill out our registration form and you will be automatically entered to win!

WSJ: FEATURES

ESB Integration Patterns

An Insider's look in to SOA's
implementation background
By Dave Chappell

10

Toward SOA with ESB

A real-world view from design to implementation
By Bernhard Borges & Ed Kahan



18

Service-Oriented Architecture

Components and modeling
can make the difference
By Ali Arsanjani, Bernhard Borges, and Kerrie Holley

34

PRODUCT REVIEW

DreamFactory Suite from DreamFactory

A user experience for service-oriented architectures
By Ahmed Sako



24

FROM THE EDITOR

SOApbox

By Sean Rhody 7

INDUSTRY COMMENTARY

Pick Your Enterprise Service Bus with Care!

By Robert Davies 8

WSJ: WEB SERVICES @ WORK

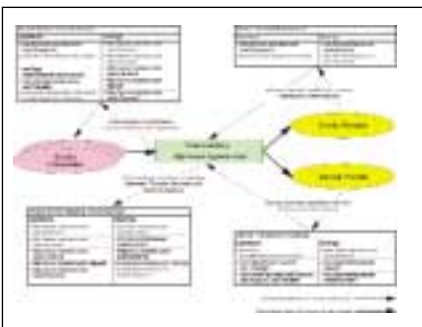
Where Web Services Meet Mobile Devices

The challenges to be met
By David S. Linthicum 26

WSJ: ENTERPRISE SERVICE BUS

SOA and the Art of Riding the Enterprise Service Bus

The need for ESB and other infrastructure technologies
By Rick Robinson 28



WSJ: MANAGEMENT

Bulletproof Web Services

Follow basic principles
By Adam Kolowa 40



Integrating XML

BY GARY KUPECZ
46

Generating XML from Relational Database Tables

BY SELIM MIMAROGU
48

Integrating XSL-FO into Web Based Applications

BY ADELENE NG
52

Middleware is Everywhere.

Can you see it?

4

3

2

1

5

IBM

DB2.

Key

1. Player attempts 30-foot chip.
2. Operator measures distance.
3. Stats entered into PDA.
4. SHOTLink truck transmits data.
5. Broadcaster broadcasts a "birdie!"

MIDDLEWARE IS IBM SOFTWARE. Fans, broadcasters, even players, are accessing every shot at PGA TOUR® events online – in real time. The scalable multiplatform technology of IBM DB2 integrates and manages information, allowing SHOTLink, the PGA TOUR's ball-following technology, to uplink and downlink every shot, run all the numbers and tell the entire story – hole by hole.

Middleware for the on demand world. Learn more at ibm.com/middleware/information **ON DEMAND BUSINESS**

IBM, the IBM logo, DB2 and the On Demand logo are registered trademarks or trademarks of International Business Machines Corporation in the United States and/or other countries. Other company, product and service names may be trademarks or service marks of others. ©2004 IBM Corporation. All rights reserved.

Achieve higher quality in less time (with fewer resources).



SOAPtest®



Jtest®



.TEST®

Parasoft Automated Error Prevention (AEP) Products:

Specifically designed for under-staffed, over-committed development organizations like yours.

Parasoft AEP Products ensure that comprehensive error prevention practices are applied consistently and uniformly across your entire team.

By automating compliance to a unified set of testing and coding standards, Parasoft AEP Products enable every team member to follow the same best practices and the same error prevention techniques – including coding standards analysis, unit testing, code review and regression testing.

Automated testing and standards compliance for any team configuration.

Parasoft AEP Products automate error prevention for Java, C/C++, Web Services, the Microsoft® .NET Framework, Embedded Development, Web Development, Database Development and more.

For details go to: www.Parasoft.com/AchieveQuality

Call 888-305-0041 x3307 Email: AchieveQuality@Parasoft.com



Availability

Parasoft AEP Products are available on Linux, Solaris, and Windows NT/2000/XP.

Contact Parasoft for details and pricing information.

Parasoft Corporation
101 E. Huntington Dr., 2nd Flr.,
Monrovia, CA 91016

INTERNATIONAL ADVISORY BOARD

Andrew Astor, David Chappell, Graham Glass, Tyson Hartman,
Paul Lipton, Anne Thomas Manes, Norbert Mikula,
George Paolini, James Phillips, Simon Phipps,
Mark Potts, Martin Wolf

TECHNICAL ADVISORY BOARD

Bernhard Borges, JP Morgenthal, Andy Roberts,
Michael A. Sick, Simeon Simeonov

EDITORIAL EDITOR-IN-CHIEF

Sean Rhody sean@sys-con.com

XML EDITOR

Hitesh Seth

INDUSTRY EDITOR

Norbert Mikula norbert@sys-con.com

PRODUCT REVIEW EDITOR

Brian Barbash bbarbash@sys-con.com

.NET EDITOR

Dave Rader davidr@fusiontech.com

SECURITY EDITOR

Michael Mosher wsjsecurity@sys-con.com

RESEARCH EDITOR

Bahadir Karuv, Ph.D. Bahadir@sys-con.com

TECHNICAL EDITORS

Andrew Astor aastor@webmethods.com

David Chappell chappell@sonicssoftware.com

Anne Thomas Manes anne@manes.net

Mike Sick msick@sys-con.com

EXECUTIVE EDITOR

Gail Schultz gail@sys-con.com

EDITOR

Nancy Valentine nancy@sys-con.com

ASSOCIATE EDITOR

Jamie Matusow jamie@sys-con.com

ASSISTANT EDITOR

Torrey Gaver torrey@sys-con.com

ONLINE EDITOR

Martin Wezdecki martin@sys-con.com

PRODUCTION

PRODUCTION CONSULTANT

Jim Morgan jim@sys-con.com

LEAD DESIGNER

Richard Silverberg richards@sys-con.com

ART DIRECTOR

Alex Botero alex@sys-con.com

ASSOCIATE ART DIRECTORS

Louis F. Cuffari louis@sys-con.com

Tami Beatty tami@sys-con.com

CONTRIBUTORS TO THIS ISSUE

Ali Arsanjani, Bernhard Borges, Cave Chappell, Robert Davies,
Ed Kahan, Kerrie Holley, Adam Kolawa, Gary Kupez,
David Linthicum, Selim Mimaroglu, Adeline Ng,
Sean Rhody, Rick Robinson,

EDITORIAL OFFICES

SYS-CON MEDIA

135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645

TELEPHONE: 201 802-3000 FAX: 201 782-9637

WEB SERVICES JOURNAL (ISSN# 1535-6906)

Is published monthly (12 times a year)

By SYS-CON Publications, Inc.

Periodicals postage pending

Montvale, NJ 07645 and additional mailing offices

POSTMASTER: Send address changes to:

WEB SERVICES JOURNAL, SYS-CON Publications, Inc.

135 Chestnut Ridge Road, Montvale, NJ 07645

©COPYRIGHT

Copyright © 2004 by SYS-CON Publications, Inc. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy or any information storage and retrieval system without written permission. For promotional reprints, contact reprint coordinator, SYS-CON Publications, Inc., reserves the right to revise, republish, and authorize its readers to use the articles submitted for publication.

All brand and product names used on these pages are trade names, service marks, or trademarks of their respective companies.

SYS-CON Publications, Inc., is not affiliated with the companies or products covered in Web Services Journal.



SOAphox

Yes, I know it's corny, but I've had a few things on my mind regarding service-oriented architecture, component-based development, and even the infamous enterprise service bus. (Hey, at least the title isn't "Get on the Bus"). All of these have something to do with Web services. None of them are "Web services," at least not entirely. And yes, you guessed it, they're all part of this month's focus

So what's the difference between SOA, ESB, and Web services? For starters, Web services is mainly about technology. Service-oriented architecture is mainly about design (although it has to be based on some technology, and Web services is a very good fit for SOA). Enterprise service bus is halfway between technology and design, and focuses on the transmission of messages between applications within an enterprise. Sometimes it uses Web services, but just as often you see people talking about ESB while speaking of protocols such as JMS, without the use of WSDL or UDDI, or even XML.

The enterprise service bus concept hinges on the passing of messages, so it's not surprising that the leading message service software providers are the strongest advocates of an enterprise service bus. It ties into Web services in the sense that Web services provide the ideal API layer for describing messages without binding them to a particular transport (that gets done under the covers by the messaging software, of course). Frequently, ESB discussions will also include some form of business process management product, either a true BPM capability, a workflow engine, or even some sort of sophisticated routing and tracking mechanism. Or even a combination of all of the above.

Enterprise service bus concepts diverge from Web services in that they don't truly adopt the Web services standards throughout, and they have a message-centric approach to interaction that largely rules out the RPC style of communication that is also a part of Web services. You'll see the use of WSDL, but perhaps not SOAP. You may see BPEL, or you may see some proprietary BPM tool. You probably won't see WS-Transaction, or WS-Orchestration. A lot depends on the vendor. As I said, it's a similar idea, but it's not quite Web services.

Service-oriented architecture is even more



WRITTEN BY

SEAN RHODY

divergent from Web services. In theory, you could create an SOA without any Web services technology. In fact, that's been done several times, with technologies such as DCE and CORBA. Service-oriented architecture is about software design centered on services. The goal is to reduce redundancy, increase software reuse, and loosely couple applications (the things the end user interacts with) to services (the things programs and computers interact with). SOA is not nearly as focused on interoperability and intercommunication, except as needed by the enterprise. For example, in an all Java shop there's no need to do SOAP and WSDL. Granted, those shops are few and far between – it's much more common to have a mixed environment, which usually leads to the adoption of Web services as the underlying technologies for implementation of a service-oriented architecture.

Service-oriented architecture provides an overall design philosophy that can be layered on top of Web services, to the benefit of all users. It leverages best practices and experiences from past attempts at distributed computing, and integrates the technologies of Web services into an overall design methodology that can be employed at the enterprise level.

Component-based development is the concept of assembling applications from pre-designed components rather than monolithic programs, or even from objects. Remarkably similar to the concept of services, but with the limitation of being bound to a particular platform, and possibly even language. In many cases, components can become the building blocks from which Web services are created.

So they all tie together in a nice little package. Build your components, put a Web services interface on them. Create the services based on a service-oriented architecture and utilize an enterprise service bus to ensure communication. Add water, stir, and simmer for several hours. Then let cool and serve. ☺

About the Author

Sean Rhody is the editor-in-chief of *Web Services Journal*.

He is a respected industry expert and a consultant with a leading consulting services company.

■ ■ ■ sean@sys-con.com

PRESIDENT AND CEO

Fuat Kircaali fuat@sys-con.com

VP, BUSINESS DEVELOPMENT

Grisha Davida grisha@sys-con.com

GROUP PUBLISHER

Jeremy Geelan jeremy@sys-con.com

ADVERTISING**SENIOR VP, SALES & MARKETING**

Carmen Gonzalez carmen@sys-con.com

VP, SALES & MARKETING

Miles Silverman miles@sys-con.com

ADVERTISING DIRECTOR

Robyn Forma robyn@sys-con.com

ADVERTISING MANAGER

Megan Mussa megan@sys-con.com

ASSOCIATE SALES MANAGERS

Kristin Kuhnle kristin@sys-con.com

Beth Jones beth@sys-con.com

Dorothy Gil dorothy@sys-con.com

SYS-CON EVENTS**PRESIDENT, SYS-CON EVENTS**

Grisha Davida grisha@sys-con.com

NATIONAL SALES MANAGER

Jim Hanchrow jimh@sys-con.com

**CUSTOMER RELATIONS/JDJ STORE
CIRCULATION SERVICE COORDINATORS**

Edna Earle Russell edna@sys-con.com

Linda Lipton linda@sys-con.com

SYS-CON.COM**VP, INFORMATION SYSTEMS**

Robert Diamond robert@sys-con.com

WEB DESIGNERS

Stephen Kilmurray stephen@sys-con.com

Matthew Pollotta matthew@sys-con.com

ACCOUNTING**FINANCIAL ANALYST**

Joan LaRose joan@sys-con.com

ACCOUNTS PAYABLE

Betty White betty@sys-con.com

ACCOUNTS RECEIVABLE

Shannon Rymasz shannon@sys-con.com

SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM

1-888-303-5282

For subscriptions and requests for bulk orders,
please send your letters to Subscription Department

Cover Price: \$6.99/issue

Domestic: \$69.99/yr (12 issues)

Canada/Mexico: \$99.99/yr

All other countries: \$99.99/yr

(U.S. Banks or Money Orders)

Worldwide Newsstand Distribution

Curtis Circulation Company, New Milford, NJ

Newsstand Distribution Consultant:

Brian J. Gregory / Gregory Associates / W.R.D.S.

732 607-9941 - BJGAssociates@cs.com

For list rental information:

Kevin Collopy: 845 731-2684,

kevin.collopy@edithroman.com;

Frank Cipolla: 845 731-3832,

frank.cipolla@epostdirect.com

SYS-CON Publications, Inc., reserves the right to revise,
republish and authorize its readers to use the articles
submitted for publication.

**SYS-CON
MEDIA**

Pick Your Enterprise Service Bus with Care!

The latest hype technology has numerous software vendors scrambling to become buzzword compliant. Analyst groups from Gartner to IDC hail the enterprise service bus (ESB) as the revolutionary technology that will transform middleware due to the vast benefits of adopting vendor-independent standards-based architectures. According to Gartner, ESBs will replace traditional middleware by 2007. So far, however, this "revolution" has seen only a few sparks.

Though just a handful of users have begun deploying ESBs, reports from early adopters imply that the advantages of putting ESBs into service are real. Financial firms such as the Netherlands-based Rabobank say the revised architecture allows their enterprise to phase away legacy MOM products, thus escaping vendor lock-in. They are now able to efficiently and cheaply migrate new and future applications towards open standards.

So What's So Special About ESBs?

The rationale of ESBs is that they are an interface-independent enabler for service orientation – a not-so-new concept that allows distributed software components to follow the flow of information through the business. Just like service-oriented architectures, ESBs are fundamentally a design pattern, but they were originally rooted in Java technology, a light nibble, a relevant pick-n-mix of some core J2EE components.

What's different from traditional EAI solutions is that connectivity is based on open messaging standards. You can interface to your service, hosted on an ESB, via SOAP/XML, traditional message-oriented middleware such as MQSeries or Tibco's messaging, or JMS or a host of other protocols, such as vanilla TCP/IP sockets, FTP, e-mail, etc. ESBs allow organizations to form a universal integration backbone. What is different from traditional EAI solutions is the price. The use of standards means that not only are they cheaper to buy, but the total cost of ownership is far lower because the IT skills required to implement solutions using an ESB are readily available.

Which Types of ESBs Exist?

The ESB space is already getting crowded! If you're convinced ESB is right for your organization, there is a wide range of choice. You can roll your own, and as most of what drives an ESB is readily available in your



WRITTEN BY

ROBERT DAVIES

local J2EE application server, this may be a good place to start.

Traditional EAI vendors, such as IBM, SeeBeyond and webMethods, have or will be deploying ESB offerings themselves, as low-cost entry points to their more traditional solutions. If you are already using EAI products heavily, why change?

Then there are the pure-play ESB vendors who provide lightweight and relatively cheap and nimble products. There are even some good open source implementations, such as Mule, that are joining the fray.

The great thing about ESBs is that because their connectivity is open and pluggable, there is no reason why all these solutions can't run side by side.

So What's the Downside?

ESBs are all about standards, but only from an external point of view.

While they leverage cross-platform standards such as XML, WSDL, and SOAP, their internals, how you access the internal service API for an ESB are still mainly proprietary and are different from vendor to vendor. So while the total cost of ownership may be reduced because ESBs leverage standards-based connectivity, the vendor lock-in trap can still grab the unaware.

What Can You Do?

Choose your ESB vendor carefully. If you are looking for independence and choice, look for vendors who use internal service APIs that are based on already existing connectivity APIs, such as JMS, or the new JSR 208 for Business Process Integration. Assess the connectivity and functional aspects of your vendors: Do they easily separate business logic from the implementation?

Are they manageable, easily deployable, and fault tolerant?

Ultimately, because an ESB is a universal transport bus, which integrates applications with services in a standard way, the cost of replacing one ESB vendor with another solution is going to be relatively cheap! ☺

About the Author

Robert Davies is the CTO of SpiritSoft. He has worked in the computational science industry for over 15 years and is one of the founders of SpiritSoft. As one of the leading experts in the Java Message Service (JMS) Robert's responsibilities as senior vice president of engineering involve the management of developing and designing SpiritSoft's products.

■■■ robert.davies@spiritsoft.com

Web Services Help

Now

LEARN

DEBUG


TEST

TUNE



SOAPscope 3.0

Try it free at www.mindreef.com



ESB Integration Patterns

An insider's look into SOA's
implementation backbone

■ The past several years have seen some significant technology trends, such as service-oriented architecture (SOA), enterprise application integration (EAI), business-to-business (B2B), and Web services. These technologies have attempted to address the challenges of improving the results and increasing the value of integrated business processes, and have garnered the widespread attention of IT leaders, vendors, and industry analysts. The enterprise service bus (ESB) draws the best traits from these and other technology trends to form a new architecture for integration. The ESB concept is a new approach to integration that can provide the underpinnings for a loosely coupled integration network that can scale beyond the limits of a hub-and-spoke EAI broker.

An ESB is a highly distributed, event-driven, enterprise SOA that is geared toward integration. It is a standards-based integration platform that combines messaging, Web services, data transformation, and intelligent routing to reliably connect and coordinate the interaction of significant numbers of diverse applications across extended enterprises with transactional integrity. An *extended enterprise* represents an organization and its business partners, which are separated by both business boundaries and physical boundaries. In an extended enterprise, even the applications that are



WRITTEN BY
DAVE CHAPPELL

under the control of a single corporation may be separated by geographic dispersion, corporate firewalls, and interdepartmental security policies.

An ESB is designed to be pervasive, meaning that it is capable of spanning the extended enterprise. But an ESB is also pervasive in the sense that it is capable of being used as a general-purpose integration environment that is suitable for any project, no matter how large or how small.

The SOA of the ESB

An ESB is the implementation backbone for a loosely coupled, event-driven SOA that

enables a highly distributed universe of named routing destinations across a multi-protocol message bus.

An SOA provides an integration architect with a broad abstract view of applications and integration components to be dealt with as high-level services. Service components in an ESB expose coarse-grained, message-driven interfaces for the purpose of sharing data between applications, both synchronously and asynchronously. In an ESB, applications and event-driven services are connected through the bus as abstract endpoints. These abstract endpoints are tied together in a loosely coupled SOA, which allows them to operate independently from one another. An integration architect uses an ESB to tie together assemblies of abstract endpoints that form composite business processes, or process flows (see Figure 1).

What the endpoints actually represent can be very diverse. For example, an endpoint may represent a discrete operation, like a specialized service for calculating sales tax. The underlying implementation of the endpoint could represent a local binding to an application adaptor, or a callout to an external Web service. The applications and services can be physically located anywhere that is accessible by the bus.

Itinerary-Based Routing

In an ESB, data is passed between endpoints using messages. The coordination of the message passing is done using an ESB concept known as *itinerary-based routing*.

Introducing

SOAPscope 3.0 **Debug**

4 Ways to Know Your Web Services

Whether you are learning how a Web service works, or troubleshooting a tough problem, you need the help of a "smart" tool. SOAPscope lets you dig deeper, faster.

- 1. Try It** Solve problems by testing your Web service with different inputs without writing any code.
- 2. See It** View WSDL and SOAP to understand what's happening. Capture from any toolkit, and see just the right detail for the task at hand.
- 3. Diff It** Compare a problem message or WSDL with a similar, working one.
- 4. Check It** When the problem's not obvious, rigorous interactive analysis finds inconsistencies, errors, and interoperability problems.

Look What's **New** in 3.0

- Integrates with Microsoft® Visual Studio® .NET and Eclipse
- Graph Message Statistics
- Interactive Message Analysis
- Interoperability Testing System
- SSL Support
- HTTP Authentication Support
- HTTP Compression Support
- Support for multi-byte encoding
- Best usability of any Web Services tool

The most comprehensive Web services diagnostic system available, and still **only \$99!**

Try it online at XMETHODS.net or download a trial at Mindreef.com

"SOAPScope 3.0 is easily the most addictive piece of software I've encountered since Halo. When does the multi-player version come out?"

Don Box
XML Messaging Architect,
Microsoft and co-inventor
of SOAP

© Copyright 2002, Mindreef, Inc. The names of companies and products mentioned herein may be the trademarks of their respective owners. This product uses Hypersonic SQL. This product includes software developed by the Contributors of XMethods and its contributors.

"The Web Services Diagnostic Experts" **Mindreef**

A *message itinerary* is metadata that gets carried with a message that provides a list of forwarding addresses. The itinerary is a set of instructions telling the ESB invocation framework which endpoints the message needs to be delivered to as it travels from endpoint to endpoint across the bus. Itineraries contribute to the distributed nature of the ESB architecture by eliminating the dependency on a centralized routing engine, which could potentially be a single point of failure. They are intended for relatively finite microflows of messages. Simple branching and merging of routing paths can be achieved through integration patterns that take advantage of specialized splitter and aggregator services. More sophisticated process orchestrations are also possible using specialized orchestration engines that can be layered onto the bus as additional services.

Configuration, Not Coding

The mantra of the ESB is “configuration rather than coding.” In an ESB, abstract endpoints, which are accessible through application adapters, message queues, Web services invocations, and a variety of other protocols, are configured through a tool interface rather than coded into applications. It’s not that there’s anything wrong with writing code, but there’s plenty of code to be written elsewhere that doesn’t have to do with hard-wiring interdependencies between applications and services.

With its distributed deployment infrastructure, an ESB can efficiently provide central configuration, deployment, and management of services that are distributed across the extended enterprise. Artifacts that affect the behavior of an integration service, such as an XSLT stylesheet that can be used by a data transformation service, are also configurable in an ESB.

The ESB Service Container

The highly distributed nature, and the ESB mantra of “configuration rather than coding” is largely due to traits of the ESB service container. A service container is the physical manifestation of the abstract endpoint, and provides the implementation of the service interface. A service container is a remote process that can host software components.

A service container is simple and lightweight, but it can have many discrete func-

tions. As shown in Figure 2, service containers take on different roles as they are deployed across an ESB.

In its simplest form, a service container is an operating system process that can be managed by the ESB’s invocation and management framework. A service container provides a number of facilities for the serv-

deployment of integration functionality exactly when and where you need it, and nothing more than what you need.

A service container can host a single service, or can combine multiple services in a single container environment (see Figure 3).

An ESB service is also scalable in a fash-

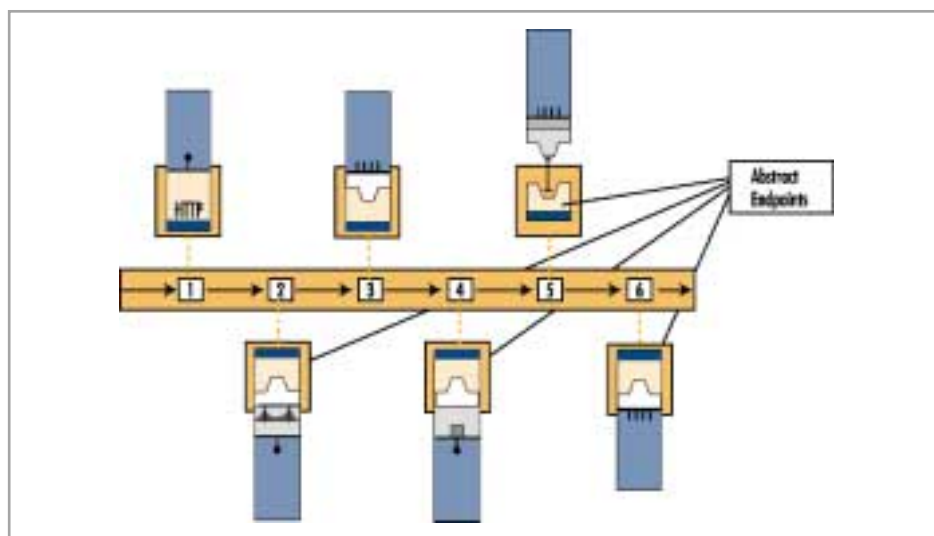


FIGURE 1 All applications and services are viewed as abstract endpoints that are tied together into coordinated process flows.

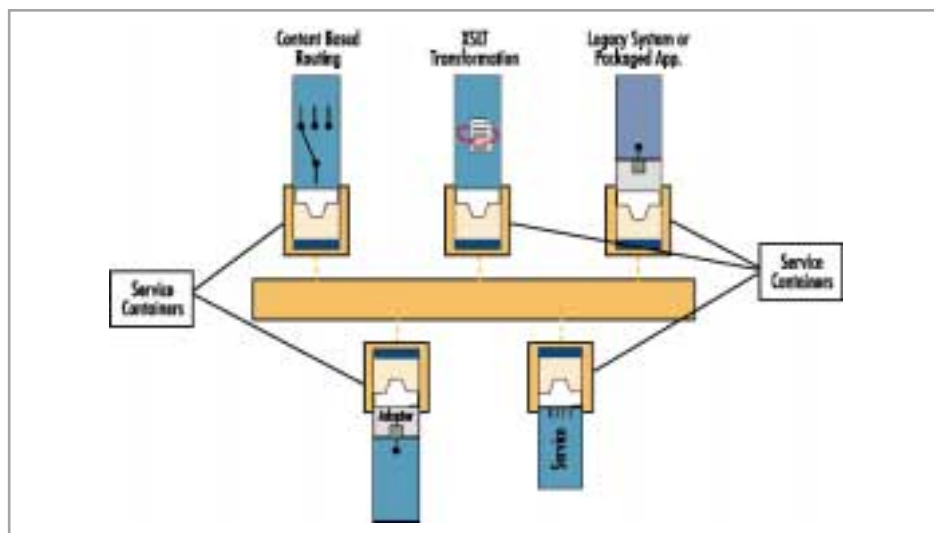


FIGURE 2 ESB service containers are specialized for integration services such as transformation, routing, and application adapters

ice implementation such as event dispatch, thread management, security (encryption, authentication, and access control), and QoS via reliable message delivery. Unlike its distant cousins, the J2EE application server container and the EAI broker, the ESB service container allows the selective

ion that is independent of all other ESB services. A service container may manage multiple instances of a service within a container. Several containers may also be distributed across multiple machines for the purposes of scaling up to handle increased message volume (see Figure 4).

The ESB Service Interface

The ESB container provides the message flow in and out of a service. It also handles a number of facilities, such as service life cycle and itinerary management. As shown in Figure 5, the container manages an entry endpoint and an exit endpoint, which are used by the container

its next destination. The next destination may be a reply to the original sender of the message, or more often may be sent along to the next leg of its journey using a forwarding address. The output message may be the same message that it received. The service may modify the message before sending it to the exit endpoint. Or, in the

case of a content-based routing (CBR) service, the message content will be unchanged, with new forwarding addresses set in the message header.

In more sophisticated cases, one input message can transform into many outputs, each with its own routing information. For example, a splitter service can receive a purchase order document, split it into multiple output messages, and send out the purchase order and its individual line items as separate messages to an inventory or order fulfillment service. The service implementation in this case does not have to be written using traditional coding practices; it can be implemented as a specialized transformation service that applies an XSLT stylesheet to the purchase order document to produce the multiple outputs.

Process Tracking and Error Handling

In addition to a normal exit endpoint to handle the outgoing flow of a message, additional destinations are available to the service for auditing the message and for reporting errors. The tracking endpoint can be utilized to monitor the progression of a message as it travels through a business process. Tracking can be handled at both the individual service level and the business-process level. From the service implementation's point of view, it simply places data into the tracking endpoint or fault endpoint, and the surrounding ESB invocation and management framework takes care of the tracking and error reporting. This approach provides a separation between the implementation of the service and the details the surrounding fault handling. The implementer of a service need only be concerned that it has a place to put such information, whether it is information concerning the successful processing of good data, or the reporting of errors and bad data.

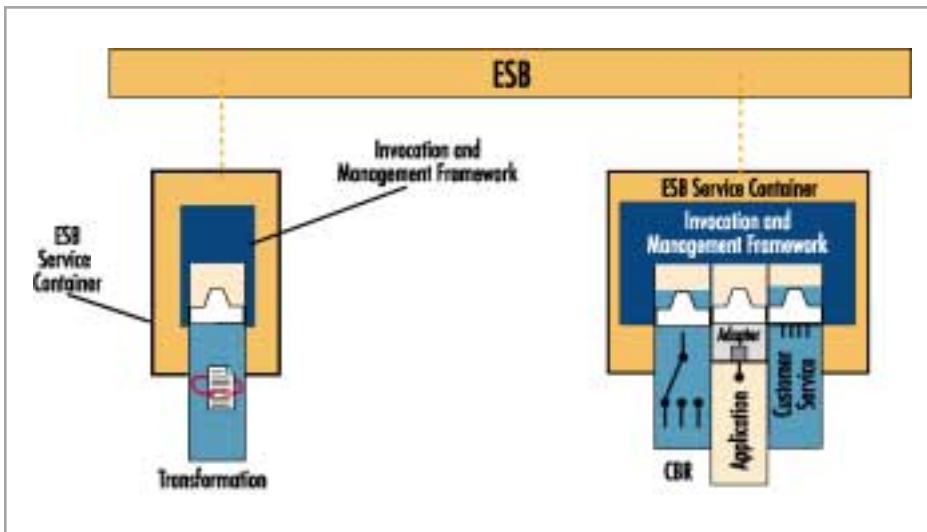


FIGURE 3 The ESB service container allows selective deployment of a single service, or can be combined with other services

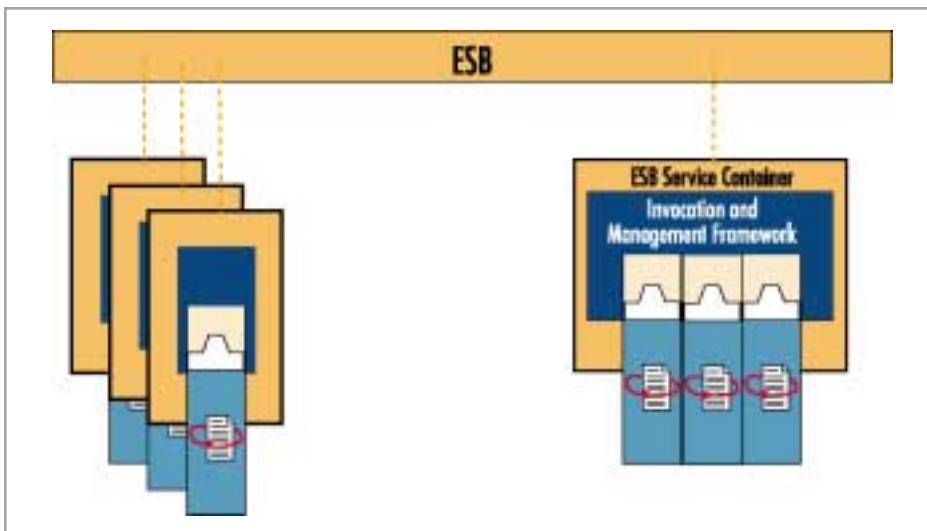


FIGURE 4 Services may be scaled within a container, and several containers may be scaled across multiple machines

to dispatch a message to and from the service.

Messages are received by the service from a configurable entry endpoint. Upon completion of its task, the service implementation simply places its output message in the exit endpoint to be carried to

service may create a completely new message to serve as a "response" to the incoming message and send the new message in the exit endpoint.

What is placed in the exit endpoint depends on the context of the situation and the message being processed. In the

Integration Patterns

One of the many benefits of using itinerary-based routing to coordinate the interactions between discrete integration services is the ease with which integration patterns can be created and reused to solve common integration challenges. A message itinerary can be a powerful and flexible tool for intercepting the path of a message and performing operations on it, thus adding value to the integration environ-

ment. Through configuration and management tools, additional processing steps can be inserted into a business process definition as event-driven services into an XML processing pipeline. The following describes two of the common integration patterns in use today: the “VETO” pattern, and a variation known as the “VETRO” pattern.

The VETO Pattern

VETO is a common integration pattern that stands for Validate, Enrich, Transform, Operate (see Figure 6). The VETO pattern and its variations can ensure that consistent, validated data will be routed throughout the ESB.

“...it is already rapidly changing the way integration is being done across a variety of industries”

Validate

The “Validate” step is usually the first part of any ESB process and can be accomplished in a number of ways. It’s important

that if possible, this step happen independently; this removes the burden of validation from all of the downstream service implementations and promotes reuse. Building validation directly into the first service of a process makes it difficult to insert an additional service in front of it without requiring that the new service also provide its own validation.

An example of validation is to simply verify that an incoming message contains a well-formed XML document and conforms to a particular schema or WSDL document that describes the message. This requires that the service always have available the up-to-date XML schema for a particular message type. The schema and WSDL can be kept in the directory service and managed remotely by the management infrastructure of the ESB. A service may also have scripting associated with it, which can be made available to the service as a configuration parameter

If the target data is not in XML format, or if there is no schema or WSDL available, then a custom service can be used to validate the incoming message.

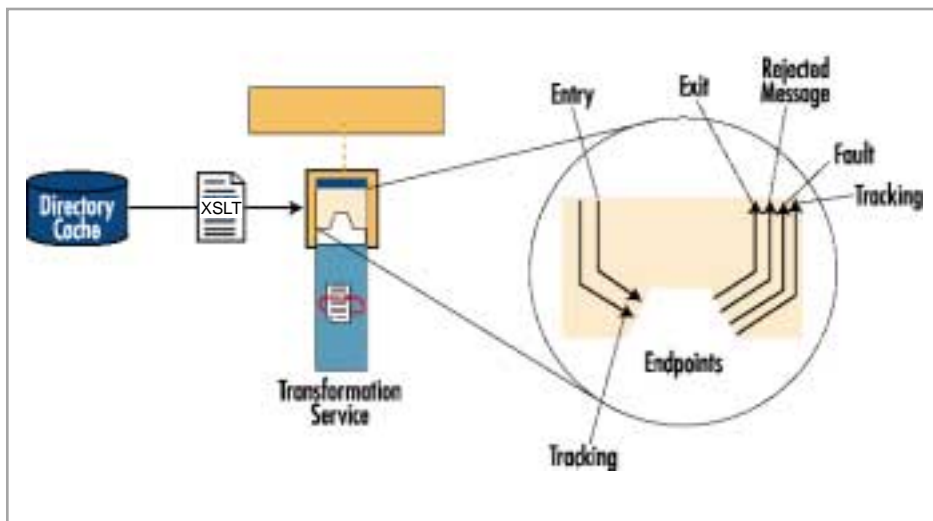


FIGURE 5 | Message dispatch to a service uses the service’s configured entry, exit, fault, and tracking endpoints

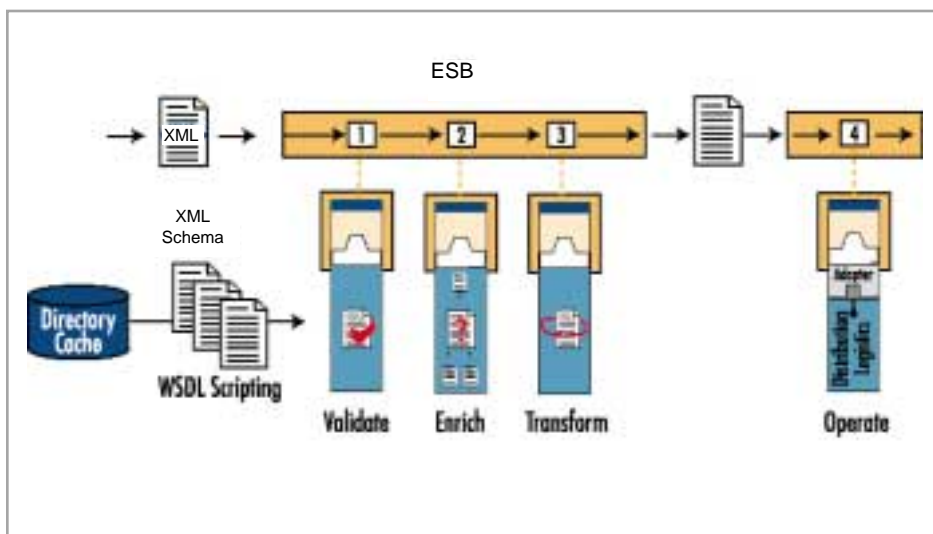


FIGURE 6 | The VETO pattern

Enrich

The “Enrich” step involves adding additional data to a message to make it more meaningful and useful to a target service or application. The Enrich service could be implemented to invoke another service to look up additional data, or it could access a database to get what it needs.

Transform

The “Transform” step converts the message to a target format. This often involves converting the data structure to an internal canonical format, or converting from the canonical format to the target format of the “Operate” step. The target system may have its own built-in validation rules requiring

How to Make Money with Web Services

By Bob Brauer

bob.brauer@strikeiron.com



A new way to work with Web services - build them and they will come?

The benefits of Web services are well understood – lower integration costs, maximum reusability, faster deployment, more automation, easier to work with new partners, and so on. You can build them but that does not necessarily mean that people will use them or that you can make money on them. Why? And more importantly, how?

What drives Web services utilization and return on investment?

Here are some key points to consider:

- Accessibility and ease-of-use
- Billing and accounting
- Subscriptions and trials
- Availability, reliability, and security
- Promotions and pricing

What tools and services do you need to commercialize your Web services?

- Tools to improve understanding and use.
- Commerce capabilities to manage subscriptions, accounting, billing, payments, account management, etc.
- Ability to manage free trials and convert trial users to subscribers.
- Service levels that ensure availability, reliability, and security.
- Knowledge about acceptable pricing structures based on value of the data, process, accessibility, and performance.
- A way to deliver and promote them to the appropriate target audience.

Creating a new revenue channel

Your Web services provide value and you need to be reimbursed for that value. However, before that can happen, you need a distribution channel with the infrastructure to publish and sell your Web services. This channel must take care of issues such as delivery, account set up, billing and collections, marketing, and customer support, to free you of having to make that investment.



The answer is Premium Web Services

Premium Web Services provided by StrikeIron and available through the StrikeIron Web Services Business Network™ (WSBizNet™) are the answer.

Whether you are a Web service provider wanting to commercialize your Web services or a company wanting to find a way to manage your Web services without the infrastructure investment, StrikeIron will do the work for you, and you make money in the process.

StrikeIron Premium Web Services are subscription-based Web services that deliver “live” data and functionality over the Internet via XML using a dynamic SOAP interface. Premium Web Services are supported by the commercial services and easier-to-use tools of the StrikeIron WSBizNet to remove barriers to purchase.

Premium Web Services benefits

- An alternate sales channel to reach more customers for minimal cost.
- Web services hosting and delivery services in a secure environment.
- Pricing, billing, collections, and support.
- Reporting for easier maintenance and tracking.

The StrikeIron Web Services Business Network

The StrikeIron WSBizNet is a subscription-based online network of services and tools for easier and faster

utilization of Web services by business users and developers.

The WSBizNet allows anyone, with minimal experience, to quickly find, understand, and utilize Web services for accessing information, integrating with existing applications, and assembling new applications.

StrikeIron WSBizNet benefits

- Service levels to meet your business requirements.
- Integrated online StrikeIron Web Services Analyzer for faster understanding of Web services.
- Integrated online Knowledge Base makes it easier to understand and work with your Web services.
- Prime positioning in the WSBizNet.
- Increased visibility in one of the largest Web services business directories.

Delivering Web Services to the World!

For Free Trials and more about how to make money with the StrikeIron Web Services Business Network and Premium Web Services, please visit our Web site at www.strikeiron.com.

StrikeIron, Inc.
+1.919.405.7010
sales@strikeiron.com
www.strikeiron.com

©2004 StrikeIron, Inc. All rights reserved. StrikeIron products and services mentioned herein are trademarks or registered trademarks of StrikeIron, Inc. All other brands or product names are the property of their respective holders.

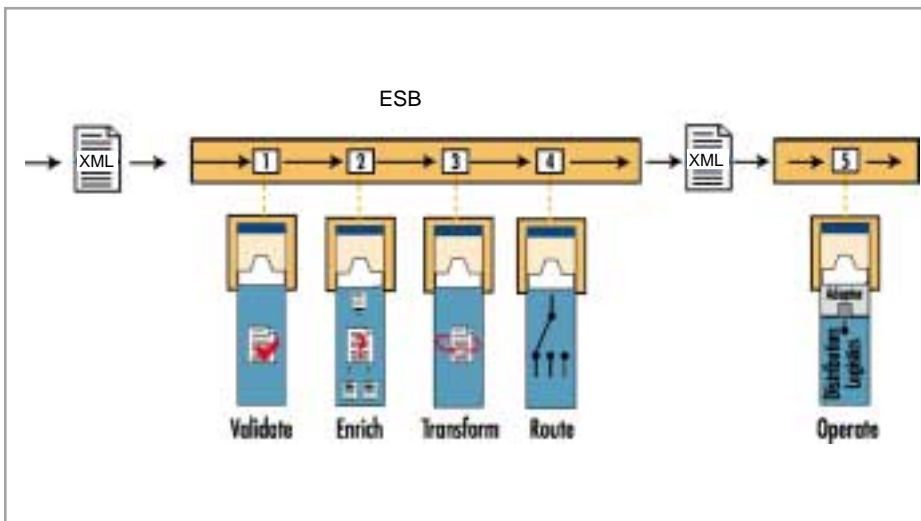


FIGURE 7 The VETRO pattern includes a “Route” step, such as a content-based router

that the transformation step modify the incoming data in order to prevent the target system from rejecting the message. In this sense, the transformation step is also providing pre-validation protection in a separate service that can be separately managed. While this may mean redundant logic in the short term, it provides more flexibility in the long term, because it allows the “Operate” step to focus on business logic.

Operate

The “Operate” step is the invocation of the target service or an interaction with the target application. If the target operation is an enterprise application that requires its own data format, then the previous transformation step converts the message to the target format required by the application.

Variations: The VETRO Pattern

The VETO pattern has many variations.



The mantra of the ESB is
‘configuration rather
than coding’



One such variation is the VETRO pattern, which includes a “Route” step such as a content-based router service (Figure 7).

In some cases the validate, enrich, and transform steps can be accomplished in one service implementation. For example, a CBR service may use a script-based validation directly in the service itself, rather than using a separate service. This may provide some convenience, particularly if the context of validation can't easily be applied to other uses. However, keeping them as separate services further promotes loose coupling and service reuse, and allows the validation to be separately defined and managed. Through the flexibility of configuration and deployment, that choice can be revisited over time without affecting all of the application endpoints that use the pattern. The stages of the VETO pattern can be implemented as separate services that can be configured, reused, and independently swapped out for alternate implementations.

The VETO concept is profoundly simple, yet is at the heart of what an integra-

tion architect does regularly with an ESB. An ESB provides an event-driven SOA for applications in an integration fabric. Regardless of the process routing and orchestration method being used – whether itineraries or the more sophisticated process modeling using an orchestration service – it is the use of integration patterns such as VETO and its variations that provide the overall value and flexibility to the integration fabric.

Summary

I hope that this brief introduction to the ESB and its use of integration patterns has provided you with insight into the internal workings of the ESB, and given

you a sense of how an integration architect can use event-driven components as services to construct reusable integration patterns in an enterprise SOA. The VETO pattern is one of many being used in ESB-based integrations.

I encourage you to learn more about the ESB as a technology concept, for it is already rapidly changing the way integration is being done across a variety of industries. So get reading, and get on the bus! @

About the Author

Dave Chappell is chief technology evangelist for Sonic Software, the leading provider of integration products and services for the real-time enterprise. He is the author of the book *Enterprise Service Bus*. Dave has over 20 years of experience in the software industry covering a broad range of roles including R&D, code-slinger, sales, support and marketing. He has a strong passion for shaping the future of technology, and enjoys sharing his knowledge and experience with others. Dave serves as a technical editor for *Web Services Journal*.

■■■ dchappell@sonicsoftware.com



Everything **OLD** Is **NEW** Again!

Imagine **XML-enabling** your existing **PowerBuilder** and **Visual Basic** applications

QUICK

In hours, without changing a line of code

CHEAP

Inexpensively, using your current skills

SAFE

Safely, preserving all logic and security

NOW YOU CAN

www.active-endpoints.com

Toward SOA with ESB

A real-world view from design to implementation



■ Enterprises have become increasingly sensitive to the need to become sense-and-response-enabled. That is, enterprises perceive a strategic need to be able to both respond faster and more effectively to, and initiate changes in their environment.

On-demand capabilities require extensive flexibility and adaptability, which in turn require a highly integrated, zero-latency enterprise with respect to both IT systems and business and IT integration within and across organizational boundaries. Collectively, these requirements stipulate a new approach to both business and IT design and service-oriented architecture methodologies and technologies to address these issues.

From a design and architectural perspective, SOA, Web services, and associated technologies, such as BPEL, have been accepted as integral components to solve the integration of distributed systems. Not surprisingly, we see the emergence of a new generation of middleware: not integration glue, but core enterprise applications that help realize the architecture, runtime, and tooling requirements of a flexible, adaptive-on-demand enterprise. Specifically, we see the (new generation) enterprise services bus (ESB) as a salient, network-centric integration core to realize SOA.

This article looks at the design and implementation requirements of an ESB in the context of



WRITTEN BY
**BERNHARD
BORGES &**



ED KAHAN

the SOA enablement of a retail client. Due to the scope of this article, a host of important (Web) services management issues are not presented. Pertinent information is available from the authors.

The Business Case: Faster, Cheaper, Better Across All Channels

Our client approached us with the problem of improving various business and operational aspects of their concern. The client is in the notoriously competitive retail business, managing some 600 stores across the U.S., a 24,000-item inventory, 20,000 employees, and \$2 billion annual revenue. Moreover, our client already set significant growth targets, i.e. approximately 100%, across all major

business drivers, such as store presence, per-square-foot store revenue, and profitability. These goals were to be met over a relatively short period of time (see Figure 1).

A client-specific analysis based on IBM's retail industry business model, framed the business analysis for the desired on-demand transformation. A key finding associated with our client's strategic plans concerned the improved flexibility

and adaptability around the design, implementation, and management of business processes, not only within its organizational boundaries but also across the entire supply chain (see Figure 2).

After assessing both business and IT operations in the context of the overall strategic goals, it became apparent that a fundamental change was required. Specifically, the incumbent IT infrastructure was too distanced from the business operations and IT itself needed to be much more integrated to be able to respond in a timely and effective manner. Also, incumbent, complex "bundled" business processes needed to be unbundled to accommodate the desired flexibility and an adaptive, cost-effective integrated supply chain. In combination, these requirements supported an ESB-enabled SOA. Figure 3 summarizes the key findings, mapping the client's strategy into services-driven business priorities.

After reviewing several alternative solution approaches, the client and IBM team jointly decided to utilize an SOA and to embark on an ESB-enabled, Web services based strategy. The SOA-based design, architecture, and implementation roadmap were developed around an ESB (see Figure 4).

It is critical to appreciate the interrelatedness of business and IT analysis when approaching and designing a SOA-based solution. SOA purports to close the business-IT gap; hence, both business and IT requirements need to be analyzed in a recursive manner. This two-pronged approach aligns the solution to the salient problems and directly impacts on the identification, (re-)definition, and implementation of business

services. This, in turn, affects legacy, ISV and custom application development, especially in light of an ESB-based solution.

It also needs to be recognized that the SOA design, architecture, and roadmap were put in place with an enterprise point of view. Yet, the client's strategic priorities emphasized the timely transformation of particular business, and associated IT, functions, thereby taking advantage of SOA's intrinsic incrementalism.

SOA and ESB Background

Fundamentally, an SOA is a framework for describing, publishing, and discovering services, where services are viewed in the context of business process descriptions. More specifically, SOA constitutes a logical construct that is realized as a new logical services layer and subsequently enabled by the ESB (see Figure 5).

In the context of an SOA, processes orchestrate one or more services to fulfill the appropriate business functions and the services, in turn, are realized through components.

Web services are instrumental in enabling SOAs in an interoperable manner. XML, coupled with a set of Web services – e.g., WS-* specifications – allow the implementation of broadly sharable, loosely coupled, and reusable services within an SOA framework. The orchestration, choreography, and management of services fulfilling business processes, especially in the context of an enterprise solution, require the appropriate nervous system, the ESB has been deemed necessary to perform that function.

The IBM ESB is J2EE-based and an essential element of an SOA realization. It supports a highly distributed services model comprised of service providers and service requesters. Moreover, it enables standards-based integration between loosely coupled applications and services within and across SOAs – an important consideration as many, if not most, enterprises adopt SOA in an incremental fashion and with an eye toward significant reuse of existing systems.

Overall, an ESB purports to enable application integration across different platforms, programming models, and messaging standards, and underlies the more extensive business integration models of business process and partner collaboration. The essential infrastructure services that an ESB needs to provide include transport, routing, quality of service, mediation, event management, de/composition, system management, security, data protocols, and policy management (see Figure 6).

An important aspect of an ESB concerns its support of Web services and associated technolo-

gies, such as BPEL. The Web services support in the IBM ESB uses relevant industry interoperability standards and supports the WS-I profile(s). Some of the published but not yet standardized WS-* specifications are supported as well. The actual access to the ESB from the Java platform uses the relevant standards, such as JMS, JAX-RPC, Web Services for J2EE (JSR 109), and JAXR.

From Design to Implementation

SOA design and implementation is an intrinsically simple endeavor – if it was occurring in a legacy-free vacuum. Most enterprises, however, operate in a context marked by incumbent business and (IT) systems. Moreover, most enterprises have neither the time nor the appetite to develop custom applications, or components, unless necessary and instead prefer to take advantage of packaged ISV solutions. In combination, these realities can amount to a significant set of constraints toward a pure play SOA, making it even more important upfront to put in place a solid SOA strategy.

A concept regarding the use of an ESB is that all services should expose a common data model. Incumbent systems, e.g., legacy or ISV applications, generally have different representations of data and hence a business process needs to choreograph the data communication across disparate systems, requiring costly transformations.

Our ESB view is to define and drive a common, standards-based data model. This enables the sustainable, long-term management of the enterprise's information model, sharable across disparate systems, and alleviates extensive, max.n², transformations. In the case of our retail client, the ARTS (Association for Retail Technology Standards) data model and IXRetail (International XML Retail) XML Schemas are used as the common data model.

Overall, a successful ESB-based SOA implementation requires a considerable degree of analysis, taking into consideration both business and IT systems. After all, an SOA-enabling ESB is an *a priori* enterprise core application facilitating the integration of highly distributed, loosely coupled services, which is vastly different from yesterday's ad hoc integration glue approach.

Requirements Analysis

In the case of our client project, as is true in most client scenarios, the incumbent environment, especially an IT environment, functioned as a baseline. A business (process) and IT environment assessment is a prerequisite for an ESB-based SOA transformation (see Figure 7).

These assessments and analyses are a nontriv-



FIGURE 1 Forces driving the retail industry



FIGURE 2 Overview strategy map



FIGURE 3 Overview strategy realization map

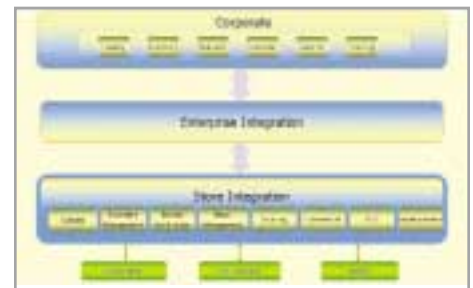


FIGURE 4 High-level ESB-based SOA approach for retail client

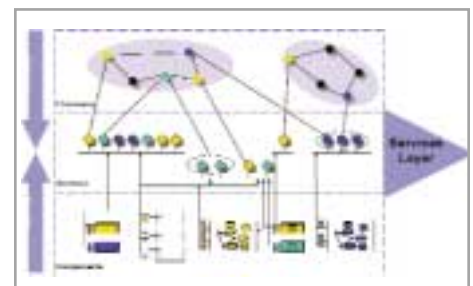


FIGURE 5 Layer representation of SOA for ESB enablement

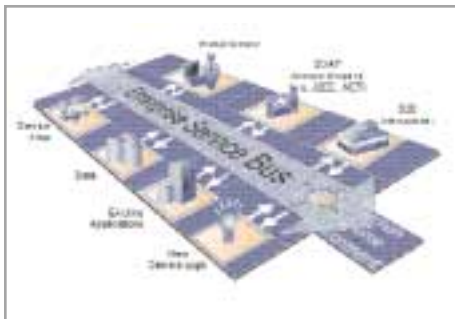


FIGURE 6 Logical IBM ESB representation

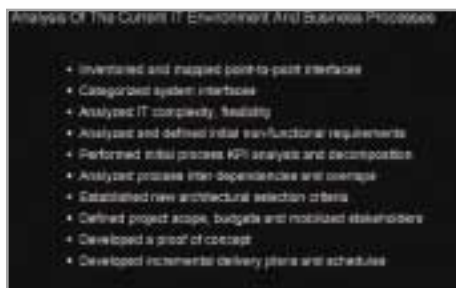


FIGURE 7 Partial business and IT environment assessment framework



FIGURE 8 IBM odOE reference model

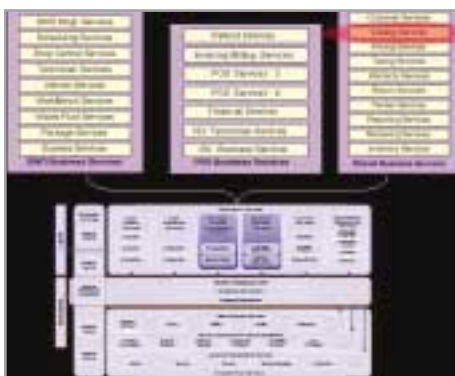


FIGURE 9 SOA-based services descriptors and IBM ESB mapping



FIGURE 10 Logical representation of ESB-based services implementation

ial, fact-finding mission and a critical prerequisite to an ESB-based on-demand transformation.

Since we are moving toward a highly distributed system “constrained” by existing systems and the well-founded desire of maximum [application] reuse, we also found it quite practical to manage the many “moving pieces” via a comprehensive, functional proof-of-concept (PoC) implementation. Hence, key service interface definitions and implementations were one of the first outcomes of this assessment and tooling quickly became an integral part of the early project phases, allowing maximum reuse for subsequent phases.

Operating Environment Reference Model

Another important aspect of the assessment and design phases concerns the choice of an operating environment reference model. Over the course of the client project, it was decided to utilize IBM’s on-demand Operating Environment (odOE). The IBM odOE is based on the concepts of an SOA and utilizes an ESB as its integration core (see Figure 8). The odOE’s inherent support for SOA drives the views of every application or resource as a service implementing a specific, identifiable set of [business] functions. In addition to the business functions it may also implement management interfaces to participate in the broader configuration, operation, and monitoring of the environment, providing a complete, end-to-end life-cycle model.

Determination of Reusable Services

An inherent aspect of the SOA strategy governing implementation concerns the identification of suitable business services. Based on the analysis and reference model outlined above, three distinct sets of business services were identified and formalized against the IBM odOE (see Figure 9): service work order (SWO), point of sales (POS), and shared business services.

Although it goes beyond the scope of this article to review each of the service categories in detail, we want to review the Catalog Services, part of the Shared Business Services portfolio, to illustrate the deterministic attributes of what actually constitutes a suitable business.

The purpose of the Catalog Service is to make available the complete product catalog in a timely and accurate fashion to both human and machine, i.e., A2A interfaces. The catalog is used by many of our client’s applications, e.g., retail POS, service work order, etc., and constitutes a strategic competence of the integrated supply chain. Moreover, the catalog is governed by a shared set of functionalities, such as adding, removing, pricing, etc. of catalog items. Aside

from the multiple access, catalog service must be able to be composed with other services in order to enable complete business processes.

In this SOA model, the Catalog Service connects through the IBM ESB via many external systems to form a “virtual catalog.” The backend systems supplying the components of the Catalog Service have many, often idiosyncratic, protocols, access mechanisms, and data formats, which needed to be abstracted. For a logical representation of the ESB-based services implementation see Figure 10.

Client-Specific IBM ESB Implementation

A variety of technologies can be used to support the implementation and exposure of Web services onto the ESB, such as IBM MQ (publish/subscribe), adapters (JCA, IBM WBI Adapters, etc.), data transformers (XSLT), and database access (JDBC). This ESB construct enables the legacy system to be exposed in a standards-based manner and provides the infrastructure needed to choreograph services into business processes. A selector framework allows for the service client, or consumer, to be spared from worrying about the location of the service implementation.

The ensuing integration layer represents a logical ESB (see Figure 11), which spans both the client’s retail stores and enterprise IT systems. A key feature of SOA is that the physical location of each service is hidden from the applications and can be changed at any time, without application changes. Our implementation facilitates this critical SOA feature.

The services that it provides are delivered to calling applications as proxy Java classes that are included in the application’s runtime environment. The application makes simple Java calls to these proxy objects, exchanging input and output parameters in the form of simple data-transfer objects. The services themselves are implemented through a series of adapters that connect to the requisite legacy, or external system or data source, that the calling applications need to access.

Our client’s environment is comprised of multiple legacy applications that run in the retail stores today, e.g., inventory, commercial sales, etc. While the client has the long-term goal of replacing or porting these systems, in the near future these applications need to remain in place and be wrapped as services to simplify their eventual conversion. The rationale for retaining these systems is, in part, based on our client’s time-to-market goals, which could not be met if these systems had to be revamped.

Talk about possibilities!

ScanSoft®

UNISYS


**Brooktrout
Technology®**
Your Hook into the New Network®

CONVERGYS

 **aculab**

Gold Sponsors

Silver Sponsors

Bronze Sponsors



Media Sponsors

- >> Discover the Latest Innovations in Speech Technologies
- >> Improve Customer Service
- >> Increase ROI
- >> Learn Industry Leading Best Practices
- >> Network with Thousands of Your Peers

ACT NOW for Biggest Savings and Free EXPO Pass with priority code WSJ1.

Registration information:

Chris Nolan

1-859-278-2223 or
Toll free 1-877-993-9767
chris@amcommexpos.com

**10TH
YEAR!**

SpeechTEK|2004
The Voice Solutions Showcase

September 13-16, 2004 >> EXPO 14-15, 2004 >> New York Marriott Marquis

www.speechtek.com for more information

IBM's WBI Server Message Broker is deployed on the client's enterprise systems and facilitates the transformation of all data that passes between the stores and the enterprise legacy systems, such as pricing. In order to provide the necessary system resilience, critical data is passed between the stores and enterprise IT systems, using IBM WebSphere MQ to provide assured delivery.

Figure 12 illustrates in more detail the n-tier enterprise ESB solution, where service requests are processed as follows:

- Requests are initiated by client applications.
- A selector framework routes the request to the appropriate service handler.
- The service handler:
 - Processes the request, invoking legacy applications and third-party systems where appropriate,
 - Performs any data re-mapping necessary to translate between the service request and the legacy and third-party application formats, and
 - Maps the responses from the legacy and third-party applications to the appropriate format for the service response.
- Synchronous responses are returned to the calling application.
- In the event of problems, a simple consistent set of error messages may be returned to the calling application.

All client access into the integration layer is through WSDL-described Web service requests and all services implemented on the integration layer are exposed as Web services. Moreover, legacy and ISV systems are wrapped with JCA and front-ended with stateless session beans, which themselves are exposed as Web services. Proxies are generated to provide the client applications with a simple access point to these services.

Business process choreography is implemented to orchestrate the execution of services based on exposed business rules. Activities within the process choreography are implemented as stateless session beans. Common infrastructure subsystems provide common services to business processes within the integration layer, which include logging and error handling, systems management, security, and configuration management.

Specific considerations must be given to ISV applications. Most ISV applications to date lack the degree of componentization required to expose, and possibly consume, services to truly instantiate an SOA. Although the IBM ESB has the ability to integrate across heterogeneous IT systems, thereby mitigating some of these constraints, the ability to easily implement end-to-end business process capabilities is hampered.

In our retail case, many of the identified shared services already existed in some (functional) form in an ISV application. A fundamental rule of an ESB is to expose these services via Web services on the bus using a common data model. Therefore, ISV packages are exposed via a new Web service based on the common data model. The implementation of the service accesses the ISV package in its native manner and transforms the output to the common data model.

Increasingly, we are finding that most third-party ISVs are very interested and highly motivated in componentizing their applications, often utilizing the IBM's odOE reference model. This architectural transition invariably improves the ISV's ability to integrate with their customers' legacy systems while simultaneously reducing their maintenance requirements. Yet today's packaged applications don't foster the required degree of componentization and a certain degree of art, i.e., project-based, is required to compensate for this shortcoming.

Summary

This article provided a detailed overview of the design, architecture, and implementation requirements to implement an ESB-based SOA for a retail client. The overall design and architecture has resulted in a demonstrable implementation approach that exploits SOA to transform existing IT systems to satisfy imminent business needs. The ensuing solution creates much of the flexibility and adaptability required to support and enable business strategy by effectively bridging the business-IT gap. Moreover, the IBM ESB-enabled SOA implementation described above has led to a significant and sustainable reduction in systems and process complexity by representing services via standard interface descriptions.

Acknowledgements

We want to thank the project team for their outstanding contribution to generate and realize innovation in what constitutes largely uncharted territory; also, Jason Weisser, VP, IBM SWG EI; George Galambos, IBM Fellow; Manish Modh, Senior Solutions Engineer, IBM SWG EI; and Marc Fiamante, Executive IT Architect, IBM SWG EI, for their invaluable comments and support.

References

- Channabasavaiah, Kishore, et al (2003). "Migrating to a service-oriented architecture, Part 1," IBM developerWorks, www-106.ibm.com/developerworks/webservices/library/ws-migratesoa and www-106.ibm.com/developerworks/webservices/library/ws-esbscen



FIGURE 10 | Logical representation of IBM ESB for retail client

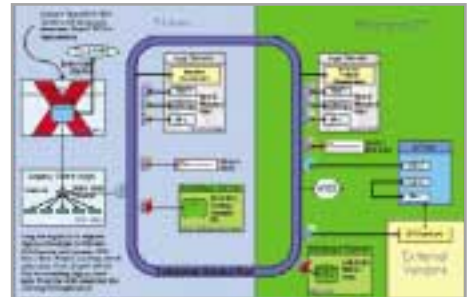


FIGURE 11 | IBM ESB Integration Layer Component Model

[erworks/webservices/library/ws-migratesoa2](http://developerworks/webservices/library/ws-migratesoa2)

- Robinson, Rick. (2004). "Understand Enterprise Service Bus Scenarios and Solutions in Service-Oriented Architecture, Parts 1, 2, and 3," IBM developerWorks. www-106.ibm.com/developerworks/webservices/library/ws-esbscen, www-106.ibm.com/developerworks/webservices/library/ws-esbscen2.html, and www-106.ibm.com/developerworks/webservices/library/ws-esbscen3
- Endrei, Mark, et al. (2004). "Patterns: Service-Oriented Architecture and Web Services." IBM Redbook, SG24-6303-00

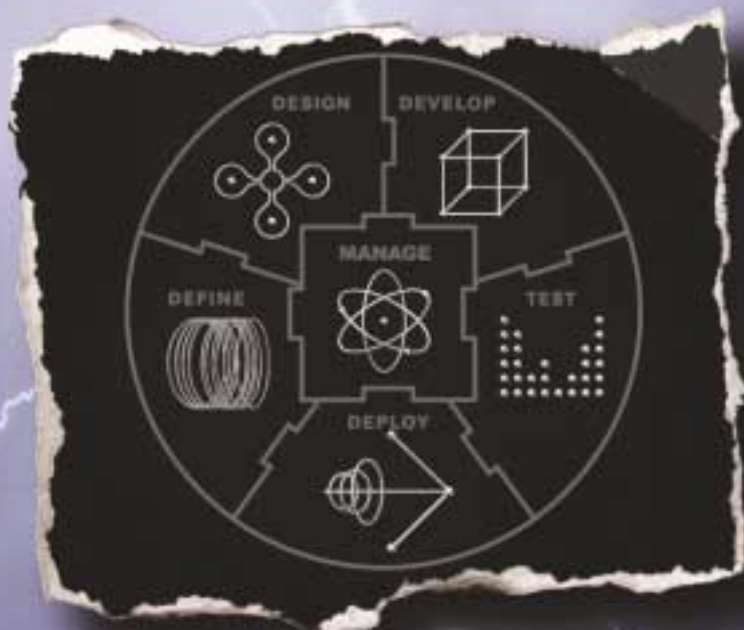
About the Authors

Bernhard Borges is a technical executive and solutions architect in the IBM Software Group's Enterprise Integration group. He specializes in distributed, open, interoperable, and portable systems, especially in the context of SOA, ESB, and Web services. Bernhard is an IBM Distinguished Engineer and a member of IBM's Academy of Technology. He is also a member of the IBM Software Architecture Board and several workgroups, IBM's SOA and Web Services Technical Council, and an extended member of IBM's BCS SOA-Web Services Center of Excellence.

■■■ bborges@us.ibm.com

Ed Kahan is an executive IT architect with IBM Software Group, Enterprise Integration, and managing solution architect of its North American Architecture group. He is an IBM Distinguished Engineer, Certified Consultant, and member of the IBM Academy of Technology. His current responsibilities include design and development of Web services, service-oriented architectures, and enterprise integration products and solutions for IBM's clients. He has extensive knowledge of design and implementation of complex systems in several industries. Ed holds a degree in mechanical engineering.

■■■ edkahan@us.ibm.com



Unleash the Power of the Application Lifecycle at the 2004 Borland Conference

The Borland Conference is a premier event for technical education, focusing on all the technologies impacting software development. With more than 200 technical sessions, you will see how you can facilitate teamwork, enhance productivity, improve quality, reduce costs, cut maintenance time, and accelerate business flexibility and success. Learn how the entire development team can create and deploy better software, faster – with the integrated Borland suite of products for the analyst, architect, developer, tester, deployment group, and manager.

Special discount of up to 50% on select Borland products • Exhibit hall • Free conference proceedings CD • Product Solution tracks covering all Borland products, including JBuilder, Delphi, Together, StarTeam, CaliberRM, C#Builder, C++BuilderX, Optimizely, ServerTrace, Borland Enterprise Server, Janeva, and InterBase • Interest Area tracks, including ALM, Methods, and Processes; Architecture, Models, and Patterns; Microsoft .NET Framework; J2EE; SOA; emerging technologies; and more!

September 11-15, 2004 • San Jose, California

For complete conference details and session information: connect.borland.com/barcon04

Made in Borland® Copyright © 2004 Borland Software Corporation. All rights reserved. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All Borland brand and product names are trademarks or registered trademarks of Borland Software Corporation in the United States and other countries. • 22127.4

Borland
Excellence Endures™

DreamFactory Suite from DreamFactory

A user experience for service-oriented architectures



■ The DreamFactory suite is a mature development and execution platform for crafting rich, event-driven interfaces.

DreamFactory picks up the UI development “gauntlet” with a thoroughly thought out metaphor, framework, and tools whose sophistication matches, and usually exceeds, that of the other application components while remaining straightforward to develop in. The resulting lightweight applications run in Internet Explorer, Mozilla, and Netscape (on the Windows and Apple Macintosh platforms, with Pocket PC, Palm OS, and Linux versions in development).

DreamFactory Flavors

The DreamFactory (DF) suite consists of three components: Runtime, Professional, and Enterprise. The DF Runtime and Professional each provide the development and execution environments. All the tools and features available in DF Runtime are available in DF Professional and vice-versa.

Rather than being locked into the browser environment (and its limitations), DF chose to find a standard, portable way to provide rich UI functionality on a variety of platforms, and settled on implementing DF as a MIME type. Within the browser (and most other MIME-aware applications, such as Word), a DF Runtime can be rendered in full-screen mode or embedded in a Web page (even as multiple instances). DF Professional, on the other hand, is a stand-alone application. DF Enterprise supports the installation and deployment of DF media across networks.



WRITTEN BY
AHMED SAKO

Developing Applications

In DF Professional, projects are opened from a Web server or file system. Runtime can do so as well and, using a key sequence, can open the IDE while an application executes (controlled via security settings), with modifications immediately reflected in the running application.

The DF IDE includes the standard development tools: UI Builder, GUI Objects (Nodes) Palettes, XML/SOAP/WSDL viewers, and wizards, script editors, debugger, etc. While these tools are not uncommon, the completeness of the assembled set is surprising, along with the depth of the functionality available and the ease of use. For example, rather than supporting a single scripting language, DF supports three (JavaScript, VBScript, and DFScript)! Better yet, developers can change the default scripting language on the fly, resulting in the scripting languages being seamlessly converted from one to the other. As another example, variable, method, and object names can be auto-corrected while typing.

The project-structuring metaphor in DF is the XML Tree, with a DF project a hierarchical collection of objects called nodes. The relationships and operations (traversals, tree and node management, introspection), supported on the DF Node Tree are similar to the operations that can be performed on an XML tree.

Dynamic behavior is implemented using scripts. Scripts can be associated with nodes to respond to events and provide utility

functions. In addition, every project has a special script called “project” that receives most system messages and parcels them out to the appropriate node. Finally, the DF libraries also contain scripts that can be invoked.

The hierarchical structure of the nodes not only determines the visual organization of the screen and the logical organization of the project, it also establishes the default message passing hierarchy when events are triggered. The messages flow via scripts, from node to node along the tree, up to the project script and then to the libraries, searching for a handler.

A developer can control the delegation of, handle, trap, hook, reroute, and trigger events to and from other controls (even in other projects, depending on the security settings), allowing for fine control over the application behavior and messaging flows. The default behavior of the nodes is generated in the node script when the node added to the project. This gives the developer even more control over the behavior by permitting him or her to modify the scripts controlling the node’s implicit behavior. Similarly, errors and system responses to abnormal conditions (errors, warnings, exceptions, etc.) can also be precisely controlled (from the code).

Rendering nodes is platform independent with DF controlling the rendering and not the browser (as DF is a MIME type). This



Company Info

DreamFactory Software, Inc
18640 Overlook Road
Los Gatos, CA 9503
www.dreamfactory.com
Phone: 1-888-399-DREAM (3732)
Fax: 1-408-351-9005
SALES: sales@dreamfactory.com

The DreamFactory Runtime and DreamFactory Professional editions are available at no cost. The DreamFactory Enterprise, providing application maintenance and network deployment facilities, can be evaluated for 30 days, after which a license agreement is required.



FIGURE 1 The DreamFactory Tool Palette (expanded to show all the node types)



FIGURE 2 WSDL wizard example: Obtaining quotes for a given symbol

allows for a degree of isolation from the variations and issues related to the underlying platform (browser, O/S, etc.) which DF abstracts and resolves to prevent issues from affecting the users. As a result, applications developed on one platform look identical on the other supported platforms. To achieve this result, the nodes (or any other DF component like scripts, etc.) have no direct interaction with the native environment and DF marshals the requests. Yet, the platform feels very responsive (even under loads), without the delay experienced, between action and response in some purely scripted environments.

Even system resources such as fonts, cursors, etc., are abstracted. For a system-specific resource to become available for development in DF, it is first captured and compressed by DF into a neutral format (for fonts this format is the "DreamFactory Font") that is native to the DF platform, enabling the developer to access the newly created resource.

Every node-type superclass is associated with a plugin that provides its implementation. The plugins, written in C, leverage a cross-platform library provided by DF (making your developed code instantly portable, across OSs and browsers). The runtime and nodes, are lightweight (400K one time runtime environment download, with additional nodes downloaded as needed, then cached).

While most of the node types used to create nodes (e.g., Document Window, XML Palette, etc.) and the features they provide (advanced media types, fonts, drag/drop, etc.) are available in other environments, the large number of node types and the quality of each one is surprising (see Figure 1).

Using XML, WSDL, SOAP, and Web Services

XML, WSDL, SOAP, and Web services can be accessed via the XML nodes (used for accessing XML and SOAP/Web services sources), the API, and variables (XML only). The XML nodes provide a uniform invocation mechanism for different request types. Wizards are available to read in WSDLs (see Figure 2), generate the appropriate proxies and bind the data to nodes. The XML data returned by queries can be persisted, updated, assigned to a variable or unloaded as needed.

Here are some enhancements that DF is planning for their next release:

- Enhancements to the WSDL wizards to map multiple inputs and outputs to/from the GUI directly, plus enhanced support for communication semantics
- New "Document Exchange" wizard that will read in WSDL and generate whole UIs

Objects and associative arrays are also implemented as XML, and XML can be assigned/and extracted directly to/from them. An API is avail-

able to manipulate the XML (e.g., to search, select, delete, cut, copy, paste, validate, transform to and from text/binary, etc., on single and groups of elements). XPATH expressions can also be used via the XML Palette Node along with the object/structure/Associative Array notation.

Project Security and Application Deployment

DF Runtime normally operates in a strict security sandbox using a specialized virtual machine (VM) similar to the Java VM (JVM).

DF provides security controls through the VM security policies and the project security (e.g., disabling the key sequence to access to the development environment at runtime).


Summary

The DreamFactory Suite from DreamFactory software facilitates the creation of enhanced user interfaces for Web-based applications.

The abundance of well fleshed out components and tools married to an intuitive, easily relatable application model, coupled with a solid implementation, makes the platform shine.

The platform has a "mature" feel, with the richness of the environment making the development experience akin to developing a native graphical application where most of the heavy lifting has been done for you and you can focus on the end-user features.

“ applications developed on one platform look identical on the other supported platforms ”

No wonder then, that some of the leading Web services providers (SalesForce.com, Grand Central) are using DreamFactory. 

About the Author

Ahmed Sako is CTO at an agency-only securities trading firm. His areas of expertise include distributed systems, transaction processing, application architecture and development, system analysis, and Web design.

■■■ asako@rblt.com

Where Web Services Meet Mobile Devices

The challenges to be met

■ Let's face it, we're going mobile. You only need to consider how you communicate these days to understand that. I, for one, find that my Blackberry is becoming my e-mail terminal of choice, as well as my best source of information via the WAP-enabled Web browser built-in. And oh yes, it's a phone. Others are finding that their mobile phones are their single point of contact, both voice and data, and this trend will only continue as we learn to cut the wires and as wireless networks become more pervasive and much faster.

The use of mobile devices as a personal information delivery platform is commonplace today. However, as mobile devices and standards progress we've begun to consider our cell phones as enterprise application delivery platforms as well. And, to this end, have also developed standards for Web services delivery on mobile devices.

The fit is clear. Mobile devices are basically becoming our new client platforms, and thus need to support most of the new computing paradigms, including SOA. Moreover, the demand is there as corporate America learns that they don't need to be tied to their computers in order to do business. Finally, we seem to have both the coverage and bandwidth needed to support this type of infrastructure. Indeed, I foresee a future where the number of applications delivered over wireless networks exceeds those delivered via more traditional means.

Web Services in Motion

The cell phone companies have not been standing by watching this trend emerge; they have begun to lead the way. At JavaOne, Nokia announced that they are



WRITTEN BY
**DAVID S.
LINTHICUM**

building a service-oriented architecture framework on smart mobile phones that could make Web services more of a household word since it will run on our phones.

This is a bit different from more traditional SOAP interfaces between desktop/server applications and telecomm-hosted servers, such as MMS message servers, location servers (see below), and presence servers. We know those approaches work, but the core shift here is a move from simple information sharing to full-blown delivery of application services/behavior down to mobile devices. This is not only new, but useful. Nokia is proposing the integration of mobile clients directly into SOAs using asynchronous Web services.

Truth be told, mixing Web services with mobile devices is not a new concept. For instance, the .NET Compact Framework has been supporting Web services for a while now. On the Java side of things, kSOAP is a J2ME-based SOAP parser and JSR 172 provides a standard set of XML and SOAP APIs on J2ME devices.

The issue with the existing approaches is the assumption that the device interacts with one service at a time, using synchronous mechanisms (much like traditional RPCs). As you may know, synchronous and mobile devices are not terms that go well together, so many developers opted for more customized types of approaches rather than leveraging these standards.

Web services, in the context of mobile computing, is all about the notion of devices that can move in and out of service areas, and at the same time find and leverage Web services as needed and with the right validation. This should be a bidirectional mechanism where mobile devices can both consume and provide services; in essence, the mobile device becomes a peer.

Other Standard

There is never just one standard, is there? In the mobile Web services world, you also have to consider the Open Mobile Alliance (OMA) and their OMA Web Services 1.0 specification. On June 15, 2004, the OMA announced the public availability of new and "up-leveled" mobile specifications that are built by OMA member companies and define how wireless data services may be shared across operators, terminals and geographies. There are 350 member companies in the OMA.

This standard, like the Nokia announcement, defines how Web services may be exposed, discovered, and consumed using standard Web services technologies. The OMA Mobile Locations Protocol defines a

“ mixing Web services with mobile devices is not a new concept ”

“

Performance is a key issue,
as well as connectivity

”

core set of operations that the location server is able to perform, and the OMA Online Certificate Status Protocol defines a protocol for trusted certificate validation.

The goal of this specification is to provide guidelines for Web services implementations within the OMA architecture, and how to leverage SOA in the world of mobile devices. Moreover, it ensures interoperability across servers and terminals supporting Web services protocols.

This specification defines security threats and services, privacy and identity management, messaging and transaction, as well as policy and system management. It also defines the ability to deploy OMA Web services such that the processing associated with common, cross-enabler capabilities may be factored out of individual service enablers and delegated to other entities. Finally, the specification defines the ability to automate the discovery and use of policy information that governs Web services interactions.

What's Next?

There are still many technical challenges to consider, the most relevant being access control. In order for Web services to be effective on a mobile device a single sign-on scheme is needed to track all of the devices and services and determine who has permission

to access what services. The OMA defines this type of mechanism, but this goes well beyond simple validation. You can bet that it is going to be a bit more complex than SOAP can handle, and you'll need new standard single sign-on services, such as those from the Liberty Alliance, perhaps built for mobile computing.

You also need to consider new runtime paradigms as well. Performance is a key issue, as well as connectivity, when you consider the lack of bandwidth that most wireless networks suffer from, as well as dead areas. We can't rely on fast networks to cover up inefficient programming techniques; the communication mechanisms need to be asynchronous and understand how to work around normal communications interruptions that you find in mobile network. It would be a hard pill to swallow to lose your sales order when you travel into a dead zone. ☺

About the Author

Dave Linthicum is the CTO of Grand Central Communications (www.grandcentral.com) and has held key technology management roles with a number of organizations including CTO of both Mercator and SAGA Software. David has authored or co-authored ten books, including the groundbreaking and best-selling *Enterprise Application Integration* released in 1998. His latest book, *Next Generation Application Integration, From Simple Information to Web Services* was just released.

■■■ e-mail: linthicum@att.net

IN THE NEXT ISSUE OF WSJ...

Focus: Standards

Automating MISMO Processes

Vertical industry standards can accomplish far more than a common base of operation. With communications standards that are relatively mature compared to other sectors, the mortgage industry is one where companies use Web services to do more than automate simple message exchange, and speed and streamline their operations.

The Case for WS-Notification

Why do we need WS-Notification and its related specifications? This article starts with a look at the current state of Web services and how SOA has been successfully standardized. It then makes the case for how, through successful agreement on and implementation of the above-mentioned standards, Web services will solve a much broader array of business problems.

WSIF and JSR-208: Flexible Binding Frameworks for Today and Tomorrow

There is a common misconception that BPEL is only useful if all of your systems are Web services. This article examines how WSIF (Web Services Invocation Framework) today enables the orchestration of nearly any legacy system as if it were a Web service, without having to explicitly wrap or publish it as a Web service. It will also highlight how JSR-208 will standardize this capability in the not-too-distant future.

Plus...

Web Services + the Grid = Prime Time

Security and the "ilities" are two major hurdles Web Services must address before they can be considered "Prime Time". We look at how the recent Grid-inspired standards proposals and Grid experiences address these missing elements.

SOA and the Art of Riding the Enterprise Service Bus

The need for the ESB and other infrastructure technologies

■ In this article (part 1 of two), I will discuss the role of the enterprise service bus (ESB) and other technologies in providing an infrastructure for service-oriented architecture (SOA) in the enterprise. This month, I'll discuss some of the middleware requirements necessary to support a full-function SOA. These requirements are driven both by the concepts of SOA, and by the need to apply them in enterprise situations. Some of the important middleware capabilities that fulfill these requirements are now characterized as an "enterprise service bus." By examining the requirements, it is possible to identify some key mandatory and optional capabilities for an ESB. These include functions such as data format or protocol translation, support for open standards such as Web services and XML, and the provision of qualities of service such as performance, security, and transactionality.

Requirements for the Enterprise Service Bus

The need for an ESB results from both the concepts of SOA and the practicalities of applying it in an enterprise. In this section, we briefly discuss some of those factors.



WRITTEN BY
RICK ROBINSON

Decoupling the Consumer View of Services from Their Implementation

Service-oriented architecture at its broadest describes:

- Viewing business as the delivery of services to consumers
- Decoupling the packaging and delivery of those services from their implementation

- Implementing those services as business processes that use a highly flexible combination of other business and technical services
- Applying this view to the supporting IT systems both by directly modelling and mapping services and processes to them, and by implementing interactions between systems as services

An example may help to make these points clear. Imagine the fictitious "Wildcat" luxury car manufacturer that offers a "roadside assistance" service to its customers. The package includes roadside and home assistance, vehicle recovery, the provision of replacement cars, and covers the cost of temporary accommodation.

However, Wildcat is not expert in any of these areas, so they subcontract delivery:

- **Roadside assistance:** To a roadside assistance specialist
- **Vehicle recovery:** To local garages with towing facilities
- **Replacement cars:** To a car rental company
- **Temporary accommodation:** To a motel chain.

Wildcat therefore offers a comprehensive service to their customers without delivering any elements of that service themselves. They can subcontract to the lowest-cost or highest-standard service provider in each area. If a new provider becomes available, offering lower cost or higher standards, Wildcat can switch to them. If Wildcat needs to offer a new package of slightly different services to some of their customers, it is easy for them to do so.

The principles of SOA apply to this example, but could also extend it in several ways:

- **Automating the provision of business services and the aggregation of service suppliers.** As described, the sourcing and aggregation of services could take place through face-to-face negotiation, telephone calls, the exchange of faxes and e-mails, etc. SOA describes the automation of these activities so that services are increasingly located, selected, and bound through interactions between systems rather than people.
- **Applying the same principles farther down the technology stack to achieve flexibility in the way technology is applied to support the business.** The ability of any business to change its services in response to marketplace demands is constrained by the agility of the IT systems that support those services. By treating IT capabilities as services and providing similar decoupling and flexibility, SOA seeks to improve the overall agility of a business.
- **Delivering outsourcing or resourcing of business services at various levels of granularity:** In the Wildcat example, we describe the outsourcing of relatively large-grained business services such as replacement cars. By extending this approach to the technology stack, SOA enables more flexible outsourcing. This might be in outsourcing finer-grained services, such as payments, credit checks

Gartner Application Development Summit

September 27 – 29, 2004

JW Marriott Desert Ridge Resort and Spa

Phoenix, Arizona

gartner.com/us/ad

Go to
gartner.com/us/ad
for agenda, session descriptions
and registration information.

Deliver lasting business value.
**Take the Path to Modern
Application Development**

Four ALL NEW tracks on today's mission-critical AD issues

*After three days of high-value learning, you'll
come away with:*

- State-of-the-art research and guidance from seasoned IT analysts
- A powerful blend of strategic insight and actionable know-how
- Hard-to-find best practices and end-user case studies
- A unbiased assessment of the leading providers and the newest solutions

Learn how to:

- Deliver software quality, security and business value
- Smoothly integrate your applications, whether in-house or packaged
- Run your IT operations and services more effectively
- Decide if outsourcing development makes sense for your enterprise
- Better manage your project portfolio and team productivity

And much more!

**Register
Now:**

gartner.com/us/ad
and use
**Promotion Code:
WSJ2004**

KEYNOTE SPEAKERS and CONFERENCE CHAIRS



Nick Carr

Author of
"Does IT Matter?"



Don Tapscott

World-renowned
technology guru



Mark Allen

Triathlete of
the Century



Theresa Lanowitz

Research Director,
Gartner



Dale Vecchio

Research Director,
Gartner

etc.; or in more dynamically outsourcing large-grained services – for example, different hotel chains could be used to provide emergency accommodation depending on current discount programmes, geographical area, etc.

In this way, we see that the decoupling of the consumer view of services from their implementation is of critical importance to an agile business; SOA enables consumers to exploit services through a specific delivery channel and brand without binding directly to the eventual provider of the service.

To fulfill this decoupling, we need to introduce the concept of intermediaries. *Intermediaries* publish services to consumers as what we will call “facade services” – the consumer binds to a facade service through an intermediary, with no direct coupling to the eventual provider of the service. It is the role of the intermediary to map the facade service to an implementation of the service, which may be another facade presented by another intermediary, or may be the actual implementation of the service. Figure 1 illustrates the provision of facade services by an intermediary, this time in a banking scenario.

One of the roles of the ESB is to provide this intermediary function, particularly within an organization. In this way, it increases the internal agility of an organization to combine its capabilities into services to offer to its customers. However, as we will see later, the same ESB capabilities

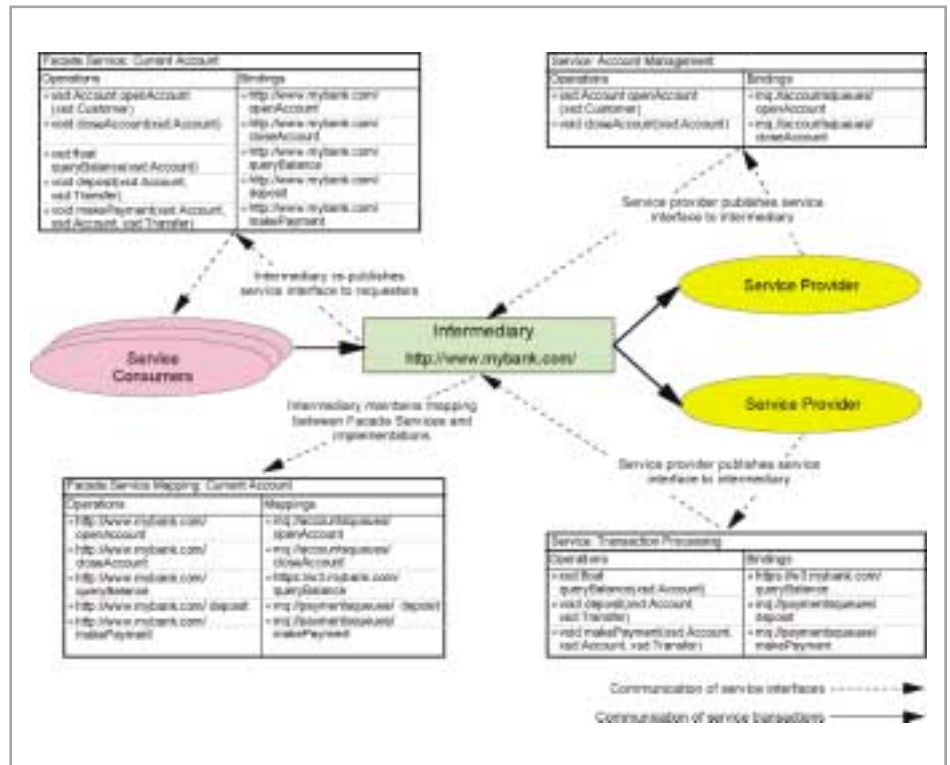


FIGURE 1 Provision of a “facade service” by an intermediary in a banking scenario. A single “Current Account” facade service is presented to consumers by the intermediary. Service provision is in fact sourced from two services, an “Account Management” service and a “Transaction Processing” service. The intermediary also performs a variety of address and protocol transformations.

ties can be applied in different scenarios, such as the “ESB Gateway,” to provide this flexibility in offering services to consumers.

Decoupling Technical Aspects of Service Interactions

While most definitions of SOA describe services as “loosely coupled,” in order to

implement real systems we need to be more specific: we can’t “loosely couple” a service (or any other interaction between systems); rather, we have to think carefully about how we couple or decouple aspects of service interactions, such as those shown in Figure 2, and including:

- The platform and language in which services are implemented, or from which they

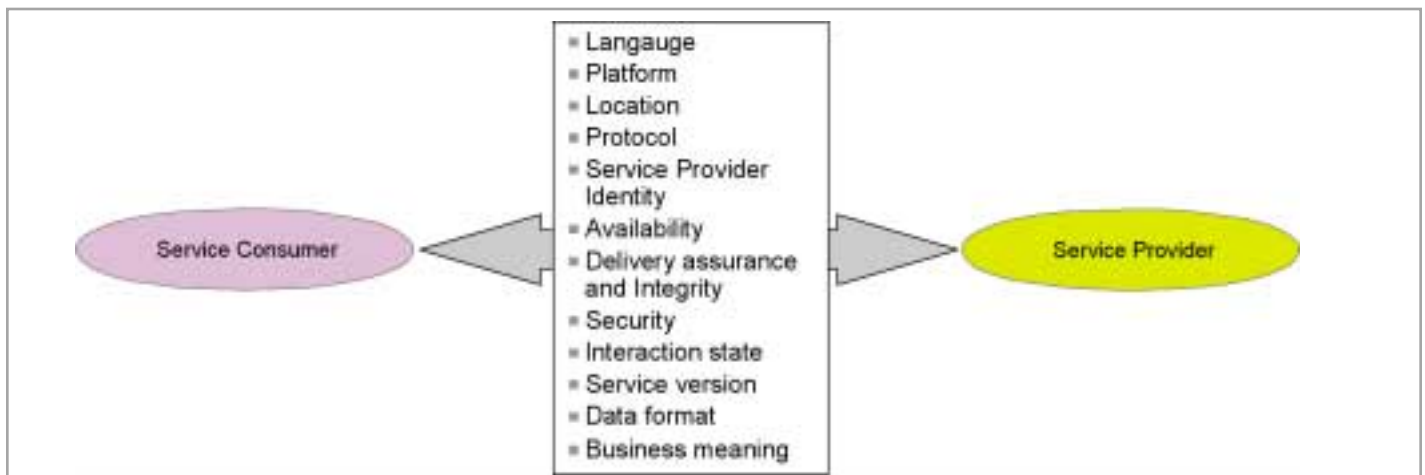


FIGURE 2 Some technical aspects of service interactions



web services
conference & expo



**Web Services Edge
2005 East**

International Web Services Conference & Expo

**Announcing Web Services Edge 2005
Extending Call for Papers to August 30th**
Submit your proposal today!

**The Largest
i-Technology
Event of
the Year!**



**Guaranteed
Minimum
Attendance
3,000
Delegates**



Tuesday, 2/15:
Conference & Expo

Wednesday, 2/16:
Conference & Expo

Thursday, 2/17:
Conference & Expo

**Hynes Convention Center, Boston, MA
February 15-17, 2005**

Join us in delivering the latest, freshest, and most proven Web Service solutions... at the Fifth Annual Web Services Edge 2005 East-International Conference & Expo as we bring together IT professionals, developers, policy makers, industry leaders and academics to share information and exchange ideas on technology trends and best practices in secure Web services and related topics including:

- Transitioning Successfully to SOA
- Federated Web services
- ebXML
- Orchestration
- Discovery
- The Business Case for SOA
- Interop & Standards
- Web Services Management
- Messaging Buses and SOA
- Enterprise Service Buses
- SOBAs (Service-Oriented Business Apps)
- Delivering ROI with SOA
- Java Web Services
- XML Web Services
- Security
- Professional Open Source
- Systems Integration
- Sarbanes-Oxley
- Grid Computing
- Business Process Management
- Web Services Choreography

3-Day Conference & Education Program features:

- Daily Keynotes from companies building successful and secure Web Services
- Daily Keynote Panels from each Technology Track
- Over 60 sessions and seminars to choose from.
- Daily Training Programs that will cover Web Service Security, J2EE, and .asp.NET
- FREE Full-Day Tutorials on .NET, J2EE, MX, and WebSphere Tutorials
- Opening Night Reception

Interested in Exhibiting, Sponsoring or Partnering?

Becoming a Web Services Edge Exhibitor, Sponsor or Partner offers you a unique opportunity to best position your organization's message and visibility to a targeted audience of Web Services Professionals. Make your plans now to reach the most qualified software developers, engineers, system architects, analysts, consultants, group leaders, and C-level management responsible for Web Services, initiatives, deployment, development and management at the region's best known IT business address, The Hynes Convention Center in Boston.

For exhibit and sponsorship information please contact Jim Hanchrow at 201-802-3066, or email at jimh@sys-con.com

Sponsored by:



WebServices
JOURNAL



SDTimes



asp.netPRO

NET JOURNAL



WebSphere
JOURNAL



All brand, and product names mentioned above are trade names, service marks or trademarks of their respective companies.

Contact for Conference Information: Jim Hanchrow, 201-802-3066, jimh@sys-con.com



www.sys-con.com/edge

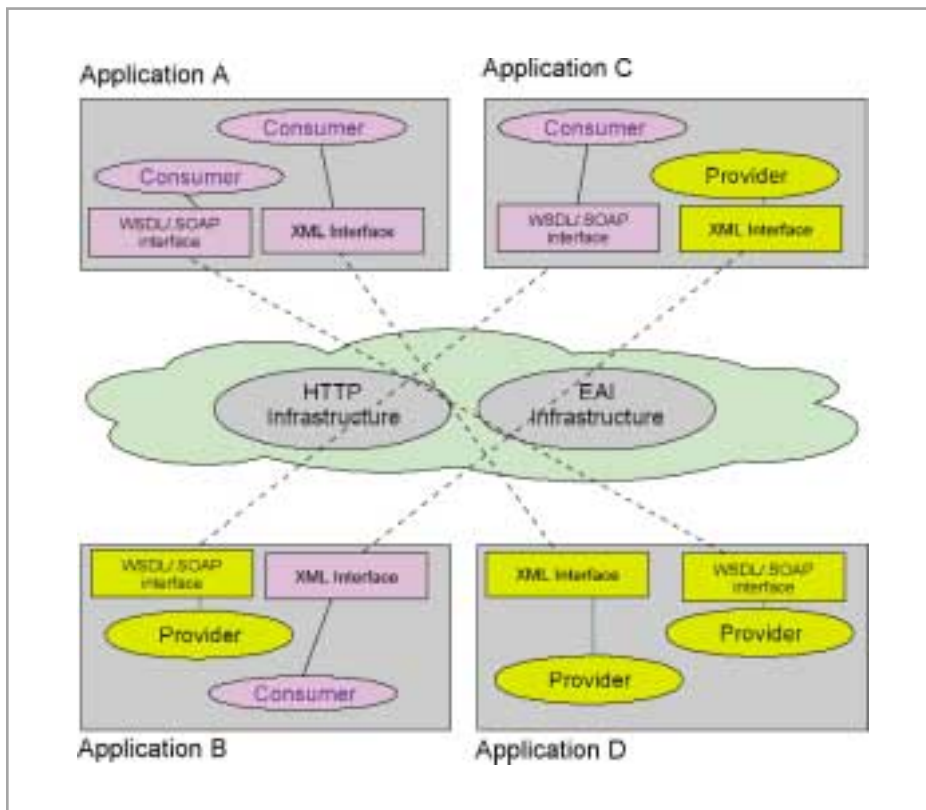


FIGURE 3 | Point-to-point integration using Web services and other technologies consistent with SOA principles

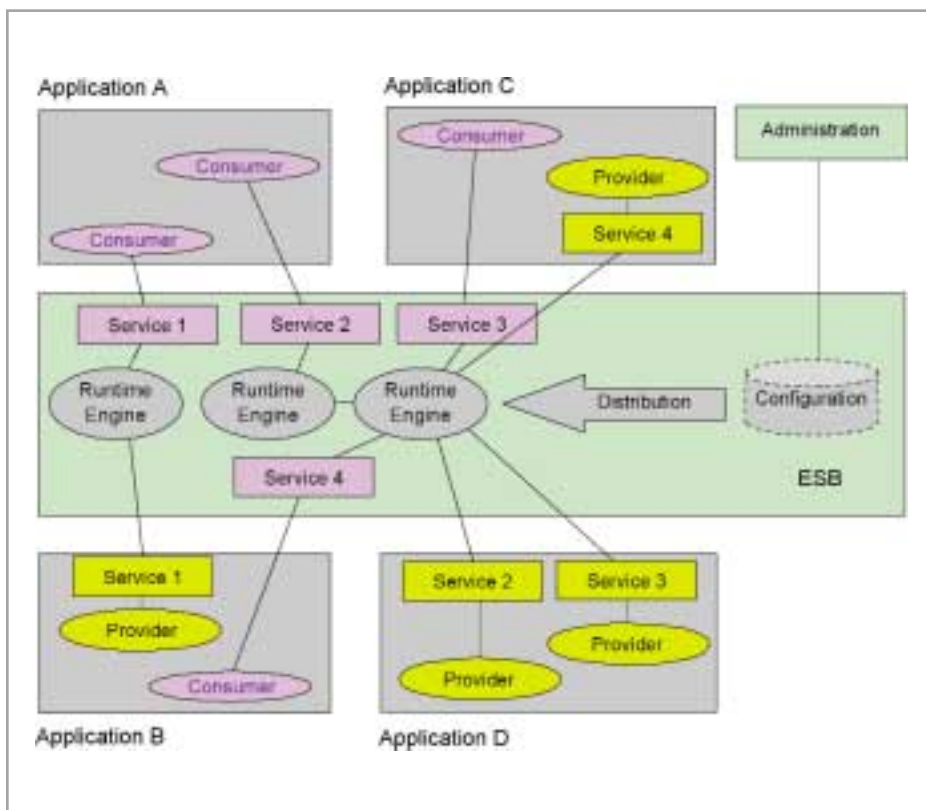


FIGURE 4 | A consistent management and administration approach to a distributed integration infrastructure. Note that this figure also clearly shows the role of the ESB in presenting facade services (colored pink) to service consumers, and mapping those to service implementations offered by service providers (colored yellow).

- are requested
- The communication protocols used to invoke services
- The data formats used to exchange input and output data between service consumers and providers

We also want to handle some of the more difficult technical aspects of transactions outside application code; at the same time, applications can't ignore them – ideally we'd like to declaratively specify them, similar to the J2EE model of deployment descriptors, and have the infrastructure take care of them. This might apply to such aspects of interactions as:

- How service interactions are secured
- How the integrity of business transactions and data is maintained through reliable messaging or the use of transaction monitors or compensation techniques
- The invocation of alternative service providers in the event that the default provider is unavailable

In this way, it increases the internal ability of an organization to combine its capabilities into services, and allow those services to be invoked by consumers within the organization. However, as we will see later, the same ESB capabilities can be applied in different scenarios, such as the "ESB Gateway," to provide this flexibility in offering services to external consumers. Therefore, some of the capabilities that are required in an ESB are to:

- Map service requests from one protocol and address to another
- Transform data formats
- Support a variety of security and transactional models between service consumers and service providers, recognizing that consumers and providers may support or require different models
- Aggregate or disaggregate messages
- Support communication protocols between multiple platforms with appropriate qualities of service
- Provide messaging capabilities such as message correlation and publish/subscribe to support different messaging models within an SOA, such as events, asynchronous request/response, etc.

Integrating and Managing Services in the Enterprise

The use of basic communication technologies or Web services may solve individual prob-

“ it increases the internal agility of an organization to combine its capabilities into services to offer to its customers ”

lems, but if unchecked, it will result in a point-to-point integration style that will quickly become unmanageable. Consider Figure 3, which depicts the basic use of SOAP/HTTP or other technologies appropriate to SOA to perform integration. We quickly enter a complex, uncontrolled scenario with multiple security and transaction models, routing control distributed throughout the infrastructure, and probably no consistent approach to logging, monitoring etc.

The addition of a simple routing capability to this picture provides a first overall point of control. Additional capabilities (such as security, store and forward capability to provide delivery assurance, etc.) may be required in some scenarios. Note that this simple routing capability, while it appears to be a hub-and-spoke approach, in fact is consistent with a distributed bus model because we have provided a consistent management and administration model to an infrastructure that may be physically distributed (see Figure 4).

Another consideration in a realistic enterprise scenario is the need to integrate a potentially large number of heterogeneous service providers (such as CRM, ERP, legacy systems, or modern distributed applications using application servers like IBM

WebSphere Application Server or Microsoft's .NET platform). It is likely that each of these will have their own integration techniques, protocols, security models, etc. Ideally, we don't want to expose this complexity to service consumers – we need to offer them a simpler model. To achieve this, the ESB is required to mediate between the multiple interaction models understood by service providers and the simplified view provided to consumers.

Summary

In my next article I will go on to describe some specific situations in which an ESB might be deployed and discuss the ESB capabilities in relation to other architectural components that support SOA. Finally, I'll briefly consider some of the technology options for implementing the ESB ©

About the Author

Rick Robinson is an IT architect in IBM Software Services for WebSphere, based at the Hursley Laboratory in the United Kingdom. He has seven years of experience in the IT industry, and his roles have included solution architecture, design, and development. Rick holds a PhD in the physics of superconducting devices from the University of Birmingham. His areas of expertise include distributed systems design, the WebSphere platform, and service-oriented architecture

■■■ rick_robinson@uk.ibm.com

Subscribe Today!

– INCLUDES –
FREE
DIGITAL EDITION!
(WITH PRINT SUBSCRIPTION)
GET YOUR ACCESS CODE
INSTANTLY!



The major infosecurity issues of the day... identity theft, cyber-terrorism, encryption, perimeter defense, and more come to the forefront in ISSJ the storage and security magazine targeted at IT professionals, managers, and decision makers

SAVE 50% OFF!

(REGULAR NEWSSTAND PRICE)

Only \$39⁹⁹ ONE YEAR 12 ISSUES

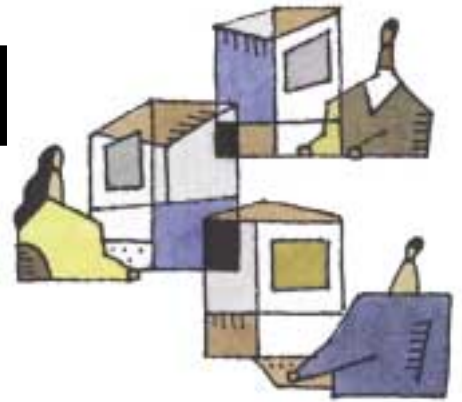
www.ISSJournal.com
or 1-888-303-5282

SYS-CON MEDIA

The World's Leading IT-Technology Publisher

Service-Oriented Architecture

Components and modeling can make the difference



■ In many respects, SOA is an evolution of the fundamental tenets governing component-based development (CBD). It also represents a quantum leap in bringing business and information technology into closer alignment through a set of SOA services grounded in business goals in support of business processes. While SOA services are visible to the service consumer, their underlying components are transparent. For the service provider, the design of components, their service exposure and management reflect key architecture and design decisions that enable services in SOA. Making these decisions requires an understanding of an SOA's components and SOA modeling to identify, classify, specify, and structure service-enabled components. In this article we briefly discuss the relationship between CBD and SOA, followed by a discussion of SOA architecture and design decisions. We describe the basic SOA components and building blocks, and a prescriptive technique for performing SOA modeling, analysis, and design for services, where services are realized using components..

Introduction

A brief review of object-oriented (OO) development and CBD will help you understand the evolution of SOA for service-oriented development. The world of objects started with the introduction of object-oriented programming languages and matured into modeling, later

WRITTEN BY



ALI
ARSANJANI



BERNHARD
BORGES



KERRIE
HOLLEY

adding techniques for design and then maturing into OO analysis and design (OOAD) methods. Infrastructure services, tools, development platforms, patterns, and reference architectures followed.

Component-based development (CBD) ensued, offering a new approach to the

design, construction, implementation, and evolution of software applications. Applications are assembled using components from a variety of sources and the components are written in different programming languages and different development environments, and run on different platforms. Just as with OO development, infrastructure services, tools, development platforms, and patterns matured to make CBD a reality.

Both OO and CBD address software economies of scale that require reuse in software development. The achievement economies of scale in software production and deployment has been elusive. Object-oriented development is an enabler, component-based development mandatory. CBD provides an opportunity for greater reuse than what is possible with OO development. SOA, with the underpinnings of both OO development and CBD, provides an even greater opportunity for reuse. SOA becomes a new mandate for achieving economies of scale in software engineering.

Evolution of SOA

The Internet and XML standardization efforts gave way to the need for a service definition and description published by a service provider (SP) that can be located and invoked by a service consumer (SC). The service description can be implemented by a number of service providers, each offering

various choices of qualities of service (QoS) based on technical requirements in the areas of availability, performance, scalability, and security. The invocation can be across Internet or intranet in a distributed object connotation and standards such as WSDL and SOAP have been created.

The roles of the SC and the SP is illustrated in Figure 1. Web services standards and



FIGURE 1 Canonical view of SOA

technologies pave the way and breathe life into the architectural style of SOA.

Infrastructure services in areas of service management and service security are required, along with new technologies to make the economies of scale made possible by SOA a reality.

SOA is the architectural style that supports loosely coupled services to enable business flexibility in an interoperable, technology-agnostic manner. SOA consists of a composite set of business-aligned services that support a flexible and dynamically re-configurable end-to-end business processes realization using interface-based service descriptions. The inherent, salient advantage of today's service descriptions is the decoupling of the SP and SC through open standards and protocols, and the deferment of implementation decisions from the SC to the SP.

Moreover, individual or collections of services that enjoy various levels of granularity can be combined and choreographed to produce "new" composite services that not only introduce new levels of reuse but also allow the dynamic reconfiguration of business systems.

Hence, service-oriented development, which SOA enables, is an evolutionary software engineering approach enabled by component-based and object-oriented development. However, the incredibly potent fusion of business and IT is a quantum leap. The flexibility of binding an SC to a new SP, based on changing QoS needs and the sta-



FIGURE 2 Service provider and service consumer views

bility of interactions through SD publication, and its invocation through standard protocols is a prerequisite for an architecture where flexibility and economies of scale are the primary movers.

SOA encourages the closure of the business-IT gap by emphasizing the reuse of rationalized, normalized services and introducing flexibility that allows end-to-end business process modeling and implementa-

multiple, possibly competing, service providers. The services, in turn, need to be "powered" by a functional piece, or pieces, of software that lends itself to a component-based view of the world. Within components we can find the design principles of object-orientation defining the internal structures of components. Thus, service modeling is about identifying the right services, organizing them in a manageable hierarchy of composite services (smaller grained often supported larger grained), choreographing them together for supporting a business process. On the provider side, these services are either allocated directly to component-based containers for realization of their functionality, or realized by adapting existing legacy functionality.

This is the fundamental evolution in service-oriented software engineering: the

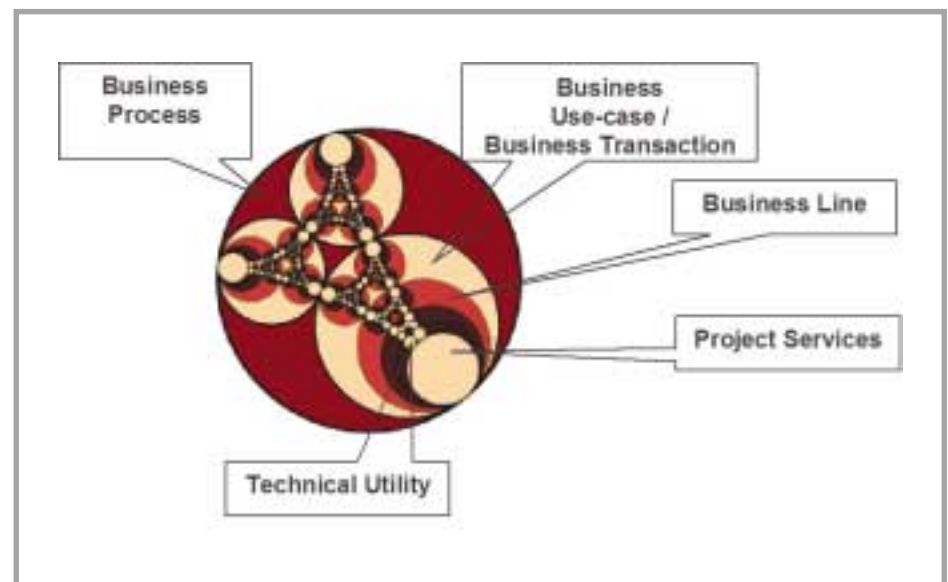


FIGURE 3 SOA fractal granularity

tion. This flexibility arises through the ensuing ability to choreograph existing composite services at near real time, i.e., "right-time", using business analyst tools, such as IBM's WebSphere Business Integration (WBI) BPEL modeler.

Component-Based Development and Service-Oriented Architecture

The concepts and disciplines of OO development and CBD should be applied to provide the appropriate frameworks guiding the design and development of SOA services.

A service description is often realized by

service consumer doesn't have to be concerned with the implementation or realization of the service, as long as it supports the required functionality and quality of service. This is called the Consumer View. Alternatively, the Provider View offers a perspective on how to design the realization of the component that offers the services; its architectural decisions and designs.

One key differentiation with more traditional OO is in the fact that we are no longer creating large object models, but rather designing the internals of larger-

grained, business-aligned component boundaries, through finer-grained object-orientation.

SOA introduces two views on components and services: the service consumer and the service provider (see Figure 2).

Components are of no concern to service consumers as long as their service-level agreements and contracts on functional and non-functional requirements are fulfilled. Service providers need to design a service realization that will meet those requirements through decisions on how they contain, manage, and implement the service description. The inward view is that of a user or consumer who sees only the service available through the provider, and the outward view where the services provider creates components that provide the services through their interfaces. In order to do so, they require an internal set of finer-grained, or peer, components that will participate in the collaborations needed to fulfill the services. Thus, within the service provider components, we will have domain-level components that correspond closely to domain object models in the traditional object-oriented sense and are implemented through standard component container mechanisms such as application servers.

The SOA model is fractal. That is, a service can be composed of finer-grained services such as those providing technical utility such as logging, security, or authentication. This fractal granularity is illustrated in Figure 3.

In addition to services that provide technical utility, there are services that may be created in the context of a project, or services that are created for utility by a business line or line of business. Each of these services can be used to fulfill a business process or a business transaction. The components that are integral to services are also fractal. This requires that both services and components be designed with the appropriate level of granularity, which is a key SOA design decision.

SOA Architectural and Design Decisions

Several architecture and design decisions are essential for SOA to meet its stated goals and benefits. The SOA architectural decisions generally relate to the choice of technologies and products necessary to meet the

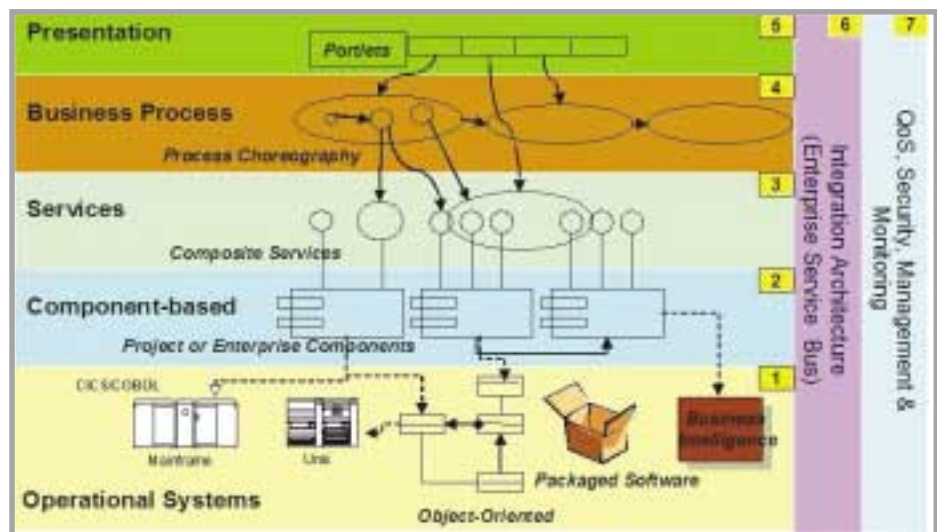


FIGURE 4 Layers and components of a service-oriented architecture

quality of service attributes required of a business process. Design choices focus on choices that must be made about services using the technique described later on service-oriented analysis and design. Some of the key decisions are:

- *Service identification* design decisions are essential for SOA success. Just as the proliferation of objects did not allow organizations to realize economies of scale as reuse is limited largely because of the lack of business utility for object proliferation the same is true if services are proliferated. Many early adopters of SOA and Web services realized quickly that the proliferation of Web services does not make for a sound SOA model. These design choices are made during the service-oriented analysis and design described later in this article.
- *Container design and realization* architecture and design decisions are critical for a service to provide the critical qualities for extensibility and maintainability.
- *Granularity* design choices about services must be matched to the level of reusability and flexibility required given the context. Larger-grained services can help encapsulate changes in finer-grained technical services that can tend to change more frequently than the higher-level business-level service interface. Factors of flexibility will be key criteria for encapsulation rather than merely the encapsulation of function.
- *State of service* often requires both architecture and design decisions on the state-

less nature of services, yet state must be maintained. This is often addressed by making architectural decisions on choreography engines (e.g., a BPEL engine) and design choices on the stateless nature of a proposed business service.

- *Loose coupling and dynamic reconfiguration* are a design decision that requires the decoupling of interfaces from implementation and protocol realization via open standards. The connection between service consumers and service providers needs to be stable and well-structured and yet flexible and readily reconfigurable. Reconfigurable means that existing service consumers and service providers can be re-assembled with their functionality intact to obtain business solutions in different technical environments and with different operational constraints with new business partners across a value chain. Therefore, integration alone is no longer adequate; dynamic reconfiguration is the name of the game and needs to be pervasive along a spectrum of support in areas of infrastructure (i.e., the enabling software infrastructure), tools, development platforms, reference architectures, patterns, methods and industry-specific reference models.

SOA Layers: The Components of an SOA

So far we've explored the relationship between services and components and have said that components can be large-grained enterprise or business-line components and

Consumer view	Service Identification	Service Classification	Service Exposure	Choreography
Provider View	Service Allocation	Component Specification	Service Realization	Service Management

TABLE 1 Activities of service-oriented modeling, analysis, and design

can be used to realize the functionality and management of exposed services as interfaces. That is, we have discussed components used to realize services. Now we will look at SOA components, not the components used to realize services in SOA.

The major components of an SOA are:

- **Services portfolio:** Describes the business services in SOA. This includes a list, classification and hierarchy of services defined through the technique of service-oriented analysis and design described later.
- **Components:** Provide the functional realization of the services.
- ^a **Service providers, service consumers, and optionally, the service broker(s):** With their service registries where service definitions and descriptions are published.
- **SOA layers:** Where components and services reside.

Figure 4 illustrates the SOA layers. For each of these layers, there are design and architectural decisions to be made.

Layer 1, the bottom layer, describes operational systems. This layer contains existing systems or applications, including existing CRM and ERP packaged applications, legacy applications, and “older” object-oriented system implementations, as well as business-intelligence applications. The composite layered architecture of an SOA can leverage existing systems, integrate them using service-oriented integration.

Layer 2, the component layer, used container-based technologies and designs in typical component-based development.

Layer 3 provides for the mechanism to take enterprise-scale components, business unit-specific components, and in some cases project-specific components and provides services through their interfaces. The interfaces get exported out as service descriptions in this layer, where services exist in isolation or as composite services.

Level 4 is an evolution of service compo-

sition into flows or choreographies of services bundled into a flow to act as an application. These applications support specific use cases and business processes. Here, visual flow composition tools can be used for design of application flow.

Layer 5, the presentation layer, is usually out of scope for an SOA. However, it is depicted because some recent standards such as Web Services for Remote Portlets version 2.0 may indeed leverage Web services at the application interface or presentation level. It is also important to note that SOA decouples the user interface from the components.

Level 6 enables the integration of services through the introduction of reliable and intelligent routing, protocol mediation, and other transformation mechanisms, often described as the enterprise service bus.

Level 7 ensures quality of service through sense-and-respond mechanisms and tools that monitor the health of SOA applications, including the all-important standards implementations of WS-Management.

Service-Oriented Modeling, Analysis, and Design

So far we’ve spoken about the end result of SOA, architecture with a composite set of business-aligned services. To create SOA, we need to identify, specify, and design services and their choreography from the angles of consumer and provider. The modeling, analysis, and design of services are not quite the same as the world of OO. There are additional activities that need to be done.

Major activities of a service-oriented modeling approach

were described in an IBM Redbook. Also, Zimmerman et al. (see References) describe a case for, and context of, service-oriented analysis and design (SOAD). In this section we describe these key differentiating activities and provide a

technique for the modeling, analysis and design of SOA services. Specifically, we examine how the services of an SOA are identified, specified, and realized.

An SOA does not start only from the bottom-up as is often the case with a merely Web services-based approach. There are a number of important activities and decisions that influence not just integration architecture but enterprise and application architectures as well. They include the activities from both the consumer and provider views described in Table 1.

Service identification consists of a combination of top-down, bottom-up, and middle-out techniques. In the top-down view, a blueprint of business use cases provides the specification for business services. This top-down process is often referred to as domain decomposition, the decomposition of the business domain into its functional areas and subsystems, including its flow or process decomposition into processes, subprocesses and high-level business use cases. These use cases are often very good candidates for business services exposed at the edge of the enterprise.

In the bottom-up portion of the process, or existing asset analysis, existing systems are analyzed and selected as viable candidates for providing lower-cost solutions to



the implementation of underlying service functionality that supports the business process. In this process, we analyze and leverage APIs, transactions, and modules from legacy and packaged applications. In some cases, componentization of the legacy systems is needed to remodularize the existing assets for supporting service functionality.

The middle-out view consists of goal service analysis to validate and unearth other services not captured by either top-down or bottom-up service identification approaches. It ties services to goals and sub-goals, key performance indicators, and metrics.

Service classification is launched once services have been identified. It is important to start service classification in a service hierarchy, reflecting the composite or fractal nature of services: services can and should be composed of finer-grained components and services. Classification helps determine composition and layering as well as coordinates building of interdependent services based on the hierarchy. Also, it helps alleviate the service proliferation syndrome in which an increasing number of small-grained services get defined, designed, and deployed with very little governance, resulting in major performance, scalability, and management issues. More importantly, service proliferation fails to provide services which are useful to the business which would allow for the economies of scale to be achieved.

Subsystem analysis takes the subsystems found above during domain decomposition and specifies the interdependencies and flow between the subsystems, and puts the use cases identified during domain decomposition as exposed services on the subsystem interface. The analysis of the subsystem consists of creating object models to represent the internal workings and designs of the containing subsystems that will expose the services and realize them. The design construct of "subsystem" will then be realized as an implementation construct of a large-grained component realizing the services in the following activity.

Component specification is the next major activity; the details of the component that will implement the service(s) are specified: data, rules, services, configurable profile, and variations. Messaging

and Events specifications and management definition occur at this step.

Service allocation is the process of assigning services to containers that will realize their published functionality. Structuring of components occurs when we use patterns to construct enterprise components with a combination of mediators, facade, rule objects, and configurable profiles and factories. Service allocation also consists of assigning the services and the components that realize them to the layers in your SOA. Allocation of components and services to layers in the SOA is a key task that will require the documentation and resolution of architectural decisions that relate not only to the application architecture but also to the technical operational architecture designed and used to support the SOA realization at runtime.

Service realization recognizes that the software that realizes a given service must be selected or custom built. Other options that are available include integration, transformation, subscription, and outsourcing of parts of the functionality using Web services. Here you decide which legacy system module will be used to realize a given service and which services will be built from the "ground-up." Other realization decisions for services outside of business functionality include: security, management and monitoring of services.

Conclusions

SOA extends CBD into the realm of ubiquity and pervasive access through extending the reach of callable components by using non-proprietary, interoperable protocols, service descriptions that separate of interface from implementation. SOA will have a more profound impact on software engineering than what we have been accustomed to with object-oriented and component-based development. Executing a sound service-oriented analysis and design technique will be a critical success factor to make this promise a reality as SOA itself has components and layers that need to be defined and designed.

References

- Endrei, M.; Ang, J.; Arsanjani, A.; Chua, Sook; Comte, Philippe; Krogdahl, Pal; Luo, Min; and Newling, Tony. (2004) *Patterns: Service-oriented Architecture and Web*

Services. IBM Redbook, ISBN 073845317X. www.redbooks.ibm.com/redbooks/SG246303/wwhelp/wwhtml/java/html/wwhelp.htm

- Levi, K.; Arsanjani, A. "A Goal-oriented Approach to Enterprise Component Identification and Specification." *Communications of the ACM*, Oct 2003.
- Arsanjani, A. "The Enterprise Component Pattern." *Proceedings of Pattern Languages of Programming*, 2000.
- Zimmerman, O.; Korgdahl, P.; and C. Gee, C. "Elements of Service-oriented Analysis and Design", IBM developerworks. June 4, 2004.
- Keen, Martin; Bishop, Susan; Hopkins, Alan; Milinski, Sven; Nott, Chris; Robinson, Rick; Adams, Jonathan; and Verschueren, Paul. (2004) *Patterns: Implementing an SOA with the Enterprise Service Bus*. IBM Redbook ISBN SG24-6346-00. ©

About the Authors

Dr. Ali Arsanjani is a senior technical staff member and chief architect of the SOA and Web Service Center of Excellence in IBM Global Services. He has 21 years of experience in software development and architecture. He holds a PhD in computer science from DeMontfort University. His areas of expertise include patterns, component-based and service-oriented software architecture and methods.

■ ■ ■ arsanjan@us.ibm.com

Bernhard Borges is a technical executive and solutions architect in the IBM Software Group's Enterprise Integration group. He specializes in distributed, open, interoperable, and portable systems, especially in the context of SOA, ESB, and Web services. Bernhard is an IBM Distinguished Engineer and a member of IBM's Academy of Technology. He is also a member of the IBM Software Architecture Board and several workgroups, IBM's SOA and Web Services Technical Council, and an extended member of IBM's BCS SOA-Web Services Center of Excellence.

■ ■ ■ bborges@us.ibm.com

Kerrie Holley is currently a Distinguished Engineer in IBM Global Services, a member of IBM's Academy of Technology, and a chief architect in the Business Consulting Services (BCS), Application innovation Services (AIS) group. Kerrie is the CTO for IBM's Center of Excellence for Web Services and SOA. His area of expertise is in software engineering, end-to-end advanced Web development, adaptive enterprise architecture, conducting architecture reviews, Web services, and service-oriented architecture.

■ ■ ■ KLHOLLEY@US.IBM.COM

The Gilbane Conference

ON CONTENT MANAGEMENT TECHNOLOGIES

November 30-December 2, 2004
Boston, Massachusetts
The Westin Copley Place



Lighthouse Seminars



Content technologies have become mainstream and need to be part of all major enterprise applications and integrated into IT architectures and infrastructures. Join us to learn what you need to know to formulate a successful strategy from our gathering of content technology experts and practitioners.

What you will take away from the conference:

Attendees benefit from an unbiased, deep, and up-to-date understanding of content management technologies, vendors, trends, and best practices, from the most experienced and respected experts in the field. Our speakers have implemented every kind of content management system across all industries, and have written the books and reports that others depend on.

WHO SHOULD ATTEND:

IT Professionals

- IT Strategists, Managers, Staff
- Content Management Project Managers
- Content Management System Designers
- Intranet, Internet, Extranet, Portal Managers
- Webmasters, Developers, and Administrators
- Enterprise Architects

Business Managers and Strategists

- Product Data Managers
- Information Architects/Knowledge Managers
- Technical Documentation Managers
- Business, Market, and Technology Analysts
- Consultants and Integrators
- Marketing and Product Executives
- Brand Managers

Register NOW!

Register now for the **CONFERENCE PLUS PACKAGE** and receive a **FREE IPOD!**
LIMITED QUANTITY, REGISTER EARLY!



LEARN about successful implementation of content technology through the unique combination of

- **Case Studies**
- **Implementation Techniques**
- **Technology Analysis**

"The Gilbane Conference has provided me with crucial information that I was looking for as my company rolls out its content management solution. I have been able to network with industry experts, the major content management technology players and other companies, like mine, that are faced with the challenge of effectively managing content," says Angie Khumdee, Senior Software Engineer, Motorola CGISS E-Business

For more detailed information, please visit:
www.lighthouseseminars.com or call
Joe Richard at **781.821.6734**

Gold Sponsors

EMC² | documentum

GMC

INTERWOVEN

STELLANTTM



VIGNETTE[®]
the efficiency experts[™]

Media Sponsors

BusinessWire
Smart Solutions for Your News

cms
WATCH

EContent

GILBANE
REPORT

InfoWorld
GET TECHNOLOGY RIGHT

TRANSFORM

WebServices

XML JOURNAL

Association Sponsors

IDEAlliance[®]

OASIS

Bulletproof Web Services

Follow basic principles

■ Web services are gaining industry-wide acceptance and usage. They are moving from proof-of-concept deployments to actual usage in mission-critical enterprise applications. While Web services allow businesses to connect to partners and customers, the same flexibility and connectivity provide an increased opportunity for errors.

As companies and consumers rely more on Web services, it is increasingly important for Web services developers to know how to properly design, develop, deploy, and ultimately manage a Web services system. However, because of the inherent complexities that can arise with a Web service implementation, it can be difficult to grasp practical fundamentals and devise a step-by-step plan for Web services development.

We will look at the nuts and bolts of implementing and deploying a reliable, high-quality integration system – or rather, a bulletproof Web service. This article explains issues specific to Web services and illustrates the engineering and testing practices required to ensure complete Web service functionality. First, we will discuss the planning and design of a sample Web service. Then, we will discuss the infrastructure needed to ensure that the Web service functions properly. Whether creating Web services from scratch or integrating legacy back-end servers via Web services, the practices and principles outlined in this paper will be of great benefit.

Web Service Creation: Planning and Design

To make the discussion as concrete and pragmatic as possible, a sample Web service implementation is discussed. The example



WRITTEN BY
DR. ADAM KOLAWA

is a service for a large realtor with office branches across the country. This realtor needs to implement a Web services initiative that supports the following requirements:

- Potential and existing customers will submit contact information, desired living location, and desired price range of a home via the Web service. These users should receive a response from the server that gives them the location of the branch closest to them, as well as an estimate of the monthly mortgage. This will enable users to contact a real estate agent and begin the process of finding a home.
- Real estate agents from different branches will submit a request for a list of potential customers who are looking for homes in the local area. This will enable the real estate agents to earn business and establish contact with interested customers.

Target Requirements

Two target requirements are needed to build the example Web service. As the name suggests, these targets are landmarks within the development process that your team should aim for. These targets will help to drive the feature set of your Web service and enable you to measure your progress. When these targets are reached, you'll know you are on the right path.

Target 1: A use case scenario shall pass after meeting these requirements:

1. A dummy request is sent to the Web service for customers.
2. A SOAP response is received.
3. Verify that the SOAP response contains a SOAP fault.

Target 2: A use case scenario shall pass after meeting these requirements:

1. A valid request is sent to the Web service for agents.
2. A SOAP response is received.
3. Verify the SOAP response contains a list of customers that may be of length zero.

A test case is created for each target, to verify that each requirement is met. At the beginning, each test case should return an "incomplete" failure message to clearly indicate that the related feature has not yet been implemented. These test cases will continue to fail until the feature is implemented. This example starts with two targets, but this number is arbitrary. For a bigger project, you may want to have 10 targets, each one measuring an incremental step. You can track how close you are to completing your project by monitoring the test cases for these targets.

You may use various frameworks or tools to create test cases for the targets. Whatever framework or tools you decide to use, the use case scenario involves a SOAP Client sending a message, waiting for a response, and then verifying the response.

Robustness Requirements

When test cases for our targets succeed, we can begin to flesh them out and verify that the new feature is implemented correctly. As seen in Figure 1, robustness requirements should be met before moving on to the next target feature.

While the target requirements drive the features set, these additional requirements ensure the robustness of the Web service:

- **Normal Use:** The Web service must function in the manner for which it was designed. For each operation exposed through the Web service, the request and response pair should adhere to the binding, and the XML should conform to the message description. In short, the server and client send and receive what is expected.
- **Abnormal use:** The Web service must

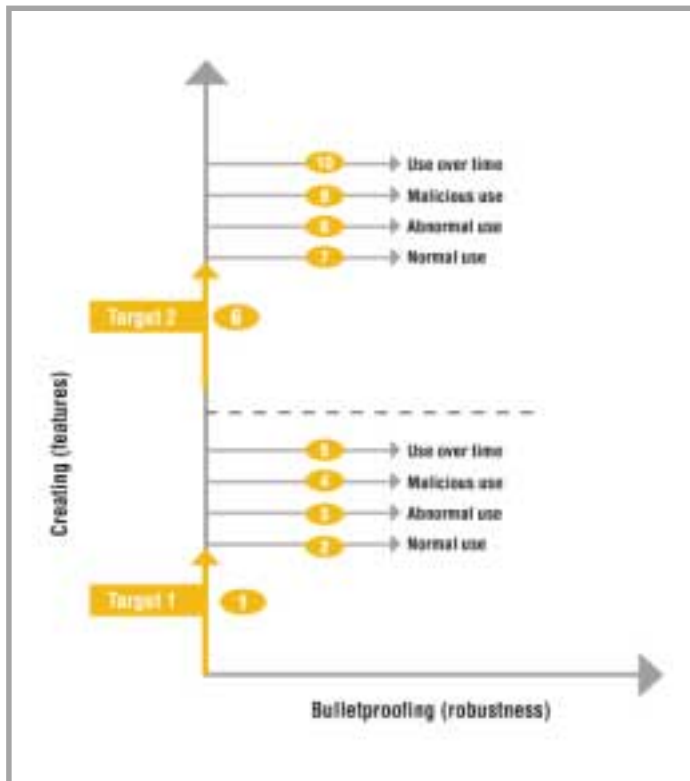


FIGURE 1 Target and robustness requirements

function even when it is being consumed outside the lines of its intended use. An abnormal use case would involve sending a value other than those expected or not sending a value at all. For example, one application may send an XML instance document based on an older version of schema, and the receiving application may use a newer version of schema. In any case, a Web service should alert the consumer appropriately without any malfunctions.

- **Malicious Use:** The Web service must function even when it is deliberately and maliciously being consumed outside the lines of its intended use. For example, hackers may try to gain access to privileged information from a Web service transaction without authorization, or they may attempt to undermine the availability of the Web service. To be able to function under, or even prevent, malicious use, a Web service should have security measures in place.
- **Use over time:** A Web service implementation is likely to change over time. For example, perhaps a Web service exposes an application that is undergoing an iterative development process. Any Web service must continue to function properly during its entire life span, even as it is evolving.

Initial Architecture

Before jumping into the steps necessary to fulfill the target and robustness requirements above, you must be aware of the parts of the Web service that will need to be developed, tested, and verified. The initial architecture of the sample Web service will be comprised of the following (see also Figure 2):

- **Application logic (or business logic):** Handles requests from customers and agents, makes necessary connection to the database, and returns responses to customers and agents.
- **Database:** Stores relevant information about customers and agents.
- **Server:** SOAP-enabled HTTP server that handles serialization from XML to objects for the application logic. The Apache Axis SOAP engine is an open source SOAP implementation that can be deployed on any J2EE server.
- **Proxy server:** Allows for security and access management, so customers and agents have different levels of access to the available Web service.
- **WSDL (Web Service Description Language) document:** A description of the Web service.
- **Client:** The Web service client that the customers and agents will use to invoke the Web services.

Critical Infrastructure

Now that the basic necessities of the service to be created have been explained, you must concentrate on the foundations from which this service can be built. For Web service development to be successful, specific practices must be implemented correctly and consistently throughout your development group. This consistent application requires you to ensure that your development group has an appropriate supporting infrastructure, then ensure that the group follows a workflow from which error prevention practices are

WSJ ADVERTISER INDEX			
ADVERTISER	URL	PHONE	PAGE
2004 RCAs	www.sys-con.com	888-303-5282	37
Active Endpoints	www.activeendpoints.com		17
Borland	http://info.borland.com/conf2004/		23
Gartner	www.gartner.com/us/ad		29
Gilbane Conference	www.lighthouseseminars.com	781-821-6734	39
IBM	www.ibm.com/middleware/information		5
Information Storage+Security	www.ISSJournal.com	888-303-5282	33
Mindreef	www.mindreef.com		9&11
OpenLink Software	www.openlinksw.com/virtuoso/	800-495-6322	Cover II
Parasoft	www.parasoft.com/AchieveQuality	888-305-0041	6
SAP	www.sap.com/usa/teched		Cover IV
Service Integrity	www.serviceintegrity.com		3
SpeechTEK 2004	www.speechtek.com	877-993-9767	21
Strikelron	www.strikeiron.com	919-405-7010	15
SYS-CON Publications	www.sys-con.com/2001/sub.cfm	888-303-5282	43
Web Services Edge East	www.sys-con.com/edge	201-802-3066	31
WebAppCabaret	www.webappcabaret.com/ws.jsp	866-256-7973	Cover III
XML-J Resource CD	www.sys-con.com/freecd	888-303-5282	57

Advertiser is fully responsible for all financial liability and terms of the contract executed by their agents or agencies who are acting on behalf of the advertiser. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

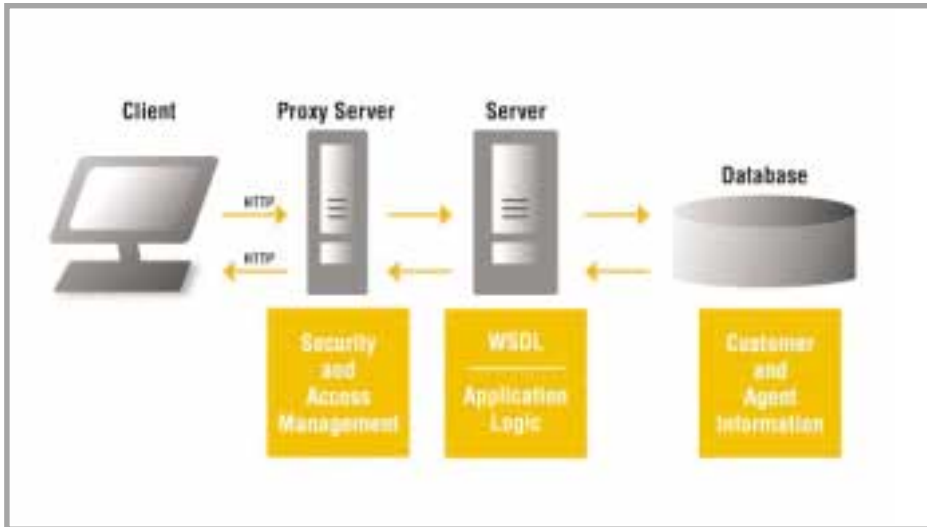


FIGURE 2 Initial architecture

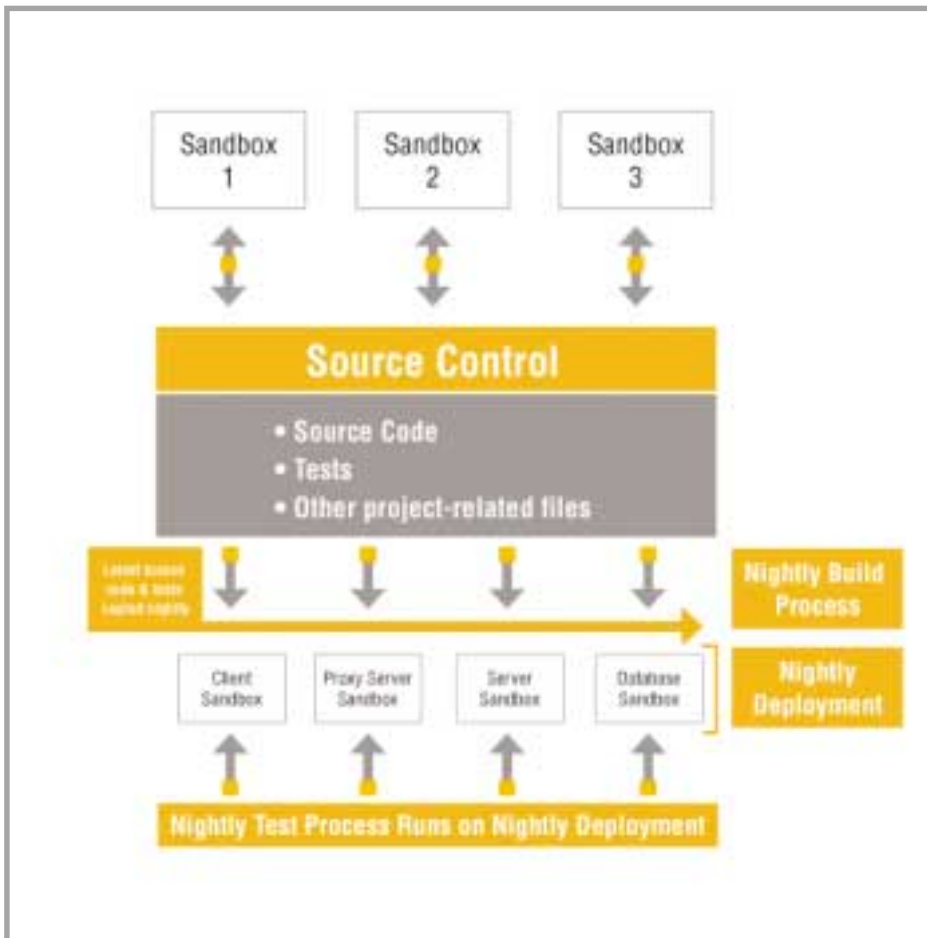


FIGURE 3 Critical infrastructure

performed appropriately. Until this critical infrastructure is in place, you cannot expect a team to begin the development of a bulletproof Web service.

As seen in Figure 3, your development group must have a functioning source con-

trol system and automated build process before its members can begin writing code. A source control system and an automated build process are the fundamental requirements needed to ensure the development of quality Web services. As you will see, estab-

lishing this infrastructure provides the necessary framework for creating reliable Web services.

Source Control

A source control system is a database where source code is stored. Its purpose is to provide a central place where the team members can store and access the entire source base. There are two main reasons why we require a source control system.

First, source control gives each developer the freedom and safety to write, modify, and refactor code – even when it is risky to do so on his own sandbox (the concept of a sandbox will be explained later). If a code change turns out to be undesirable, the developers can easily undo their changes by reverting back to the code in the source control. While they are working on their changes, the rest of the developers always have a working version of the code.

Second, having a source control system is a prerequisite for the nightly build process. All of the files needed for the build process should be in source control. As explained later, the nightly processes access all required files from the source control.

Most organizations do not understand how to effectively use their source control system. Many simply underuse their source control system, don't require its use at all, or have it configured incorrectly for the group environment. To put your source control system to proper use, it is important that you understand and establish guidelines for your developers. Source control systems are so important that, without a properly configured system, quality software cannot be made.

Using a Sandbox

A *sandbox* is an area where copies of source code and other project-related files can be stored and manipulated without affecting the master source-code base. As I mentioned earlier, the reason that each developer should have his or her own developer sandbox is so that he or she can have the freedom and safety to undertake code changes even when it is risky. In addition to these individual sandboxes, organizations should keep one sandbox called the build sandbox. The nightly build process will take place on the build sandbox.

None of the files in the developer sandbox should be checked out for extended

A LIMITED TIME SAVINGS OFFER FROM SYS-CON MEDIA

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

AND SAVE UP TO \$340 AND RECEIVE UP TO 3 FREE CDs!



RECEIVE
YOUR DIGITAL
EDITION
ACCESS CODE
INSTANTLY
WITH YOUR PAID
SUBSCRIPTIONS

3-Pack

Pick any 3 of our magazines and save up to **\$210⁰⁰**
Pay only \$99 for a 1 year subscription plus a **FREE CD**

- 2 Year - \$179.00
- Canada/Mexico - \$189.00
- International - \$199.00

6-Pack

Pick any 6 of our magazines and save up to **\$340⁰⁰**
Pay only \$199 for a 1 year subscription plus 2 **FREE CDs**

- 2 Year - \$379.00
- Canada/Mexico - \$399.00
- International - \$449.00

9-Pack

Pick 9 of our magazines and save up to **\$270⁰⁰**
Pay only \$399 for a 1 year subscription plus 3 **FREE CDs**

- 2 Year - \$699.00
- Canada/Mexico - \$749.00
- International - \$849.00

CALL TODAY! 888-303-5282

Pick a 3-Pack, a 6-Pack or a 9-Pack

<input type="checkbox"/> 3-Pack	<input type="checkbox"/> 1YR	<input type="checkbox"/> 2YR	<input type="checkbox"/> U.S.	<input type="checkbox"/> Can/Mex	<input type="checkbox"/> Intl.
<input type="checkbox"/> 6-Pack	<input type="checkbox"/> 1YR	<input type="checkbox"/> 2YR	<input type="checkbox"/> U.S.	<input type="checkbox"/> Can/Mex	<input type="checkbox"/> Intl.
<input type="checkbox"/> 9-Pack	<input type="checkbox"/> 1YR	<input type="checkbox"/> 2YR	<input type="checkbox"/> U.S.	<input type="checkbox"/> Can/Mex	<input type="checkbox"/> Intl.

TO
ORDER

• Choose the Multi-Pack you want to order by checking next to it below. • Check the number of years you want to order. • Indicate your location by checking either U.S., Canada/Mexico or International. • Then choose which magazines you want to include with your Multi-Pack order.

LinuxWorld Magazine

U.S. - Two Years (24) Cover: \$143	You Pay: \$79.99 /	Save: \$63 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 /	Save: \$32
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$79.99 /	Save: \$4
Intl - Two Years (24) \$216	You Pay: \$175 /	Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

JDI

U.S. - Two Years (24) Cover: \$144	You Pay: \$99.99 /	Save: \$45 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$69.99 /	Save: \$12
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$89.99 /	Save: \$40
Intl - Two Years (24) \$216	You Pay: \$175 /	Save: \$40 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

Web Services Journal

U.S. - Two Years (24) Cover: \$160	You Pay: \$99.99 /	Save: \$60 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$186	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$92	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$175 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

NET Developer's Journal

U.S. - Two Years (24) Cover: \$168	You Pay: \$99.99 /	Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84	You Pay: \$69.99 /	Save: \$14
Can/Mex - Two Years (24) \$192	You Pay: \$129 /	Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96	You Pay: \$89.99 /	Save: \$6
Intl - Two Years (24) \$216	You Pay: \$175 /	Save: \$46 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

Information Storage + Security Journal

U.S. - Two Years (24) Cover: \$143	You Pay: \$49.99 /	Save: \$93 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 /	Save: \$39
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$49.99 /	Save: \$34
Intl - Two Years (24) \$216	You Pay: \$89.99 /	Save: \$126 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$39.99 /	Save: \$48

Wireless Business & Technology

U.S. - Two Years (12) Cover: \$129	You Pay: \$49.99 /	Save: \$71 + FREE \$198 CD
U.S. - One Year (6) Cover: \$69	You Pay: \$29.99 /	Save: \$39
Can/Mex - Two Years (12) \$129	You Pay: \$69.99 /	Save: \$51 + FREE \$198 CD
Can/Mex - One Year (6) \$69	You Pay: \$49.99 /	Save: \$10
Intl - Two Years (12) \$129	You Pay: \$69.99 /	Save: \$20 + FREE \$198 CD
Intl - One Year (6) \$72	You Pay: \$29.99 /	Save: \$2

MX Developer's Journal

U.S. - Two Years (24) Cover: \$143	You Pay: \$49.99 /	Save: \$93 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72	You Pay: \$39.99 /	Save: \$32
Can/Mex - Two Years (24) \$168	You Pay: \$119.99 /	Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84	You Pay: \$49.99 /	Save: \$34
Intl - Two Years (24) \$216	You Pay: \$89.99 /	Save: \$126 + FREE \$198 CD
Intl - One Year (12) \$108	You Pay: \$39.99 /	Save: \$48

ColdFusion Developer's Journal

U.S. - Two Years (24) Cover: \$216	You Pay: \$129 /	Save: \$87 + FREE \$198 CD
U.S. - One Year (12) Cover: \$108	You Pay: \$89.99 /	Save: \$18
Can/Mex - Two Years (24) \$240	You Pay: \$159.99 /	Save: \$88 + FREE \$198 CD
Can/Mex - One Year (12) \$120	You Pay: \$99.99 /	Save: \$20
Intl - Two Years (24) \$264	You Pay: \$189 /	Save: \$75 + FREE \$198 CD
Intl - One Year (12) \$132	You Pay: \$129.99 /	Save: \$2

WebSphere Journal

U.S. - Two Years (24) Cover: \$216	You Pay: \$129.00 /	Save: \$87 + FREE \$198 CD
U.S. - One Year (12) Cover: \$108	You Pay: \$89.99 /	Save: \$18
Can/Mex - Two Years (24) \$240	You Pay: \$159.99 /	Save: \$88 + FREE \$198 CD
Can/Mex - One Year (12) \$120	You Pay: \$99.99 /	Save: \$20
Intl - Two Years (24) \$264	You Pay: \$189.99 /	Save: \$75
Intl - One Year (12) \$132	You Pay: \$129.99 /	Save: \$2

PowerBuilder Developer's Journal

U.S. - Two Years (24) Cover: \$363	You Pay: \$169.99 /	Save: \$193 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$363	You Pay: \$179.99 /	Save: \$183 + FREE \$198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Intl - Two Years (24) \$363	You Pay: \$189.99 /	Save: \$170 + FREE \$198 CD
Intl - One Year (12) \$180	You Pay: \$179 /	Save: \$1

WLDJ

U.S. - Four Years (24) Cover: \$240	You Pay: \$89.99 /	Save: \$140 + FREE \$198 CD
U.S. - Two Year (12) Cover: \$120	You Pay: \$49.99 /	Save: \$70
Can/Mex - Four Years (24) \$240	You Pay: \$89.99 /	Save: \$140 + FREE \$198 CD
Can/Mex - Two Year (12) \$120	You Pay: \$69.99 /	Save: \$50
Intl - Four Years (24) \$240	You Pay: \$129 /	Save: \$120 + FREE \$198 CD
Intl - Two Year (12) \$120	You Pay: \$79.99 /	Save: \$40

WHILE SUPPLIES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE.

Subscribe Online Today www.sys-con.com/2001/sub.cfm

**SYS-CON
MEDIA**

“Source control systems are so important that, without a properly configured system, quality software cannot be made”

periods of time. Once a developer is finished writing a feature, he or she should delete all files from his developer sandbox and then shadow a new copy. This way, the sandbox stays in synch with the source of the application.

Before checking any code into the source control system, it is vitally important that the code be compiled first. Developers should never check in code that has not been compiled. The moment code is written, developers should resolve all compilation errors and compile-level warnings.

The build sandbox should be cleanly shadowed (i.e., receive read-only copies of the master files stored in the source control system) from the source control system and deleted on a daily basis.

These principles may sound obvious, or even naive, but it is surprising to find that many software organizations lack a clear policy on what is allowed to be checked in and out of their source control systems.

Nightly Build, Deployment, and Test Process

Once a source control system is in place, the next step is to establish an automated nightly build, deployment, and test process. The main reason for establishing these processes is to be able to monitor the progress of your development. The results from the nightly build process tell you whether there are any incompatible changes in the application components. A side benefit to having a nightly build process is that, in cases where a build is shipped or released, you already have a tested build process that has been running all along. The nightly deployment process serves to set up a context in which a set of tests can be run that verify your Web service. This enables you to run the nightly test process, which, in turn, tells you whether

the Web service continues to run as expected even as it continues to grow and change.

Nightly Build Process

A separate build computer (different from the computers used by developers) should be designated for the nightly build process. A separate computer is used because configurations and system settings on a developer machine can sometimes hide various dependency errors.

Each night, all of the source code and related files should be shadowed from the source control system. A scheduled task should initiate the build script that compiles the necessary components and builds the application. The results of the nightly build process must be monitored each morning. If for any reason the build process fails, the failure must be investigated and resolved so that the build process succeeds and the following processes can be performed.

Nightly Deployment Process

If the nightly build process succeeds, the nightly deployment process should be launched. This process should also be automated via a script that is initiated by a scheduled task after the nightly build process succeeds. The deployment consists of the WSDL, server, database, client, and proxy server.

First the WSDL should be created from the latest source code and exposed on a port on the same machine that the build process ran on. The WSDL should reference the most recent versions of the schemas. The application should then be exposed as a Web service on the same machine. This machine, which should be the server, should be accessible on the network and should have a reliable connection to the

database. The database should be set to its default configuration during this process. The Web service client, when applicable, should be created from the latest source code and deployed. The nightly deployment process should also be monitored each night so that any errors can be detected and fixed right away.

Nightly Test Process

Finally, a nightly test process must be implemented. In this process, the newly built application, WSDL, Web service, and client are automatically tested to verify that they satisfy all the requirements and that no regressions occur in the functionality. All the test cases should be shadowed from the source control and run. Any failures must be reported and monitored the next morning. In this way, a feedback loop is established whereby errors are detected and fixed as soon as they are introduced.

Conclusion

In order to develop fully functional, robust Web services, you must begin with the basic principles of creating and effectively using your source control system and automatic build process. For the example used here, we took a systematic approach to creating a Web service. At each step, we defined requirements and froze them once they were met. We were able to do this through creation of verification tests that were incorporated into source control and the nightly build process. By incorporating these fundamental building blocks throughout your entire development cycle, you will be able to confidently say that you created a bulletproof Web service. ©

About the Author

Dr. Adam Kolawa is cofounder and CEO of Parasoft, a leading provider of Automated Error Prevention software solutions. Dr. Kolawa, co-author of *Bulletproofing Web Applications* (Hungry Minds, 2001), has contributed to and written over 100 commentary pieces and technical articles for publications such as *The Wall Street Journal*, *CIO*, *Computerworld*, *Dr. Dobbs' Journal*, and *IEEE Computer*. He has also authored numerous scientific papers on physics and parallel processing. His recent media engagements include CNN, CNBC, the BBC, and NPR. Dr. Kolawa holds a PhD in theoretical physics from the California Institute of Technology, and has been granted 10 patents for his recent inventions.

■■■ ak@parasoft.com

This Month

Integrating XML

BY GARY KUPECZ

The way XML has been positioned over the last several years – namely, as a savior for companies that have invested in myriad systems with numerous incompatible data types – makes you wonder why organizations aren't adopting any XML solution they can get their hands on.

Generating XML from Relational Database Tables

BY SELIM MIMAROGLU

This article looks in detail at how to generate XML data from your relational database. Although the examples were run on Oracle, very little of the code is



Oracle specific. You can easily use all the ideas and examples presented here in other relational databases. We did this project at the University of Massachusetts–Boston as part of the Electronic Field Guide (EFG) project.

Integrating XSL-FO into Web Based Applications

BY ADELENE NG

This article demonstrates how we can integrate XSL-FO, XSLT, and JavaMail into our existing Web-based applications.



I show you how we can generate PDF reports for an application through the use of XSLT and XSL-FO embedded within the Java application. I also illustrate how the generated PDF file can be sent as an e-mail attachment using JavaMail.



Pg.48

XML-Based Interop, Close Up

In addition to the strategy side of Web services, there is also the protocol-oriented side of things, the XML side. Embracing not only XML itself but also the full range of mainstream XML-based technologies like XPath, XSLT, XML Schema, and SOAP, *XML-Journal* has been delivering insightful articles to the world of developers and development managers since the year 2000.

It is our privilege to bring *XML-Journal* directly to readers of *Web Services Journal*, and vice versa. Anyone already familiar with the Web services world of SOAP, UDDI, and WSDL will find here articles and features each month that will interest them – about the cutting-edge technologies and latest products that are changing not only our industry, but the way the world exchanges information. To make it easy for you to find your way around, we have four distinct sections:

Content Management:

Organization, dissemination, and presentation of information

Data Management:

Storage, transformation, representation, and general use of structured and unstructured data

Enterprise Solutions:

Systems and applications that manage mission-critical functions in the enterprise

Labs:

Product reviews, book reviews, tutorials, and standards analysis



Integrating XML

WRITTEN BY
GARY KUPECZ

The executive initiative

The way XML has been positioned over the last several years – namely, as some type of savior for companies that have invested in myriad systems with numerous incompatible data types – it makes you wonder why organizations aren't adopting any XML solution they can get their hands on.

But for companies that have been burned once too often by the promise of the next "can't-miss" technology, it's easy to understand the reluctance many might have toward XML. Yes, it's true, XML has been a hyped technology. The question becomes, has it earned the right to hype?

A Lot to Like About XML

Let's consider why XML is being positioned as a so-called savior. The most uttered praise around XML is its adaptability – by encoding data in XML, organizations can quickly make information available in a simple and usable format, thus allowing for greater interoperability between previously incompatible systems. Also, because information content is separated from information rendering, it is much easier to provide multiple views of the same data. In short, XML is designed for data representation that is simple and easy to use.

Moreover, the attraction to XML is the ease with which it handles legacy data. This point cannot be overemphasized. After all, companies have spent years and millions of dollars developing legacy infrastructures. More often than not, these mainframe-based systems cannot "talk" to each other, so trading data becomes a frustrating exercise. The problem is, these legacy systems are often business critical.

In a perfect world, companies would rewrite their legacy applications to run natively in our increasingly Internet-centric world. Of course, in that utopian view they would also have a blank check in hand to pay for that rewriting – not to mention the considerable time to undertake such a project. But time-to-market is critical today, which is why companies are turning to XML middleware as a way to quickly transform old data for use in new applications.

It's no longer really about IT – it's about finding solutions that will solve critical business needs. Companies are grappling with a multitude of noncompatible standards – some

old, some new, but almost all evolving – combined with a multiplicity of IT disciplines. This breeds frustrating complexity and, most importantly, added cost to the bottom line.

It's the Industry

Every vertical industry comes with its own set of unique challenges: everything from compliance to industry-specific standards – such as HIPAA regulations for health care – to partners who refuse to conduct business unless one can electronically "talk" in their standard. While XML may not be the panacea, it will go a long way to overcoming most of these obstacles.

Unlike most legacy data, XML is self-describing – organizations can create a more human, readable data format (as opposed to data that is solely machine readable). From a system integration point of view, that's critical. For example, take a common challenge with legacy systems that might be 20, 40, 60 years old. Instead of finding the original programmers to perform the data translation, today's data analysts perform gap analysis between different data formats using XML as the middleware layer.

"The question becomes, has it earned the right to hype?"

Add to these horizontal issues the unique challenges native to a particular industry. Insurance companies are under constant pressure to drive down costs and streamline processes to sharpen customer focus and reach out to target markets with competitive products. Complicating the problem is the fact that these companies often own a mix of incompatible systems, applications, and databases that store information in a variety of formats, including COBOL, AL3, flat files and several proprietary data formats.

This makes automation complex and costly. But the implications are huge. After all, if a company can take a dollar off every transaction – or even 10% off every transaction – that's a huge impact on the bottom line.

Enter XML. XML adoption is becoming widespread in the insurance industry among vendors, institutions, and other organizations adopting the ACORD standards. Insurance companies are exploring ways to use ACORD forms and XML to provide a standard vocabulary to facilitate communication among business systems. For organizations that are considering ACORD forms and XML, the key is to understand how the technology works and the specific business benefits to be gained. By adopting the new standards via XML, organizations can share data with trading partners consistently, accurately, and instantaneously, as well as retain and expand customer relationships.

Compliance with industry standards is also a huge issue. Using XML, an organization can store metadata or specific data that's required for compliance in an XML repository. Also, there's a rapid movement to technology that's being developed by software companies to enable enhanced search functionality using XML.

For example, a financial organization can file budget statements in a searchable XML repository. This allows searches down to the field level or metadata tag in XML – as opposed to creating some sort of proprietary format in the database and having to deal with referential integrity. As a result, if you have specific schemas associated with select data, you can quickly get to the information.

XML has benefits in other areas, such as customer relationship management. Most CRM systems have their own proprietary formats. If an organization wants to trade information between those two systems, XML is a logical technology to allow them to “talk” to each other. Even these large software companies see the value of XML as an integration layer and are adopting Web services or adapters to the core business processes of their applications.

Once a company recognizes that XML is a viable platform for their organization, enter the next challenge: how to implement it with minimal disruption. From the outset, one of the biggest obstacles is making the decision between using a standards-based or proprietary schema. There are advantages and disadvantages to both – trade-offs, in other words. Early on, there was a tendency to go the proprietary route, simply given the nature of the standards' world: changes to standards were slow since decisions were made by committee. As a result, many organizations came out with proprietary schemas because there wasn't time to wait for a standard. But the groundswell for standards is strong, particularly since they attempt to harmonize all the hard efforts of individual companies and people to adopt comparable technologies.

Often what's needed – and this is a challenge that's outside the technical realm, but one that is just as critical – is someone within the organization to take the bull by the horns and say, “This is the direction we need to go in.”

Vendor Q&A

A company looking to adopt XML should ask its vendor several questions. Most importantly, does the vendor know the industry? Companies should feel confident that the chosen vendor understands their particular business challenges – their pain – and that the technology solves that pain in an organization's particular infrastructure.

Does the vendor know the challenges of the legacy for-


mat being presented? For example, the insurance industry often uses AL3 data and COBOL data (the EDI of insurance). While an XML or middleware vendor might position itself as having the best technology available, this won't mean anything if the vendor doesn't know how the company or its industry uses that specific data. The vendor needs to understand the business processes of the industry's data.

A challenge for any organization is recognizing the trade-off between the value of an IT solution and its price – the return on investment. An XML solution is no exception. Ultimately, the value of a solution comes down to a simple, bottom-line answer to the question: will the solution do what we require?

There are a few things an IT department should know when it comes to the technical challenges of implementing an XML solution. For one, it's important to have a solution that is not intrusive on existing systems. The old adage of “don't break something that's already working” holds true. Don't put a square peg into a round hole. Other technical challenges include ensuring that the solution is adaptable across the enterprise and that there's adequate support for integration of other solutions. After all, no application today exists in a vacuum. You should also pay attention to automation needs – being able to schedule business-process rules and automating pull-push data – and to whether the solution can scale up to your trading partners. And, of course, you can never forget about security.

“Companies should feel confident that the chosen vendor understands their particular business challenges”

A good way to get one's feet wet with XML is to start small – risk something that may not be strategic to the organization. For example, export a content management software system into XML. It's important to understand the technology – as you better understand the functionality, it becomes easier to add to it.

In the ever-changing world of business, companies are faced with the hard fact that people and their systems need to communicate in a more efficient manner. Is XML a panacea? No single technology solution will solve the entirety of an organization's business challenges. But the benefits of adopting XML – allowing improved system interoperability and greater information sharing, not to mention its potential ROI gains – will go a long way to addressing key organizational pain points. So don't necessarily consider XML a savior – but it's certainly a good, dependable friend. 

AUTHOR BIO

Gary Kupecz is director of QoXML product marketing with Xenos Group Inc. (www.xenos.com) – the data to e-content company. With over 12 years of enterprise software product development and consulting experience, he has successfully identified new trends in the market and developed infrastructure solutions that effectively address today's business needs.

 GKUPECZ@XENOS.COM



WRITTEN BY SELIM MIMAROGLU

Generating XML from Relational Database Tables

Prepare for the future

This article looks in detail at how to generate XML data from your relational database. Although the examples were run on Oracle, very little of the code is Oracle specific. You can easily use all the ideas and examples presented here in other relational databases. We did this project at University of Massachusetts Boston as part of the Electronic Field Guide (EFG) project.

XML is the de facto standard for data exchange. It's simple, Unicode based, and platform independent. XML is a metadata language; it contains information about the data. All these features make it an attractive standard for exchanging data.

Why Generate XML from Relational Data?

Today most data (80% or more), is stored in relational databases such as Oracle, DB2, SQL Server 2000, and others. The Internet and Web services are present in our daily lives. A tremendous amount of data is transferred over the Internet between businesses. Data transfers may happen within a company, between different branches, or between different companies and individuals. No matter which of these situations applies to you, it's very likely that you will be asked to ship your data in XML, or that you are already doing it. Relational Database Management Systems (RDBMS) are still the best way to hold data in bulk. In the following paragraph we will give some information about our project and related resources that we used to provide relational data in XML.

Our Project

Our XML project is part of the Electronic Field Guide (EFG). The EFG project is an object-oriented, Web-based database for the identification of species and recording of ecological observations. With funding from the National Science Foundation, this project is the result of the collaborative efforts between the Departments of Computer Science and Biology at the University of Massachusetts Boston. EFG queries the Integrated Taxonomic Information System (ITIS). On each query we only get a very small part of the ITIS database. XML can describe exactly what data it is delivering, and thus is the preferred format for the query results.

There are three ITIS branches: Canadian, Mexican, and U.S. U.S. ITIS provides the whole data in bulk (about 85MB), but, like many other organizations, it has been slow to join the modern trend to answer queries on it in XML format. Canadian ITIS provides query results in XML, but due to networking delays and some other problems, getting the result takes a long time. We decided to bring the whole database home and return the results of queries in XML format. We are using

an RDBMS to store the bulk data. In the following paragraphs we will explain how to generate XML from this relational data.

SQL/XML

This emerging standard enables us to generate XML fragments from relational data. Oracle and many other relational databases support the following standard SQL/XML functions. We will explain most of the SQL/XML functions, XMLElement, XMLAttributes, XMLForest, XMLAgg, and give an example for each.

XMLElement

As its name implies, this function generates an XML element. It takes an element name, an optional collection of attributes for the element, and zero or more arguments that make up the element content and returns an instance of type XMLType.

Table 1 is the schema of the "experts" table from our database. All the simple examples are related to this table, and Oracle was used for most cases.

Table 2 shows some of the data from the experts table.

in ORACLE:
SQL> SELECT

Name	Null?	Type
EXPERT_ID_PREFIX	NOT NULL	CHAR(3)
EXPERT_ID	NOT NULL	NUMBER(38)
EXPERT	NOT NULL	VARCHAR2(100)
EXP_COMMENT		VARCHAR2(255)
UPDATE_DATE	NOT NULL	DATE

Table 1 • Table schema

AUTHOR BIO

Selim Mimaroglu is a PhD candidate in computer science at the University of Massachusetts in Boston.

He holds an MS in computer science from that school and also has a BS in electrical engineering.

expert_id_prefix	expert_id	expert	expert_comment	update_date
EXP	4	Stotler, Raymond E.		26-JUL-01
EXP	5	Alfred L. Gardner	Curator of North American mammals and Chief of Mammal Section, National Biological Service, Smithsonian Institution	30-DEC-02
EXP	6	Steve J. Upton	Professor Division of Biology, Ackert Hall Kansas State University Manhattan	28-APR-98
EXP	8	Wayne Starnes	North Carolina State Museum of Natural Science	05-MAY-03
EXP	10	Lynne Parenti	Curator of Fishes, Smithsonian Institution	16-SEP-98

Table 2 • Data from the experts table

```

XMLELEMENT("name", expert)
FROM experts
WHERE expert_id BETWEEN 4 and 10;

```

```

in DB2:
SELECT
XML2CLOB( XMLELEMENT(NAME "name",
expert) )
FROM experts
WHERE expert_id BETWEEN 4 and 10

```

```

XMLELEMENT("NAME",EXPERT)
-----
<name>Stotler, Raymond E.</name>
<name>Alfred L. Gardner</name>
<name>Steve J. Upton</name>
<name>Wayne Starnes</name>
<name>Lynne Parenti</name>

```

Here, name is the tag name; expert is the corresponding column name in the experts table. This query obtains the expert column value from the experts table and puts <name> and </name> tags around it.

XMLAttributes

This function simply creates attributes for an element. Listing 1 shows how it works.

Each name element has an id attribute. Each id attribute is obtained from the expert_id column value of the experts table. The XMLAttributes clause isn't used alone; it's used inside an XMLElement function. This makes sense since, in XML, an attribute is a name-value pair attached to the associated element's start tag.

XMLForest

The XMLForest() function produces a

forest of XML elements from the given list of arguments. In other words, it produces many XML elements at a time.

In Listing 2, expert is the root element. The id, name, and info elements are children of expert element. The XMLForest function created three elements: the id element corresponds to expert_id column value; the name element corresponds to expert column value; and the info element corresponds to the exp_comment column value in experts table. Naming the elements in this function is accomplished by using AS keyword. For the first element it is expert_id as "id".

What if you don't specify an id (id = 6 in this example)? In this case you will have the appropriate XML fragment for each row.

XMLAgg

XMLAgg() is an aggregate function that produces a forest of XML elements derived from a set of rows.

In Listing 3, there are five name elements as children of the experts root element. Without using XMLAgg() function it's impossible to have many name elements in a single XML document. If you don't use the XMLAgg() function and instead submit the query, you will get Listing 4.

That's not what we wanted. For this output, each experts element has only one name child element, and there are many experts elements. To combine information from multiple rows of the table we need to use the XMLAgg() function.

Creating hierarchical data is easy. Listing 5 shows you how to do this. You

can use this idea in similar queries. This example involves the taxonomic_units table, which stores data about species, phyla, etc., i.e. nodes in the tree of life. Note the select within the select, to run through the inner loop.

I hope these examples have convinced you that it's possible to generate XML that obeys any XML Schema or DTD. You can use these functions, nested in each other, to display any kind of parent-child relation. Another such query could display all the experts for each taxon.

This is easy! You can start using these functions as soon as you finish reading this article. Being flexible adds value to this approach. An important point that's worth mentioning is that all the work is done inside the database by using the SQL/XML functions. In the Oracle case, all the work is done in the XML DB Engine. This is much faster and more efficient than doing the work outside the database. Another approach would be to get the data from the database and tag it outside the database for creating XML. This is possible but less efficient, more time consuming. The following are more advanced topics, such as XMLType View and XML Schema validation.

XMLType View

A view is a table that results from a subquery, but which has its own name and can be treated in most ways as if it were an ordinary table. Thus a view table is a logical window on selected data from the base tables and other views. XMLType views wrap the relational data in XML formats. In other words you can have a

virtual XML file over your relational data inside the database. We can treat this new view as XML, and use XML specific operations on it, such as XPath and XQuery. These types of operations will be converted to corresponding SQL; this is known as query rewrite. You can create this view by using SQL/XML functions. Our SQL, which generates an XMLType view is shown in the source code, create.sql, online at www.syscon.com/xmlj/sourcecec.cfm. Below is a simpler example, for better understanding.

```
SQL> CREATE OR REPLACE VIEW
expert_view of XMLTYPE WITH OBJECT ID
(EXTRACT(sys_nc_rowinfo$, '/experts/expert/@id').getnumberval()) AS
SELECT XMLELEMENT("experts",
XMLAGG( XMLELEMENT("expert", XMLATTRIBUTES( expert_id as "id", expert)))
FROM experts
WHERE expert_id BETWEEN 4 and 10;
```

View created.

This is how the view looks when you query it. This XMLType view is created over the experts table. The data displayed comes from the underlying relational table.

```
SQL> SELECT * FROM expert_view;

SYS_NC_ROWINFO$

<experts>
  <expert id="4">Stotler, Raymond
E.</expert>
  <expert id="5">Alfred L.
Gardner</expert>
  <expert id="6">Steve J.
Upton</expert>
  <expert id="8">Wayne Starnes</expert>
  <expert id="10">Lynne
Parenti</expert>
</experts>
```

It's possible to use XPath expressions on this view. Following is a query that has an XPath expression. This query extracts the value of an expert element that has an id attribute equal to 4.

```
SQL> SELECT extract(value(x),
'/experts/expert[@id=4]') FROM
```

```
expert_view x;
EXTRACT(VALUE(X), '/EXPERTS/EXPERT[@ID=4]
')
```

```
<expert id="4">Stotler, Raymond
E.</expert>
```

First Step: Register an XSD

You can validate an XMLType instance against XML Schema Documentation (XSD) in Oracle. First, register the XSD in Oracle. Next, call a function that does the validation. There are several functions in Oracle for accomplishing this. Listing 6 is the only one shown, for simplicity.

This will register the XSD and name it as expert_view.xsd. You can name it anything you want of course.

Second Step: Validation

This section introduces isSchemaValid (argument1, argument2). The first argument is the name of the registered XSD. The second argument is the name of the root element in the specified XSD. XSD can have more than one root. If the XML document (x in this case) is valid, this isSchemaValid returns 1.

```
SQL> select
x.isSchemaValid('expert_view.xsd',
'experts') from expert_view x;
```

```
X.ISSCHEMAVALID('EXPERT_VIEW.XSD','EXPERTS')
1
```

Putting It Together

For our project we generated XML that obeys our XML Schema Documentation (XSD)[itis.xsd]. Our SQL is shown in the source code for this article (available online at www.sys-con.com/xmlj/sourcecec.cfm). It will output all the data in XML. As you can see, this query is significantly nested and long, but it's easy to understand (I hope). There are a few more functions in this query that we would like to mention: trim() simply gets rid of the white space in an entry and the to_char() function is used to change the date format. For example, to_char(v.up date_date, 'YYYY-MM-DD') will display the date as something like 2004-04-23. This is somewhat important because the date format in the database doesn't match the date format of the XML Schema.

XML Delivery

In this project we deliver XML through a search server. Using this server, you can query the database by tsu, which is identical to the identification number of a taxonomic unit. e.g. a particular animal or plant. For this part of the project we used:

1. Sun Enterprise 250 Server: 2 X UltraSPARC-II 400MHz, 2GB memory, SCSI disks
2. Sun Solaris Operating System 5.8
3. Oracle 9.2i
4. JAVA 1.4.2
5. Java Database Connectivity (JDBC) inside JavaBeans for connecting to and querying the database. We preferred the Oracle thin driver to the oci driver.
6. JavaServer Pages (JSP) at the user interface level
7. Tomcat 5.0.12

JSP gets a request from the user, and passes it to a JavaBean. The JavaBean connects to Oracle using JDBC, gets information from the database, and passes it back to the JSP. For better performance:

- Use Oracle Connection Pool
- Turn off the auto commit feature
- Use predefined column types in the select statement.

Conclusion

If you are not already doing so, you will probably be delivering your data in XML format soon. Keep in mind that most data resides in relational databases so generating XML from relational databases will be a very common task. Many of the database vendors, such as Oracle and DB2, provide the SQL/XML function support to make this task easier. SQL Server 2000 adds a new clause, the FOR XML clause, to the SELECT statement, which instructs SQL Server to return the result of a query in XML format.

Based on our experiments, generating XML data inside the database is fast; it took an average of 0.032 seconds per taxon query. Storing your data without the start and end tags is space efficient. Relational databases are very mature about storing, querying, and retrieving data quickly and efficiently. There has been over 20 years of work done in these technologies.

For larger projects, having XMLType views over the relational tables is advanta-

geous. It gives you the chance to query the data using XPath and XQuery besides SQL.

In this article we provided some insight about SQL/XML functions, mentioned the benefits of storing data in relational databases, and explored the benefits of using XMLType views over relational data. For more detailed information about SQL/XML functions, check your database documentation.

References

- O'Neil, Patrick and Elizabeth. (2000) *Database: Principles, Programming, Performance*. Morgan Kaufman.
- and "Electronic Field Guide: An Object-Oriented WWW Database to Identify Species and Record Ecological Observations" www.cs.umb.edu/efg
- Oracle9i Database Online Documentation <http://otn.oracle.com/>

[pls/db92/db92.homepage](http://pls.db92/db92.homepage)

- Appelquist, Daniel K. (2001). *XML and SQL: Developing Web Applications*. Addison-Wesley.
- Katz, Howard; and Chamberlin, D.D. (2003) *XQuery from the Experts: A Guide to the W3C XML Query Language*. Addison-Wesley.

SMIMAROG@CS.UMB.EDU

LISTING 1. XML Attributes

```
SQL> SELECT
XMLELEMENT("name", XMLATTRIBUTES( expert_id as "id",
expert) FROM experts
WHERE expert_id BETWEEN 4 and 10;

XMLELEMENT("NAME",XMLATTRIBUTES(EXPERT_ID AS "ID"),EXPERT)

<name id="4">Stotler, Raymond E.</name>
<name id="5">Alfred L. Gardner</name>
<name id="6">Steve J. Upton</name>
<name id="8">Wayne Starnes</name>
<name id="10">Lynne Parenti</name>
```

LISTING 2. XMLForest

```
SQL>SELECT
XMLFOREST("expert",
XMLFOREST(expert_id as "id",
expert as "name",
exp_comment as "info")) as "result"
FROM experts
WHERE expert_id = 6;

result

<expert>
  <id>6</id>
  <name>Steve J. Upton</name>
  <info>Professor Division of Biology, Ackert Hall Kansas
    State University Manhattan</info>
</expert>
```

LISTING 3. XMLAgg

```
SQL>SELECT
XMLELEMENT("experts",
XMLAGG( XMLELEMENT("name", expert)))
FROM experts
WHERE expert_id BETWEEN 4 and 10;

XMLELEMENT(" EXPERTS",XMLAGG(XMLELEMENT("NAME",EXPERT)))

<experts>
  <name>Stotler, Raymond E.</name>
  <name>Alfred L. Gardner</name>
  <name>Steve J. Upton</name>
  <name>Wayne Starnes</name>
  <name>Lynne Parenti</name>
</experts>
```

LISTING 4. Result of not using XMLAgg

```
SQL>SELECT
XMLELEMENT("experts",
XMLELEMENT("name", expert))
FROM experts
WHERE expert_id BETWEEN 4 and 10;

<experts>
  <name>Stotler, Raymond E.</name>
</experts>

<experts>
  <name>Alfred L. Gardner</name>
</experts>
```

LISTING 5. Creating hierarchical data

```
SQL> SELECT
XMLELEMENT("taxon", XMLATTRIBUTES(tsn as "tsn"),
XMLELEMENT("name", trim(unit_name1)),
```

```
(SELECT XMLELEMENT("children",
XMLAGG(
XMLELEMENT("taxon",XMLATTRIBUTES(child.tsn as "tsn"),
XMLELEMENT("name", trim(child.unit_name1))))
FROM taxonomic_units child
WHERE child.parent_tsn = prnt.tsn))
FROM taxonomic_units prnt
WHERE prnt.tsn < 10000;
```

```
.
.
<taxon tsn="9993">
  <name>Plectadinium</name>
  <children/>
</taxon>

<taxon tsn="9994">
  <name>Gymnodiniales</name>
  <children>
    <taxon tsn="9996">
      <name>Gymnodiniaceae</name>
    </taxon>
    <taxon tsn="10127">
      <name>Warnowiaceae</name>
    </taxon>
  </children>
</taxon>
```

LISTING 6. First Step: Register an XSD

```
BEGIN
dbms_xmlschema.registerSchema('expert_view.xsd',
'<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XMLSPY v5 rel. 4 U
(http://www.xmlspy.com)-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="expert">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension
          base="xs:string"
          <xs:attribute
            name="id" use=
              "required">
      </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="experts">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="expert"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
');
END;
```

Download the Code
www.sys-con.com/xml



Integrating XSL-FO into Web-Based Applications

JavaMail and PDF report creation

This article demonstrates how we can integrate XSL-FO, XSLT, and JavaMail into our existing web-based applications. I show you how we can generate PDF reports for an application through the use of XSLT and XSL-FO embedded within the Java application. I also illustrate how the generated PDF file can be sent as an e-mail attachment using JavaMail.

Although a variety of Web-based technologies such as servlets or Web services are available, I chose the JSP approach. A simple JSP-based test harness was written to demonstrate the integration of all these technologies. An HTML table report generation example is also included to show how the XSL-FO table elements correspond to the HTML table elements. Although using XSL-FO and XSLT may be overkill, the greater degree of formatting control and flexibility may prove advantageous.

Tools Used

The entire system was written in Java running on Windows 2000 Professional. The tools used to build this system include:

- **Java 2 Platform, Standard Edition (J2SE):** <http://java.sun.com/j2se/>
- **JavaMail 1.3:** <http://java.sun.com/products/javamail/>
- **JavaBeans Activation Framework 1.0.2:** <http://java.sun.com/products/javabeans/glasgow/jaf.html>
- **JDOM Beta 9:** www.jdom.org
- **Log4j v 1.2.8 from Apache:** <http://logging.apache.org/log4j>
- **FOP-0.20.5:** <http://xml.apache.org/fop/index.html>

- **Apache Tomcat 5.0.16 (JSP and Servlet Engine):** <http://jakarta.apache.org/tomcat/index.html>

Formatting Objects Processor (FOP)

XSL-FO is part of the Extensible Stylesheet Language (XSL) family of recommendations from the W3C. XSL is used to define XML document transformations and presentations and is made up of:

- XSLT/XPath
- XSL-FO

XSLT is a language for transforming XML documents (mainly from XML to XML or XML to HTML).

XPath allows you to identify specific parts of your XML document and to write expressions to refer to, for example, the *n*th child element of the specified XML file. It is used extensively by XSLT for referencing specified elements within the input document for further processing.

XSL-FO is an XML language that defines page formatting and layout.

FOP is an implementation of the XSL-FO specification defined by W3C. It is both an open source library and an appli-

cation used to convert your XML documents into paginated output. FOP supports a number of different output formats, such as PDF, Postscript, PCL, and text.

Why XSL-FO?

XSL-FO, in particular Apache FOP, was used because FOP allows for easy conversion from XML to PDF. The formatting commands are not embedded into the Java application and are stored in a separate file. This means I can easily change the "look" of the resulting document. Also, XSL-FO may be elevated to a W3C standard in the near future.

Although there are a number of free PDF libraries (non-XSL-FO) available, such as PJ by Ethymon (www.ethymon.com/epub.html) and retepPDF (www.retep.org.uk/retep/home.do), I have chosen to compare iText (www.lowagie.com/iText) to Apache FOP because iText has been around much longer and is a popular package. Table 1 highlights some of the differences between iText and Apache FOP.

System Architecture

I have adopted an *n*-tier architecture for this system (see Figure 1). Clients

iText	Apache FOP
Uses its own input format, iText-xml	Uses the XSL-FO specification
Used for postprocessing FOP-generated PDF files (merging, updating, and encrypting)	
Document generation is much faster for long documents	Slower for long documents
Experimental XML2PDF functionality	Fully supports XML to PDF conversion

Table 1 • iText vs. FOP

AUTHOR BIO

Adelene Ng is an independent software consultant. She holds a Ph.D. in Computer Science from the University of London, United Kingdom, and an M.Sc. from the University of Manchester, United Kingdom.

communicate with the Web server, which serves up the HTML and JSP pages. The relationship between these pages is described in detail below. The JSP page instantiates helper objects (GenStatistics, Pairs) that live on the Web server. These in turn connect to the application server, retrieving the results and storing them in

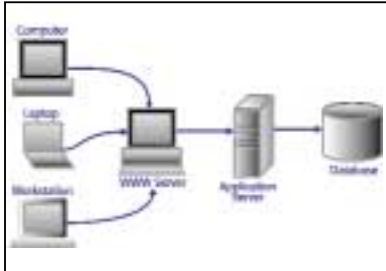


Figure 1 • System architectural overview

the helper objects. The results are then presented to the clients.

Design Overview

Figure 2 shows the relationship between the various HTML and JSP pages in the application. The start.html page is the entry point into the system. These pages are not elaborate, as their purpose is to serve as a test framework. When the "Submit" button on the start page is pressed, the genReport.jsp page is called. This checks the user selection. Depending on the type of report requested, either the genHTMLReport.jsp or genPDFReport.jsp page is invoked.

Both pages invoke the main class, GenStatistics, which connects to the application server, retrieves the data from the database, and stores the results in an array of the Pairs Bean. However, for illustration, I have removed this unnecessary complexity from the code examples and replaced it with a simple "read the data from file" illustration.

If the report is requested in HTML format, the genHTMLReport.jsp page is invoked.

If a PDF report is requested, the

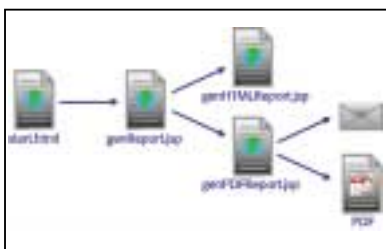


Figure 2 • Overall page flow

genPDFReport.jsp page is invoked. The results are stored in the Pairs array. This is traversed and the data is "XML-ized" using JDOM. An XSLT file containing the embedded XSL-FO commands is defined and applied to the XML-ized data, creating a FO file, which can then be passed to the Apache FOP driver to be rendered.

When the PDF report has been successfully generated, it can either be mailed out as an attachment to the specified recipient(s) or saved and displayed.

- **Supporting Data Structures, Pairs Bean:** I have defined a "Pairs" JavaBean to store the information retrieved from the application. It contains two attributes: a name (type String) and its value (type int). It contains methods for setting and retrieving attributes.
- **The Main Class, GenStatistics:** The application reads the data from the database and stores it in the "Pairs" array. When the results are returned, the data is "XML-ized." I used JDOM for this purpose. JDOM is much easier to use than DOM. It is a Java representation of a XML document.

–To use JDOM, I first create the root node,

```
Element root = new Element("statistics")
```

– Next, I create a Document object, passing in the root node:

```
Document myDoc = new Document(root)
```

–To add more nodes to the root element, I iterate through the "Pairs" array. A new Element is created for each "Pairs" element in the array:

```
Element e = new Element(pairs[i].getName());
```

–To set the text value associated with the node, I call the setText method:

```
e.setText(Integer.toString(pairs[i].getValue()));
```

– This element is added to the root node, via root.addContent.

–When the XML Document object has been populated, it will have the structure shown in Figure 3.

Finally, the transform method is called to produce the XSL-FO commands. This

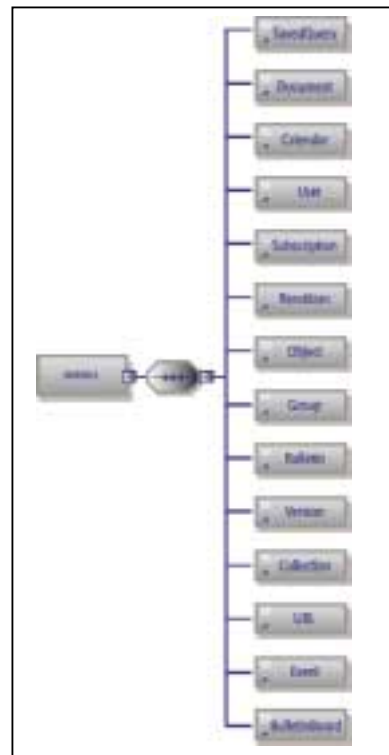


Figure 3 • XML Document Structure

takes as input the Document object, an XSLT stylesheet containing the embedded XSL-FO commands, and an FO output filename. An FO file containing the XSL-FO commands is created. The PDF is generated invoking the FOPDriver. Alternatively, the XSLT stylesheet can be

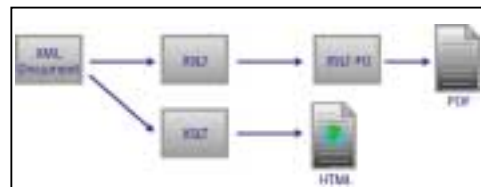


Figure 4 • Events flow

set up to transform the XML document into HTML, which is then displayed on your browser (see Figure 4).

Using the XSLT Style Sheet with XSL-FO

Although it is possible to create the XSL-FO commands by hand, it is troublesome to edit and modify it each time the XML contents change. Instead, it is easier to use an XSLT stylesheet to transform the XML data into an XSL-FO file (see Listing 1).

1. The XSLT file, dsstats.xsl, starts with the XML and namespace declaration (see lines 1 – 4).
2. A template rule is declared on line 5. This rule searches for and matches the

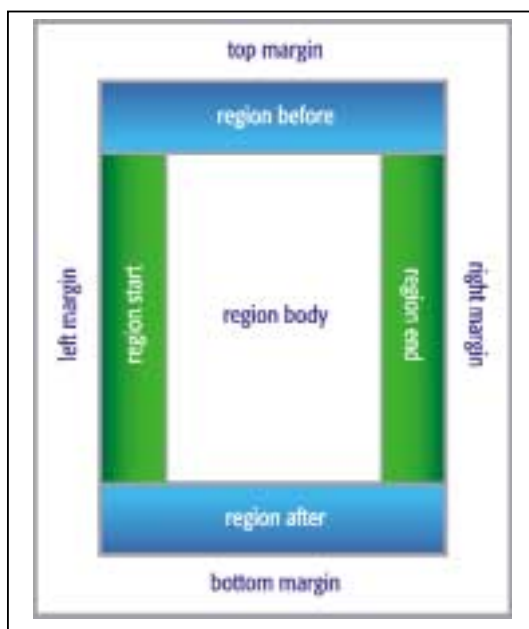


Figure 5 • Page layout and region placements in XSL-FO

root tag, replacing it with the content following it.

3. Line 6 is the root element for the XSL-FO document. This typically contains the `<fo:layout-master-set>` followed by one or more `<fo:page-sequence>` elements.
4. The `<fo:layout-master-set>` element defines the page for our document (line 7). The `<fo:simple-page-master>` element embedded within the `<fo:layout-master-set>` tag defines the page layout required for this application (lines 8–19).
5. The `page-height` and `page-width` attributes (lines 9–10) define the size of the physical page. The `master-name` attribute (line 8) declares a name for this master page. It is referenced by the `master-reference` attribute in the `<fo:page-sequence>` element (line 21). The meaning of the remaining attributes is shown in Figure 5 (lines 11–18).
6. The `<fo:static-content>` element (lines 23 and 28) allows the data occurring within these tags to appear on every page. Lines 23–30 show how the title and page number can be made to appear on every page.
7. The `<fo:block>` element is used to format paragraphs, titles, figure captions, and table titles. It can also contain raw text. In the code snippets presented here, I show an example containing raw text plus an XSL command (line 25). The XSL command matches the attribute “date” asso-

ciated with the tag “statistics”, extracting its value. The second example shows raw text plus an embedded `<fo:page-number>` command (line 29).

8. The `<fo:flow>` element (line 31) contains the actual content. This is made up of sequences of `fo:block`, `fo:block-container`, `fo:table-and-caption`, `fo:table` and `fo:list-block`. The `flow-name` attribute specifies where the flow’s content will be placed.
9. The XSL-FO `<fo:table>` command is used to generate tables. The `<fo:table>` command (line 33) is embedded within the `<fo:block>` elements (line 32). The `table-layout` attribute is set to “fixed”. This is the only option currently supported. Next, I specify the `<fo-table-column>` element (lines 34–35), setting the `column-width` attribute to a specified value.
10. The `<fo:table-body>` element (line 36) contains all the `<fo:table-row>` elements (line 37). Each `<fo:table-row>` element contains the `<fo:table-cell>` elements (line 38). This is where all the work for the tables is done. A number of properties are associated with the `<fo:table-cell>` element. This controls the table look and feel.
11. Finally, to populate the cells of the table, I use the XSL command to iterate through all the child elements contained in the XML data via `<xsl:for-each select="/statistics/*">` (line 51). For each child element of the statistics tag, get the name of the node `<xsl:value-of select="name()"/>` (line 58) and its value, `<xsl:value-of select="."/>` (line 66) and populate the table cells.
12. To ensure that the XSLT and XSL-FO commands are error free (before attempting to integrate all the components), I verified that the XSLT file produced the correct PDF output by running the files through the `fop.bat` utility:

```
fop -xml myTest.xml -xsl dsstats.xsl
-pdf myTest.pdf
```

Transforming the Document

Having created the XSLT stylesheet containing the embedded XSL-FO commands, the next step is to programmatically invoke this stylesheet within the application. The output of this transformation step produces a .FO file. This contains the XSL-FO commands and data for populating the table rows. This data was extracted from the XML document created previously. The

sequence of steps is outlined below:

1. A `StreamSource` is first created from the given XSLT style sheet File object:

```
StreamSource strmSource = new
StreamSource(styleSheetFile);
```

2. Next, create the `Transformer` object from the `TransformerFactory`, passing in an instance of the `StreamSource` object.

```
TransformerFactory transformerFactory =
TransformerFactory.newInstance();
Transformer transformer =
transformerFactory.newTransformer(strmS
ource);
```

3. Invoke the `transform` method, passing in the `JDOMSource` and `JDOMResult` as arguments. Note, the `JDOMSource` constructor takes in a `Document` object as input.

```
transformer.transform(jdSrc, jdRes);
```

4. Finally, output the resulting .FO file using the `XMLOutputter` object:

```
XMLOutputter xmlOutputter = new
XMLOutputter(" ", true);
```

This will create an `XMLOutputter` object with the specified indent (usually a number of spaces). If the second argument is true, new lines will be printed.

```
xmlOutputter.output(jdRes.getDocument()
, new FileOutputStream(fopFileName));
```

PDF Report Creation

Once the .FO file is created, I call the `FOP Driver run()` method to render the document.

1. First, set up logging. The `MessageHandler` object handles the global logging of all FOP processes. The FOP Driver handles per-instance logging. Both of these have to be set using an implementation of `org.apache.avalon.framework.logger.Logger`. I used the `Log4JLogger` implementation because existing code on the application server currently utilizes `Log4J`.
2. Create a `Log4JLogger` object and associate this with the static logger object created on start up in the `GenStatistics` class, i.e.

```
org.apache.avalon.framework.logger.Log
4JLogger Alogger = new
```

```
org.apache.avalon.framework.logger.Log
4JLogger(GenStatistics.logger);
```

3. Next, instantiate the FOP Driver.

```
Driver driver = new Driver();
```

4. Set the logger associated with the FOP Driver to point to ALogger.

```
driver.setLogger(ALogger);
```

5. Make the screen logger point to ALogger.

```
MessageHandler.setScreenLogger(ALogger);
```

6. Set the type of rendering desired via

```
driver.setRenderer(Driver.RENDER_PDF);
```

7. Set the input source to

```
driver.setInputSource(new
InputSource(new FileInputStream(new
File(fopFileName))));
```

8. Set the output source to

```
driver.setOutputStream(new
FileOutputStream(new
File(pdfFileName));
```

9. Finally, render it.

```
driver.run();
```

If you are specifying XSLT and XML files as input, change steps 7 and 9 to:

```
InputHandler inputHandler = new
XSLTInputHandler(xmlFile, xsltFile);
driver.render(inputHandler.getParser(
), inputHandler.getInputSource());
```

Mailing the Generated PDF Report as an Attachment

After the PDF report has been generated, I use the JavaMail API to send the document out as an email attachment.

1. Obtain a default Session.

```
Session s =
Session.getDefaultInstance(myProperties, null);
```

myProperties is the properties

object. I supply the mail.smtp.host property defined in the Mail.properties file. The Authenticator object (the second argument) is used to indirectly check access permissions. If the Authenticator object is set to null, this means anyone can get the default session.

2. Create a new Message.

```
Message msg = new MimeMessage(s);
```

3. Set the To, From, Subject, and Date Sent fields.

4. Create the Message Body Parts.

```
MimeBodyPart mbp2 = new
MimeBodyPart();
```

5. Attach the file to the message.

```
FileDataSource fds = new
FileDataSource(attachment);
mbp2.setDataHandler(new
DataHandler(fds));
mbp2.setFileName(attachment);
```

6. Create a multipart object and add the message body parts from step 4 to it.

```
Multipart mp = new MimeMultipart();
mp.addBodyPart(mbp2);
```

7. Add the multipart object to the message.

```
msg.setContent(mp);
```

8. Send the message.

```
Transport.send(msg);
```

HTML Report Generation

Instead of generating a PDF report, I also show how XSLT can be used to transform the XML document created previously into an HTML report. A complete listing of the XSLT commands for this can be seen in Listing 2.

1. Lines 1 – 3 show the XML and namespace declarations.

2. The template rule is shown on line 4. This searches for and matches the root tag, replacing it with the content following it.

3. Table creation begins on line 13. The table headings are specified on lines 14–17.

4. Iterate through all the child nodes of

the <statistics> element, populating each row of the table with the object name and number of objects (lines 18–23).

5. From Listing 2, we see that the XSL-FO table and HTML table commands are very similar. Table 2 shows the relationship between the two.

XSL-FO Element	HTML Element
<fo:table>	<table>
<fo:table-body>	<tbody>
<fo:table-column>	cols attribute of <table> i.e. <table cols= >
<fo:table-row>	<tr>
<fo:table-cell>	<td>
<fo:table-caption>	<caption>

Table 2 • Mapping between XSL-FO and HTML table elements

Running the Test Application

1. Install Tomcat 5.X on your system
2. Drop your JSP files into the C:\jakarta-tomcat-5.0.16\webapps\MyApplication\jsp folder
3. Make sure that the associated JAR files are in the C:\jakarta-tomcat-5.0.16\webapps\MyApplication\WEB-INF\lib directory
4. Start up Tomcat, C:\jakarta-tomcat-5.0.16\bin\startup.bat
5. Start up your browser and point it at the start.jsp page

Conclusion

This article has shown you how the various technologies such as XSL-FO, XSLT, and JavaMail can be integrated into existing Web-based applications. A simple front end using JSP is provided as a test harness. I also explained how XSLT can be used to generate the XSL-FO commands, illustrated how the PDF or HTML reports are generated, and showed how JavaMail is used to e-mail the PDF reports to a specified recipient list.

Reference

- *Extensible Stylesheet Language, Version 1.0*: www.w3.org/TR/xsl/

Acknowledgements

I want to thank Tzu-Khiau Koh and Yee-Koon Loh for their comments on early drafts of this article. ☺

NG_A@HOTMAIL.COM

LISTING 1 • Transform the XML data into an XSL-FO file

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet version="1.0"
3
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
4
xmlns:fo="http://www.w3.org/1999/XSL/Format">
5
<xsl:template match="/">
6
<fo:root
xmlns:fo="http://www.w3.org/1999/XSL/Format">
7
<fo:layout-master-set>
8
<fo:simple-page-master master-name="simple"
9
page-height="29.7cm"
10
page-width="21cm"
11
margin-top="1cm"
12
margin-bottom="2cm"
13
margin-left="2.5cm"
14
margin-right="2.5cm">
15
<fo:region-body margin-top="3cm"/>
16
<fo:region-before extent="1.5cm"/>
17
<fo:region-after extent="1.5cm"/>
18
<fo:region-end extent="1.5cm"/>
19
</fo:simple-page-master>
20
</fo:layout-master-set>
21
<fo:page-sequence master-reference="simple">
22
<fo:title>Server Statistics</fo:title>
23
<fo:static-content flow-name="xsl-region-
before">
24
<fo:block text-align="center">
25
Server Statistics at <xsl:value-of
select="/statistics/@date"/>
26
</fo:block>
27
</fo:static-content>
28
<fo:static-content flow-name="xsl-region-
after">
29
<fo:block text-align="center">Page
<fo:page-number/></fo:block>
30
</fo:static-content>
31
<fo:flow flow-name="xsl-region-body">
32
<fo:block>
33
<fo:table table-layout="fixed">
34
<fo:table-column column-width="150pt"/>
35
<fo:table-column column-width="150pt"/>
36
<fo:table-body>
37
<fo:table-row>
38
<fo:table-cell border-
style="solid" display-
align="center"
39
border-color="black"
border-width="1pt"
padding-before="3pt"
padding-after="3pt"
padding-start="3pt"
padding-end="3pt">
40
<fo:block>Type of
Object</fo:block>
41
</fo:table-cell>
42
<fo:table-cell border-style="solid"
display-align="center"
43
border-color="black" border-
width="1pt"
padding-before="3pt"
padding-after="3pt"
padding-start="3pt"
padding-end="3pt">
44
<fo:block>Total
Number</fo:block>
45
</fo:table-cell>
46
</fo:table-row>
47
</fo:table-body>
48
</fo:table>
49
</fo:block>
50

```

```

51
<xsl:for-each select="/statistics/*">
52
<fo:table-row>
53
<fo:table-cell border-style="solid"
display-align="center"
54
border-color="black" border-
width="1pt"
padding-before="3pt"
padding-after="3pt"
padding-start="3pt"
padding-end="3pt">
55
<fo:block>
56
<xsl:value-of
select="name()" />
57
</fo:block>
58
</fo:table-cell>
59
<fo:table-cell border-style="solid"
display-align="center"
60
border-color="black" border-
width="1pt"
padding-before="3pt" padding-
after="3pt"
padding-start="3pt" padding-
end="3pt">
61
<fo:block>
62
<xsl:value-of select="."/>
63
</fo:block>
64
</fo:table-cell>
65
</fo:table-row>
66
</xsl:for-each>
67
</fo:table-body>
68
</fo:table>
69
</fo:flow>
70
</fo:page-sequence>
71
</fo:root>
72
</xsl:template>
73
</xsl:stylesheet>

```

LISTING 2 • XSLT commands to translate XML to HTML

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet version="1.0"
3
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4
<xsl:template match="/">
5
<HTML>
6
<HEAD>
7
<TITLE>Server Statistics - HTML
format</TITLE>
8
</HEAD>
9
<BODY>
10
<H1>
11
Server Statistics at <xsl:value-of select="
/statistics/@date"/>
12
</H1>
13
<Table border="1">
14
<TR>
15
<TH>Type of Object</TH>
16
<TH>Total Number</TH>
17
</TR>
18
<xsl:for-each select="/statistics/*">
19
<TR>
20
<TD><xsl:value-of select="name()" /></TD>
21
<TD><xsl:value-of select="."/></TD>
22
</TR>
23
</xsl:for-each>
24
</Table>
25
</BODY>
26
</HTML>
27
</xsl:template>
28
</xsl:stylesheet>

```

▼ Download the Code
www.sys-con.com/xml

FREE*CD! (\$198.00 VALUE!)

Secrets of the XML Masters

Every *XML-J* Article on One CD!



— The Complete Works —

CD is organized into 33 chapters containing more than 450 exclusive *XML-J* articles!

All in an easy-to-navigate HTML format! **BONUS: Full source code included!**

ORDER AT WWW.SYS-CON.COM/FREECD

**PLUS \$9.95 SHIPPING AND PROCESSING (U.S. ONLY)*

©COPYRIGHT 2004 SYS-CON MEDIA. WHILE SUPPLIES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE. ALL BRAND AND PRODUCT NAMES ARE TRADE NAMES, SERVICE MARKS OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES.

Only from the World's Leading i-Technology Publisher

Novell Releases SOA Suite That Supports Linux

(San Francisco) – Novell has released Novell exteNd 5.2, a Web services–based service-oriented architecture (SOA) suite with Linux support. Novell exteNd 5.2 enables developers to visually design and deploy Web services–based applications, and to integrate diverse systems to build composite applications on a broad range of application servers. Novell also unveiled plans to support and eventually bundle the JBoss Application Server with exteNd.

exteNd offers visual Web services development tools that hide the complexity of J2EE from Linux users. exteNd 5.2 now has Linux support from the desktop to the server, which should speed development and deployment of composite applications on Linux. It supports SUSE, Red Hat Linux, Windows, NetWare, and Solaris.

Novell exteNd 5.2 is available immediately, with pricing starting at \$50,000 per CPU. A free 30-day evaluation is available for download at www.novell.com.

Microsoft Releases New SharePoint Products and Technologies Toolkits

(Redmond, WA) – Microsoft has released two new Web Part toolkits and a Web services toolkit for Microsoft SharePoint Products and Technologies that enable them to connect systems from multiple vendors using XML Web services standards. Available at no charge on GotDotNet.com, and complete with source code, the two Web Part toolkits facilitate the integration of Microsoft Office SharePoint Portal Server 2003 and Windows SharePoint Services sites with information line-of-business applications from SAP AG and other vendors.

Microsoft's WSRP Web Part Toolkit for SharePoint Products and Technologies leverages the Web Services for Remote Portlets (WSRP) specification to enable developers to build portlets that interact with other portal sites, regardless of the business system they use.

Adding to the several means already available for integrating SAP functionality into SharePoint Portal Server environments, Microsoft has also released its Web Part Toolkit for SharePoint Products and Technologies for SAP iViews. This capability is especially useful for developers and system integrators creating composite applications that combine SAP resources with other

application, people, team and knowledge resources within SharePoint sites and enterprise portals. www.microsoft.com

Lenders First Choice Selects Fiorano ESB to Enable Real-Time Operations

(Los Gatos, CA) – Lenders First Choice (LFC), a national provider of title and settlement services, has selected Fiorano ESB as a platform to enable seamless process integration and real-time agility.

LFC deals with numerous third-party vendors to access, analyze, and update title information. The information bought from these vendors had to be extracted and synchronized with the LFC internal databases, after which it was published to the LFC Web site and CRM systems.

Fiorano's combination of integration, business process management, and service-oriented architecture alleviates this problem, allowing dynamic runtime changes in business processes and dramatically reducing the overall costs of integration.

www.lendersfirstchoice.com, www.fiorano.com

Network Director 3.0 Transforms Services Chaos

(San Francisco) – Blue Titan Software has announced the immediate availability of Network Director 3.0, their Enterprise SOA Fabric. The new release enables Fortune 500 enterprise architects to better control their rapidly growing SOA deployments. Network Director customers have already scaled Network Director deployments beyond 50 million service requests per day, while managing hundreds of service endpoints supporting thousands of shared services.

Network Director 3.0 introduces deep, proactive monitoring of all enterprise shared services and increased flexibility and ease in global policy provisioning and SLA management. It delivers broader reach, faster and broader provisioning of services, increased service reuse, and proactive management of the entire service network. It also facilitates simple, unified SOA adoption across the global enterprise with proven interoperability across any enterprise service bus, flexible non-SOAP message routing, while facilitating deployment on any J2EE-compliant runtime.

www.bluetitan.com

British Columbia Attorney General Streamlines Legal Proceedings

(Vancouver, BC) – Layer 7 Technologies, a provider of Web services security and policy management solu-

tions, announced that the British Columbia Attorney General has implemented SecureSpan to streamline legal proceedings. SecureSpan is a centrally administrable, standards-based solution that manages end-to-end security and integration policies across Web services transactions. The Ministry was tasked to establish an electronic filing system that would provide secure access to confidential legal data and availability among multiple parties. SecureSpan incorporated Web services and increased overall productivity, established legal process efficiencies while allowing information to be managed with current IT investments.

The Attorney General's office manages confidential information through an Integrated Justice System (JUSTIN) comprised of centralized databases. Layer 7 enabled the Ministry to expand access of critical information through Web services while maintaining tight control of how data is accessed. Through the SecureSpan Gateway, appropriate legal parties associated with the Ministry can request information that is transported through Web services.

www.layer7tech.com

Active Endpoints and JBoss Inc. Announce Comprehensive BPEL Development and Deployment Platform

(Shelton, CT) – Active Endpoints, Inc., has joined the JBoss Marketing Affiliates Program. This partnership will allow Active Endpoints and JBoss to offer a complete services-oriented integration platform to organizations that wish to create composite applications using the BPEL (Business Process Execution Language) standard.

Through the partnership, Active Endpoints will optimize its ActiveWebflow server technology to support the JBoss Application Server's capabilities, such as clustering, connection pooling, and management. The combined solution, ActiveWebflow Enterprise for JBoss, will include ActiveWebflow Professional, ActiveWebflow Server for JBoss, and ActiveBPEL Engine

www.active-endpoints.com, www.jboss.com.



J2EE Web Hosting

<http://www.webappcabaret.com/ws.jsp>
1.866.256.7973

Quality Web Hosting at a reasonable price...

<JAVA J2EE HOSTING AND OUTSOURCING>

JAVA Developers: Design and Architect with Struts. Develop with Eclipse. Build with Ant. Deploy to WebAppCabaret. It is that easy with a hosting company that understands the Java J2EE standard. Web Services - a buzz word or reality. How many web hosts provide the facilities to deploy and run applications written for Web Services? We Do. At WebAppCabaret, we lead the way in providing comprehensive Java J2EE and PHP/Perl Web Hosting solutions.

Easy Application Deployment is a reality at WebAppCabaret. Each Account's J2EE Application Server is installed with its own instance and default settings so you have complete control. Our Web Control Panel makes it a breeze for JVM restarts. Such is an example of our Advanced Web Hosting Infrastructure and Tools. If you are a consultant, do you have a complex web hosting requirement for your client? Web Host or Reseller, are you looking to provide services without the headaches of managing your own network infrastructure? Look no more.

For JAVA J2EE we offer the latest versions of Tomcat, JBoss, and Jetty Application Servers. For Scripting programming we offer the latest PHP and Perl. We also offer the latest MySQL and PostgreSQL Databases. Commercial software, including Resin, JRun, and Oracle, is also available for an extra charge. In addition we provide valuable tools such as Bugzilla Bug Tracking, CVS Version Control plus a whole lot more. Does your service provider offer ENCRYPTED ACCESS for Email (POP3S/SMTPS/IMAPS), Shell, and FTP? WE DO. All of this is backed by our Tier 1 Data Center with 100% Network Uptime Guarantee.

Below is a partial price list of our standard hosting plans. (Reseller accounts also available). For more details please log on to <http://www.webappcabaret.com/ws.jsp> or give us a call at 1(866)-256-7973.

\$39/mo Enterprise

Latest JBoss/Tomcat/Jetty
Latest JSP/Servlets/EJBs
Private JVM
Choice of latest JDKs
Dedicated IP Address
NGASI Control Panel
PHP and Perl
Web Stats
1GB Disk
200MB DB
MySQL
PostgreSQL
Dedicated Apache
Telnet . SSH . FTP
5 Domains
100 Emails
Web Mail . POP . IMAP
more...

Options

The following are some of the add-on options with the associated extra starting at costs:

Resin: \$10/mo

300MB Oracle: \$40/mo

JRun: \$20/mo

ColdFusion: \$30/mo

\$191/mo Dedicated

Managed Dedicated Server starts at \$191 per month for:

Our J2EE-Ready dedicated servers make deploying complex applications a breeze.

NGASI Control Panel with your own Logo
Each dedicated server configured for standalone web application and web hosting (resellers) at no extra charge.
more...

\$3000/mo 4Balance

4Balance is our entry level High Availability Load Balancing service comprised of:
1 Database Server
4GB RAM
Dual Xeon
2 Application Servers
2GB RAM
Single Xeon
1 Load Balancer
1GB RAM
Single Xeon

VPN

Need to host a site outside your corporate network with secure access? Our VPN service is the solution for peace of mind.

\$99/mo Reseller

If you cannot afford a Dedicated server for reselling web hosting, for \$99/mo the Shared Reseller plan gives you:

10 Professional Plans
Your Web Site
NGASI Control Panel with your own Logo
50% Discount
No System Admin
Easy Account Mgt
more...

LEARN FROM EXPERIENCE.

SAP TECHEd '04 OCTOBER 5-8 SAN DIEGO

At SAP® TechEd, you'll learn how to exploit SAP NetWeaver™ to tap the full potential of your IT infrastructure and increase your company's competitive advantage — all from leading SAP development experts and technical pros. You'll gain from your own experience too: hands-on training that lets you learn by doing, demonstrations of development tools, and practical strategies for ramping up renewed innovation in your IT organization. For the third consecutive year, an amazing 97% of attendees polled would recommend SAP TechEd to their colleagues. That's you! So come learn how SAP NetWeaver helps your organization achieve growth through innovation while reducing TCO. To help you start lowering cost, register three people from your company, and a fourth can come for free.

Register online now at www.sap.com/usa/teched

THE BEST-RUN BUSINESSES RUN SAP

