

# Web Services JOURNAL

**.NET J2EE XML**

July 2005 Volume 5 Issue 7

## Open Grid Service Architecture (OGSA)

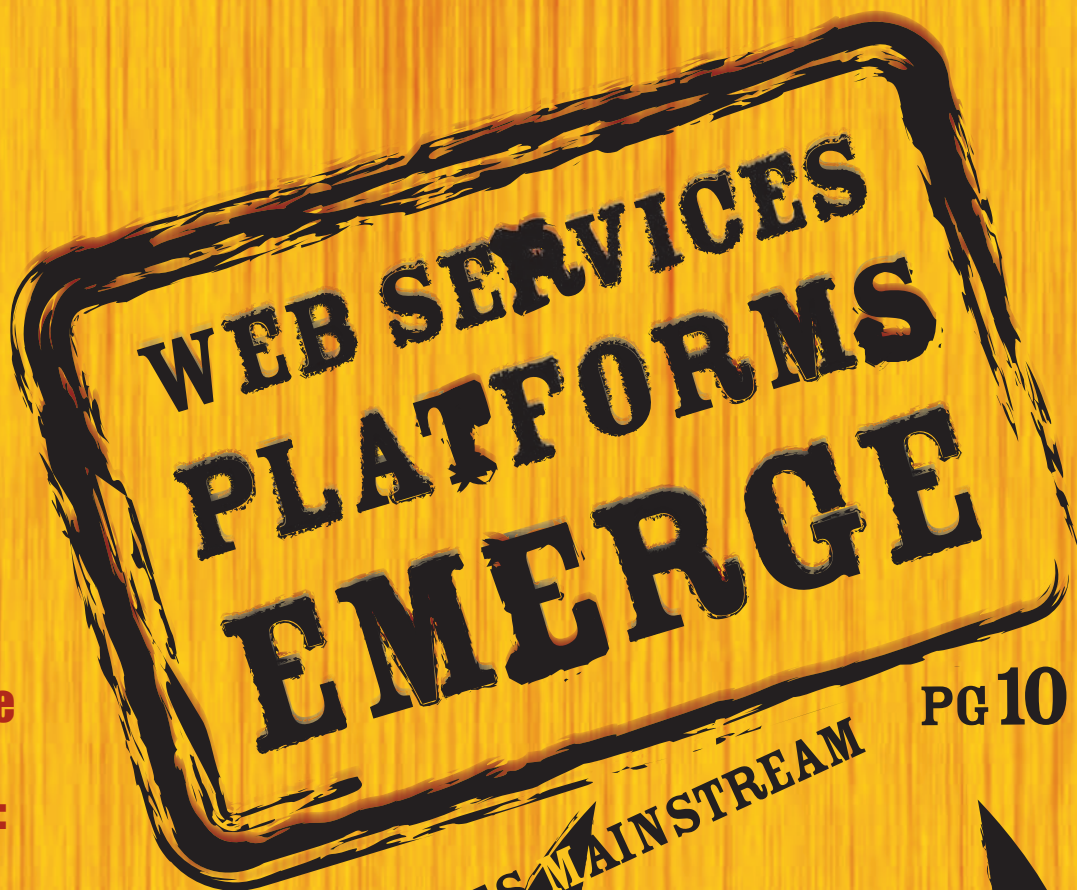
SOA Journey:  
From Web services  
to grid computing  
pg.20

## Why WSDM Matters

The role of WSDM  
in distributed  
IT management  
pg.26

## Managing Enterprise Data Complexity Using Web Services: Part 2

Developing enterprise  
digital dashboards using  
data services architecture  
pg.32



SOA GOES MAINSTREAM PG 10



RETAILERS PLEASE DISPLAY  
UNTIL SEPTEMBER 30, 2005

\$6.99US \$7.99CAN



71486 03420 9

07>

## Keep Your Skills Ahead of the Crowd

Keeping your IT skills ahead of the crowd is not as difficult as most people fear. Staying on top of the trends may seem like a daunting task if, like most people, you assume that each new technology is a completely new invention that you must learn from the ground up. Fortunately, nothing is really all that new. Inventors typically create new technologies by studying existing technologies, then building upon them in ways that extend and improve them. 100% new technological advancements are very rare.

Inventors almost always leverage legacy technologies as they invent new ones. Why not leverage your own knowledge of those legacy technologies as you try to learn about the new inventions? To learn about new technologies as painlessly as possible, consider how each new advancement is similar to what you already know.

For example, consider Web services. Web services are a new trend, but — at a technological level — the parts of a Web service are not all that unique. Web services are based on remote procedural calls — messages sent to a server, which calls the requested function. RPCs were developed years ago, and are hardly a new concept. Really, the only "new" thing in Web services is the standard that is being used to write the application. If you break down Web services in this way, it's easy to learn about them. To continue with this process, you might next explore the payload requirements, the process for determining what function to call, and how the call works. As you can imagine, it's a lot more efficient — and interesting — to learn about a new technology based on its relation to familiar technologies than to learn about it by reading the specification cover to cover.

As always, the devil is in the details. But most details are critical only if you want to specialize in a given technology. For instance, if you want to specialize in Web services, you need to familiarize yourself with the details of Web service development. In that case, your next step would be to learn how to format the messages, how to expose Web services, and so on.

— Adam Kolawa, Ph.D.  
Chairman/CEO of Parasoft

## It's automated. It's fast. And it's the most versatile Web Services testing tool.



## SOAPtest® enables you to deliver better Web Services in less time.

The simple fact is, no other Web service testing product can do what SOAPtest can do. From verifying functionality and performance to ensuring security and interoperability, SOAPtest automates all critical testing processes across the entire lifecycle of your Web service.

**But don't take our word for it....** Go to [www.parasoft.com/soaptest\\_WSD](http://www.parasoft.com/soaptest_WSD) and try it for free — no commitment, no obligation. If you like it (and we suspect you will) just let us know. We'll be more than happy to help you and your development team get up and running.

For Downloads go to [www.parasoft.com/soaptest\\_WSD](http://www.parasoft.com/soaptest_WSD)



Email: [soaptest@parasoft.com](mailto:soaptest@parasoft.com) Call: 888-305-0041 x1209

### Features

- WSDL schema verification and compliance to standards
- Automatic test creation using WSDL and HTTP Traffic
- Data-driven testing through data sources (Excel, CSV, Database Queries, etc)
- Scenario-based testing through XML Data Bank and Test Suite Logic
- Flexible scripting with Java, JavaScript, Python
- WS-I Conformance: Basic Profile 1.0
- WS-Security, SAML, Username Token, X.509, XML Encryption, and XML Signature support
- WS-Security Interop testing emulator
- MIME Attachment support
- Asynchronous Testing: JMS, Parlay (X), SCP, and WS-Addressing support
- Windows Perfmon, SNMP, and JMX monitors
- Detailed Report generation in HTML, XML and Text formats
- Real-Time graphs and charts

### Benefits

- Uniform test suites can be rolled over from unit testing to functional testing to load testing
- Prevent errors, pinpoint weaknesses, and stress test long before deployment
- Ensure the reliability, quality, security and interoperability of your Web service
- Verify data integrity and server/client functionality
- Identify server capabilities under stress and load
- Accelerate time to market

### Protocol Support

- HTTP 1.0
- HTTP 1.1 w/Keep-Alive Connection
- HTTPS
- TCP/IP
- JMS

### Platforms

- Windows 2000/XP
- Linux
- Solaris

### Contact Info:

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

[www.parasoft.com](http://www.parasoft.com)





**XML'S ENDLESS POSSIBILITIES,**

**NONE OF THE RISK.**

## **FORUM XWall™ Web Services Firewall - Reinventing Security**

SECURITY SHOULD NEVER BE AN INHIBITOR TO NEW OPPORTUNITY: FORUM XWall™ Web Services Firewall has been enabling Fortune 1000 companies to move forward with XML Web Services confidently. Forum XWall regulates the flow of XML data, prevents unwanted intrusions and controls access to critical Web Services.

VISIT US AT [WWW.FORUMSYS.COM](http://WWW.FORUMSYS.COM) TO LEARN MORE ABOUT HOW YOU CAN TAKE YOUR NEXT LEAP FORWARD WITHOUT INCREASING THE RISKS TO YOUR BUSINESS.



**FORUM SYSTEMS™ — THE LEADER IN WEB SERVICES SECURITY**





# InsideWSJ

## Web Services Platforms Emerge

SOA goes mainstream

By Dave Shaffer

10

## Open Grid Service Architecture (OGSA)

SOA journey: From Web services to grid computing

By Ramy Abaas

20

## Managing Enterprise Data Complexity Using Web Services: Part 2

Developing enterprise digital dashboards using data services architecture

By Dr. Jai Ganesh & Dr. Sriram Anand

32

### From the Editor

#### Platform Shoes

Sean Rhody ..... 7

#### Industry Commentary Phasing in SOA and Web Services

Ajit Sagar ..... 8

#### WSJ: Basics

##### Where HTTP Fails SOAP

Alternative choices for situations when HTTP does not scale sufficiently for enterprise Web service deployments

Frank Lynch & Mark Fynes ..... 16

#### WSJ: Management

##### Why WSDM Matters

The role of WSDM in distributed IT management

Chris Peltz ..... 26

#### WSJ: Architecture

##### Designing Services for Performance – Part II

Continued discussion of the secrets of building and operating a realistic SOA

David Linthicum ..... 38

#### WSJ: Product Review

##### Altova Enterprise Suite 2005

Effective, powerful, and useful

Brian Barbash ..... 40

#### WSJ: Data Binding

##### XML Binding Frameworks in the Context of Service-Oriented Architecture

Make an informed choice about a binding framework for your SOA needs

Dr. Srinivas Padmanabhuni, et al ..... 44



## An Easy Introduction to XML Publishing – Part 3 of a Five-Part Series

PG BARTLETT

50

## Exploring XML Schema Styles Using JAXB in Enterprise Applications

PUSHKAR VARMA

54



# WebRenderer™



## Standards compliant embeddable web browser component

WebRenderer is a cutting edge embeddable Java™ web content rendering component that provides Java applications and applets with a fast, standards compliant HTML and multimedia rendering engine. WebRenderer provides a feature-packed API including complete browser control, a full array of events, JavaScript interface, DOM access, document history and more.

### Why WebRenderer?

- Standards Support (HTML 4.01, CSS 1 & 2, SSL, JavaScript, XSL, XSLT etc.)
- Exceptionally Fast Rendering
- Predictable Rendering
- Scalability (deploy in Applications or Applets)
- Security (based on industry standard components)
- Stability and Robustness

Embed WebRenderer to provide your Java® application with standards compliant web content rendering support.

To download a 30 day trial of WebRenderer visit  
[www.webrenderer.com](http://www.webrenderer.com)

**JadeLiquid™  
Software**

Copyright JadeLiquid Software 2004. JadeLiquid and WebRenderer are trademarks of JadeLiquid Software in the United States and other countries. Sun, Sun Microsystems, the Sun Logo and Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All other trademarks and product names are property of their respective owners.



# Mindreef® SOAPscope®

## Take Control

of your

## Web Services

### Developers • Testers • Operations • Support

Web services and SOA have changed the rules with the introduction of a new XML abstraction layer.

Though most development organizations have tools and processes for managing code objects, they have little or no help for testing, diagnosing, or supporting the XML portion of their Web services.

Mindreef SOAPscope completes your Web services development toolkit by combining XML-aware tools for developers, testers, support, and operations. This flexible combination gives you the right tool for any diagnostic task, whether working alone or collaborating with other team members.

Start taking control of your Web services by downloading a FREE trial version of Mindreef SOAPscope today!



*Mindreef SOAPscope: Named best Web services development tool in the 2005 InfoWorld Technology of the Year awards*

**Test** *No coding required!*  
Test service contracts and underlying service code with an easy-to-use, forms-based interface

**Diagnose** *No more wading through angle brackets!*  
Find the cause of a problem with powerful message collection, analysis, and diagnostics tools

**Collaborate** *No more emailing XML snippets!*  
Share test or problem data with others, regardless of their roles or system requirements

**Support** *No more finger pointing!*  
Web services require support at the API level – a real burden for most organizations. Mindreef SOAPscope makes it easy by allowing customers and support staff to share Mindreef package files that contain complete problem data

**Download a free version of Mindreef SOAPscope at [www.Mindreef.com/tryout](http://www.Mindreef.com/tryout)**

© Copyright 2005, 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 Politecnico di Torino and its contributors.



**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**

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 andy@enterprisedb.com

David Chappell chappell@sonicsoftware.com

Anne Thomas Manes anne@manes.net

Mike Sick msick@sys-con.com

Michael Wacey mwacey@csc.com

**INTERNATIONAL TECHNICAL EDITOR**

Ajit Sagar ajitsagar@sys-con.com

**MANAGING EDITOR**

Seta Papazian seta@sys-con.com

**EDITOR**

Nancy Valentine nancy@sys-con.com

**ONLINE EDITOR**

Roger Strukhoff roger@sys-con.com

**PRODUCTION**

**PRODUCTION CONSULTANT**

Jim Morgan jim@sys-con.com

**LEAD DESIGNER**

Andrea Boden andrea@sys-con.com

**ART DIRECTOR**

Alex Botero alex@sys-con.com

**ASSOCIATE ART DIRECTORS**

Abraham Addo abraham@sys-con.com

Louis F. Cuffari louis@sys-con.com

Tami Beatty tami@sys-con.com

**CONTRIBUTORS TO THIS ISSUE**

Ramy Abaas, Sriram Anand, Brian Barbash, PG Bartlett,  
Mark Fynes, Jai Ganesh, Dion Hinchcliffe, David Linthicum,  
Frank Lynch, Bijoy Majumdar, Ujval Mysore,  
Srinivas Padmanabhuni, Chris Peltz, Sean Rhody,  
Ajit Sagar, Dave Shaffer, Vikram Sitaram, Pushkar Varma

**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 © 2005 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.



# Platform Shoes

This column might have been titled "on the SOAPbox," except I think I used that one already. Nevertheless, I want to discuss platforms. Politicians used to use platforms, (real ones, not some murky promises that they abandon after the election) to stand above the crowd, so as to be seen and heard better. This was back before the days of television and radio, but still, even now, everybody loves to be up on stage, to be seen. A platform gave them that opportunity to present themselves as larger than life, better, and right for the people.

In much the same fashion, a Web services platform allows you to do a better job of designing, developing, testing, deploying, and managing Web services. Picking a platform allows you to concentrate your efforts along well-defined lines, achieving efficiencies along the way.

It's easy when discussing Web services to get caught up in the higher levels of service-oriented architecture. There is a host of issues, including security and transactionality, that have to be considered and addressed in an enterprise architecture. The typical Web services person spends as much time worrying about standards such as WS-Transaction or BPEL as he or she does about the underlying implementation of the actual service itself.

This just makes the selection of a platform all the more important. A platform should enable Web services at various levels. First and foremost, it should offer an effective mechanism for creating, and exposing, services themselves. Although this sounds elementary, it isn't trivial. It's impossible for an organization to rewrite all of its software to accommodate a paradigm switch – there's too much of an investment already built into the existing technology. So while an ideal world might have all services written in Java, the real world will have them mixed with Cobol, Visual Basic, Fortran, and a host of other languages. Adding to this dilemma is the fact that most large organizations purchase packaged software to accomplish some of the business processes they employ. So in addition to organic development, Web services designers need to take into account things like SAP, PeopleSoft, and Oracle Financials, not to mention other industry-specific packages. In order to be effective,



WRITTEN BY  
**SEAN RHODY**

a platform must integrate into multiple environments and enable development around various APIs.

However for a platform to be truly effective it also has to manage these concepts at higher levels. Many of the tasks that used to fall to the individual programmers, such as security, transaction management, logging, and routing are now handled at higher levels. Applica-

tion servers began this drive to extract common services from developer code. CORBA, DCE, J2EE, and .NET all provided the concept of an interface in some fashion, and Web services has taken it to the logical end point – a neutral, ubiquitous mechanism for defining services. It has also taken the task of defining the service away from the application server, and moved it squarely into the services arena. So now the tasks that previously fell to the developer fall to the architect or to the business process designer.

A good platform would need to provide tools and productivity enhancements to support these higher-level users. Process modeling tools that actually connect to actual services (or even drive the creation of services as a result of the modeling process) will enhance the productivity of designers and move service definition closer to the actual end users. Web services are largely about computer-to-computer interaction, but the main reason for the existence of the computers themselves is to enable some business process.

Last, a platform must be flexible. No one vendor can provide all of the pieces of a complete SOA. So, a good platform vendor will realize that the components that make up an SOA will be standards based, and the platform should be able to replace components as desired without compromising the overall usefulness of the platform.

That's a lot of work for a single vendor, or even a community, but platforms provide clear benefits over a la carte development. In this issue we bring you some more insight into platforms and effective development of Web services. Enjoy. ©

■ **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

**NATIONAL SALES & MARKETING MANAGER**

Dennis Leavey dennis@sys-con.com

**ADVERTISING MANAGER**

Megan Mussa megan@sys-con.com

**ASSOCIATE SALES MANAGERS**

Dorothy Gil dorothy@sys-con.com

Kim Hughes kim@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**

**CONSULTANT, INFORMATION SYSTEMS**

Robert Diamond robert@sys-con.com

**WEB DESIGNERS**

Stephen Kilmurray stephen@sys-con.com

Percy Yip percy@sys-con.com

Vincent Santaiti vincent@sys-con.com

**ACCOUNTING**

**FINANCIAL ANALYST**

Joan LaRose joan@sys-con.com

**ACCOUNTS PAYABLE**

Betty White betty@sys-con.com

**ACCOUNTS RECEIVABLE**

Gail Naples gailn@sys-con.com

**SUBSCRIPTIONS**

SUBSCRIBE@SYS-CON.COM

1-201-802-3012

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: \$89.99/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.



# Phasing in SOA and Web Services

Over the last couple of years, the industry has rallied around SOA and its main realization platform – Web services. While many of the clients I meet are still wary about the adoption of new technology, integration dilemmas posed by the variety of software and hardware platforms has led them to buy into the promise of improving on business agility through SOA. The ubiquitous nature of Web services is something that even business owners appreciate, as they have been burned before by the disparity in the technologies that their applications have been based on.

Many of the clients I have met for assessing the feasibility and process of adopting SOA have similar questions. Often it takes a bit of cross-questioning to understand the actual intent of these questions, as some of them are so generic that you can always answer them with a “one size fits all” approach. However, this would not help in addressing the actual issues. Some of the commonly asked questions are as follows.

*How do we implement SOA and Web services at our organization?* This is, of course, a very generic question. However, upon digging further into the source, basically the clients are interested in knowing how they can redo their existing applications so that they can achieve a better level of integration within their enterprise. The main concern here is that there are existing applications that are already deployed. What is the best way to plan an iterative approach for adopting Web services, while minimizing business impact?

*How do you keep track of how Web services are being utilized once they are implemented?* This basically translates into two aspects of Web services management – the value provided to the end customers who are consuming the service, and the management of Web services to track usage.

*Web services / SOA are new concepts. How do you separate the hype from the reality?* Again, this is a very generic question. The real question here is for the project sponsor (the one who is trying to bring this substantial change into the organization) to be able to present the cost-benefit analysis of adopting SOA and Web services. The technologies and the concepts may promise the world with a fence around it, but how does one practically measure the success of a project and justify the cost incurred in introducing the change?



WRITTEN BY  
**AJIT SAGAR**

*How do we do reusable business services when many projects are in transition?* This is a very valid concern. Web services will obviously be introduced into a dynamic environment, unless the company is just starting to build their product. Many of the projects will be well underway. How does one plan to continue projects on their existing path while adopting the new paradigms?

*How can we promote and deliver on the promise of reuse?* How do we incorporate a governance process so that reusable services are introduced into a multitude of applications? At the same time, how can we extract reusable services from different applications? This requires a center of excellence or a governance body to be set up at the onset, so that maximum reuse can be derived from the project. This also requires a formalized strategy that promotes the adoption of industry standards and their percolation throughout the organization.

*What are the performance impacts of adopting Web services?* The meaning of the question is obvious. What is not clear is how the project can justify the degradation in performance against the benefits achieved by moving to an SOA through Web services. This is, again, a cost-benefit question.

These questions need to be addressed before starting a project. Addressing such questions and concerns requires a careful assessment of the client's environment. This in turn requires answering another series of questions. What is the skillset of their existing resources? What technology base are they starting from – are their applications based on a messaging infrastructure or are they primarily based on synchronous interactions? The best way to mitigate the risks involved in the adoption of SOA and Web services and to address client concerns is to invest in a planning and analysis phase that addresses and answers these concerns and delivers a roadmap for Web services adoption – thus leading to a more agile enterprise. ©

## About the Author

Ajit Sagar is a principal architect with Infosys Technologies, Ltd., a global consulting and IT services company. Ajit has been working with Java since 1997, and has more than 15 years experience in the IT industry. During this tenure, he has been a programmer, lead architect, director of engineering, and product manager for companies from 15 to 25,000 people in size. Ajit has served as JDJ's J2EE editor, was the founding editor of XML Journal, and has been a frequent speaker at SYS-CON's Web Services Edge series of conferences. He has published more than 100 articles.

■ ■ ■ [ajitsagar@sys-con.com](mailto:ajitsagar@sys-con.com)



**Special Co-located Program: Gartner Enterprise Architecture Summit**  
September 14-15. Visit [gartner.com/us/adea](http://gartner.com/us/adea) for details.

# Gartner Application Development Summit 2005

## Service-Oriented Architecture and Agility:

Are You Ready for  
the Next Step?

**September 12-14, 2005 • Gaylord Texan Resort and Convention Center • Dallas, TX**

Join Gartner analysts and your peers to understand what the AD organization will evolve into as agility and services take center stage.

### Four Unique Tracks for 2005:

- Application Lifecycle for Service-Oriented Architecture
- AD Resource Management
- Modernization and Integration (Transition)
- AD Governance and Control

### Register Now!

Visit [gartner.com/us/ad](http://gartner.com/us/ad) or call 1 800 778 1997

**Interested in exhibiting?** For available sponsorship opportunities, contact:  
David McMahon (Companies A to N) [david.mcmahon@gartner.com](mailto:david.mcmahon@gartner.com) or +1 203 316 6603  
Michael McGrath (Companies O to Z) [michael.mcgrath@gartner.com](mailto:michael.mcgrath@gartner.com) or +1 203 316 1729

### Special Keynote Guests



**Michael A. Cusumano**  
Distinguished Professor at  
the Massachusetts Institute of  
Technology's Sloan  
School of Management



**Roger Staubach**  
Heisman Trophy winner,  
Super Bowl MVP,  
Pro Football Hall of Fame,  
Chairman and CEO of  
The Staubach Company

**Priority Code: ADWSJ**

**Gartner  
Application  
Development Summit**

# WEB SERVICES PLATFORMS EMERGE

## SOA goes mainstream

■ J2EE application servers enabled mainstream developers to build sophisticated multitiered Web applications that use mature standards and any of several commercial and open source application server platforms. A similar pattern has now emerged with the maturation of standard platforms for enabling service-oriented architecture (SOA) through Web services. These Web services platforms provide the same mainstream developer community with design patterns for implementing composite applications that leverage adapters, integration, transformation, business process management, and Web services. This article examines the key requirements for an effective Web services platform, related standards, and what impact this is having on developers and enterprise IT.

### A Use Case: DSL Provisioning

To keep this discussion grounded, let's look at a real-world example for doing DSL provisioning at a major European telco. (Actually, this use case is a composite of several production SOA customers.) In this example, the requirements are to build several composite applications that automate the provisioning of new



WRITTEN BY  
**DAVE  
SHAFFER**

DSL orders. These orders can come in through a variety of channels, including customer service representative call centers and a public Web application. Once initiated, an order typically takes several days to complete, including sending a modem to the customer for self-installation, scheduling the required network activations, and initiating the associated back-end billing systems. Keep in mind that although this example

involves a long-running process, because it adds some extra complexity and new requirements, a subset of these same requirements applies to short-lived, synchronous processes. In fact, the real telco behind this use case also has synchronous processes – for performing network tests when customers call in to report a problem with their line, for example. A simplified version of the process is shown in Figure 1.

### Web Services Platform Requirements

If you think about building this application on a pure Java/J2EE platform, you will quickly see that several specific challenges will make this a difficult problem to solve – and an even more difficult situation to maintain and manage over time. Challenges include:

- Integrating with many heterogeneous back-end systems and protocols (some of which are SOAP, but most are not)
- Avoiding “hard-wired” process flows and making process state persistent
- Interleaving UI page flows with process flows
- Handling complex data types and data transformations
- Securing interactions with services through uniformly managed security policies
- Incorporating business rules, which should be changeable without recompilation or deployment of apps (“automated business policies”)



- Unit testing and stress testing the composite apps
- Monitoring, administering, and reporting against the processes – for example, “Page me when the exception rate exceeds a given threshold,” “What are the bottlenecks in the production process?”
- Leveraging middleware technology that is already in place

Conveniently, most of these are not new requirements, and by now the problem space is fairly well understood. Furthermore, over the past 10 years standards and vendor technologies for EAI, BPM, workflow, B2B, and the like have evolved and matured to the point where platforms can now address all of these requirements in a standard fashion.

To get to this point, industry progress has been generally incremental, but I see two key developments that have combined for a more revolutionary impact. One of these is the industry-wide acceptance of the XML Schema, WSDL, SOAP, and BPEL (Business Process Execution Language) standards. The second is the emergence of architectural platforms that use these standards to both address individual functional requirements and provide the glue between the components of a full solution. The value of these standards, when implemented religiously by vendors, is to enable an out-of-the-box, fully functional platform solution along with the ability to plug in best-of-breed technology components when appropriate. Table 1 shows

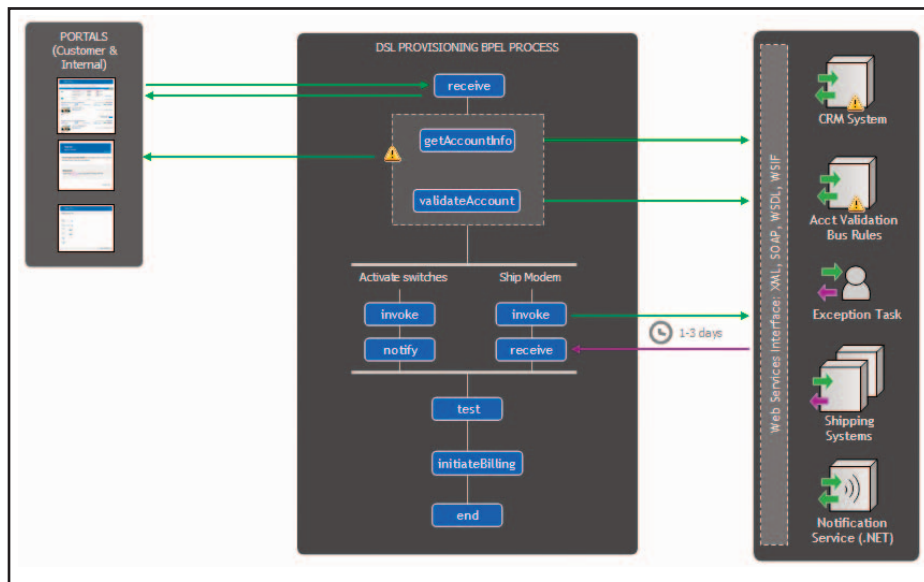


FIGURE 1 DSL provisioning use case

best practices architecture for addressing each of the preceding requirements in a Web services platform and below is an examination of each of these approaches in more detail, along with the key standards that are relevant to each.

### Standard Web Services Interfaces (XML, XML Schema, WSDL)

XML and XML Schema are now widely used for enterprise integration data models, as is WSDL for the component interface model. In the case of the DSL provisioning application, this means that a preexisting Microsoft.NET notification service can be easily integrated into

the composite application. It also suggests that platform vendors need to work hard on interoperability, encouraging conformance to the WS-I Basic Profile and performing rigorous interoperability testing. For example, Oracle runs frequent interoperability tests against all of the service WSDLs at [www.xmethods.org](http://www.xmethods.org). Integrating several Web services via SOAP, WSDL, and XML into a composite app is much quicker and easier than using traditional EAI technologies.

### Flexible Binding Framework (WSIF) and Adapters (JCA)

Web services means much more than just SOAP. The Oracle platform leverages many open source technologies; one of the most interesting is WSIF, the Web Services Invocation Framework, from Apache (<http://ws.apache.org/wsif/>). WSIF provides a mechanism for gaining the benefits of Web services – a standard WSDL interface and an XML data model – without requiring SOAP and its associated performance impact and other limitations. Specifically, with WSIF, you can use native protocol bindings (such as Java, EJBs, non-SOAP HTTP, JCA, pure socket-level TCP/IP, and so on) but still have a WSDL interface for your service. This enables a Web services component model but keeps the performance and transactionality of the underlying protocol.

For the telco in our example, the existence of 6,000+ network equipment endpoints

Architectural Approach	Requirement Addressed
Standard Web services interfaces	Integrate many endpoints, with reasonable effort
Flexible binding framework and adapters	Integrate directly to non-SOAP services
BPEL for process flow logic	Avoid hard-wired process flows and make long-running process state persistent
UI frameworks such as Struts/JSF and portals alongside BPEL	Interleave page flows with process flows
Native XML and XSLT/XQuery support	Handle complex data types and transformations
Policy- and standards-based security	Secure interactions with heterogeneous services
External business rules	Make process logic flexible and changeable at runtime
Full life-cycle support	Test and monitor process flows
Interoperable and pluggable architecture	Leverage existing middleware

TABLE 1 Best practices architecture for addressing Web services platform requirements

“ Although many people are interested in automating a process for operational efficiency improvements, it is often just as important to gain the visibility an explicit process model enables ”

with Java interfaces means that wrapping all of those endpoints as SOAP services would require a lot of work and have major performance implications. Using the WSIF Java binding allows those network endpoints to be “orchestrated” as if they were Web services, but without the overhead of SOAP. In addition, once a WSDL file with a WSIF binding has been created (and tools such as Oracle JDeveloper exist to do this automatically for Java classes and EJBs), the WSDL file can be put into a service directory and incorporated into a BPEL process as easily as any SOAP Web service.

Finally, WSIF includes a JCA binding, through which hundreds of adapters can easily be accessed from standard BPEL applications. This allows a Web services platform to provide an open, extensible approach along with numerous packaged adapters.

### BPEL for Process Flow Logic

It is now evident that every major technology vendor (platform or otherwise) is committed to BPEL support. However bear in mind that the WSBPEL standard, owned by OASIS, provides an XML language for describing executable business processes. If the process logic is explicit and standard, rather than proprietary or hard-wired in custom Java code, developers write less code, customers avoid vendor lock-in, and tools are more interoperable.

### UI Frameworks and Portals Alongside BPEL (Struts, JSF, JSR 168)

A full platform approach to Web services and SOA provides an implementation blueprint for

incorporating UI technologies and standards with process flow and integration logic. I’m often asked where the line should be drawn between UI page flows and process flows. Although it is certainly possible to use workflow languages such as BPEL to orchestrate page flows, page flows are usually better implemented with technologies such as Struts or JSF and integrated with process flows via SOAP or Java APIs at appropriate points. In the DSL provisioning example, sophisticated portals already existed for call center reps to initiate DSL applications, check status, log trouble tickets, and the like. Moving to an SOA in this case should not require that the UIs be reimplemented. I also see people frequently integrate BPEL processes into portal UIs, and here a Web services platform should support JSR 168 and make it easy to define portlets for common UI tasks, such as initiating processes and interacting with task lists.

### Native XML and XSLT/XQuery Support

Any SOA platform must address data trans-

formation requirements, because integration problems, almost by definition, require interfacing to many different systems, each with its own data models and native formats. However, once you get to a standard data model (XML), this problem is greatly simplified, and XSLT and XQuery are the standards most enterprises are using for XML data transformation. Another common requirement stems from the fact that many XML Schemas are *very* complex. Specifically, I see this causing problems with JAXB-style bindings for highly complex schemas such as OAG, in manufacturing; IFX, in financial services; OpenGIS, for geographical data; OTA, for travel; and so forth. Such complex schemas mean that a Web services platform should be based on native XML but offer XML/Java bindings as an option, so that automated mapping between XML and Java does not have to be a bottleneck.

However, beyond Java-to-XML mapping, any Web services platform must also provide a nice tool for mapping between non-XML data and XML, because lots of data remains in non-XML formats (CSV, fixed-length file formats, COBOL copy books, HL7, and the like).

### Policy- and Standards-Based Security (WS-Security, SAML)

Security is a key requirement of any distributed architecture, and regulatory pressures make this issue even more critical. An SOA introduces the requirements of authentication for access to distributed services as well as selective encryption of the data exchanged with them. Once again, there have been lots of proprietary technology solutions for these problems, but they did not allow interoperability across heterogeneous services. I now see WS-Security and SAML as enabling services to be implemented with different technologies, yet still sharing single-sign-on (SSO) credentials and exchanging data securely.

### External Business Rules

A best practice I have seen is to externalize configurable business rules so they are independent of the deployed process flow logic. This allows these rules to be created and edited with different tools, enabling some of the work to be done by nondevelopers. I also encourage the development of data-driven processes, which allow for fewer separate process flows for implementing common patterns.

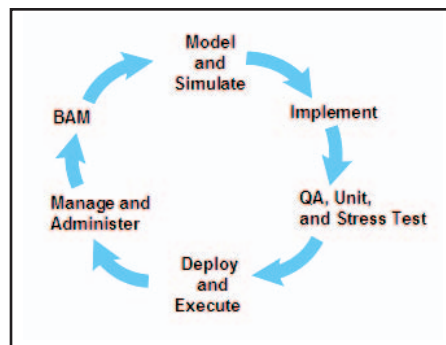


FIGURE 2 Web services platform life-cycle support





## THE FAST LANE TO INTELLIGENT INTEGRATION

Are you being asked to deliver more fixed price application and data integration projects? Are those integration projects taking too long and increasing your risk of not delivering on budget? Do you find that the cost and complexity of the integration products available in the market prohibitive? It doesn't have to be that way...

With the FusionWare Integration Server, Solution Providers like you are:

- Expanding the reach of their business with an affordable and easy-to-use Web services solution built entirely on open standards
- Enhancing their existing customer loyalty by providing integration technology which helps them leverage their existing systems quickly and affordably
- Minimizing complexity and risk of integration with a product that runs on any of the major operating systems and requires no additional infrastructure components
- Increasing their sales and profitability

Find out how! Contact FusionWare today at: [sales@fusionware.net](mailto:sales@fusionware.net).

**FusionWare**  
Corporation  
[www.fusionware.net](http://www.fusionware.net)

## Full Life-Cycle Support (BPMN/UML, BPEL, JUnit, JMX)

The power of a Web services platform is fully realized when it gracefully supports all of the phases of the design/development/deployment life cycle. To me, this means at least the ones shown in Figure 2.

A complete platform provides support for all of these phases, minimizing any loss of information in the transition from one to another. An open platform with a firm commitment to standards enables the different actors in this life cycle to use existing tools and methodologies for each phase while minimizing information loss.

Several articles have focused on how BPEL provides the glue between the modeling/simulation and implementation phases of this life cycle (see "Supporting the Business Process Lifecycle Using Standards-Based Tools," *Web Services Journal*, April 2005). So here I want to focus on some of the other life-cycle phases in Figure 2. For example, testing composite applications in an SOA introduces a new class of problems. Stress testing becomes critical, because applications and services do not control the loads that may be placed on them. A platform needs to make it easy to perform stress testing and performance analysis through instrumentation of the executing processes. Similarly, unit testing and QA of composite applications requires a nice way to simulate the different responses (or lack thereof) that may come back from services, introduce assertions that are tested, and analyze code coverage. Although standards don't really exist in this area, Oracle has extended JUnit to provide a unit test framework called BPEL TestRun, which automates testing and code coverage analysis for service-oriented applications.

Later come the managing and monitoring of these processes, and here I believe that BAM (Business Activity Monitoring) is often the "killer app" for automating business processes. Although many people are interested in automating a process for operational efficiency improvements, it is often just as important to gain the visibility an explicit process model enables. By making the flow explicit by use of a standard such as BPEL, you make it easy for a platform to provide a visual representation of process states and audit trails as well as to provide real-time monitoring and alerts. Alerts allow problems such as potential SLA violations to be identified – and preventive actions

taken – before the violation actually occurs.

## Interoperable and Pluggable Architecture

The functional footprint for an SOA platform includes the following:

- Application server
- Directory server
- Portal
- Web services security and management
- Rules engine
- Database
- Message-oriented middleware
- Developer tools/IDE

Any complete SOA platform must include all of these capabilities. However, enterprises should utilize a Web services platform architecture, whether or not all of the components come from a single vendor, and choose vendors that support a pluggable architecture. Achieving this requires that a platform utilize an open and standard architecture, so, for example, a process engine can leverage a customer's application server or database of choice, and vice versa. Enabling customers to get the cost and other benefits of a single-vendor Web services platform is smart, but *requiring* customers to use only a single vendor's technology does not make sense in the enterprise integration space. When an open platform is done right, we are often not even aware of the technologies a customer integrates with the platform. For example, we discovered that one of our larger customers was using TIBCO for messaging and Actional for Web services management, alongside Oracle BPEL Process manager, only when it was time for us to write up a marketing success story. This was because all of these products played nicely with each other during the implementation phase of the project, and beyond.

## Conclusion

If an organization has met the requirements for a Web services/SOA platform, what specific benefits should it expect? Organizations that adopt such a platform may:

- Lower learning curve costs (which include not just product training but also the costs of becoming proficient in new design patterns)
- Require fewer specialized consultants
- Find it easier to build composite applications around existing systems and technologies
- Have a more agile IT organization and a lower

cost of managing and maintaining production apps

I'd like to take a moment to defend this last point, because I think "IT agility" is the Holy Grail for why organizations should adopt a service-oriented architecture and a Web services platform that supports the related standards. As mentioned, integration is not a new problem. In the past, organizations had to choose between locking themselves into a proprietary integration technology and building their own homegrown SOA layers. Adoption of the commercial integration technologies was slowed, due to cost, complexity, and vendor lock-in issues. Typically, the homegrown approach was done in Java, on top of message-oriented middleware, and it often worked – *except* it required very smart architects to pull it off. New developers had a very hard time understanding how the system worked, and in general, complexity increased exponentially as the business evolved.


Now however, an SOA architecture and a Web services platform provide architectural blueprints and more explicit data models, interfaces, and process logic without the associated problems of proprietary integration technologies. With this architecture, developers need to be familiar with things like BPEL, XML Schema, WSDL, and messaging – not easy tasks, but becoming familiar with them is easier than becoming super gurus in a single vendor's technologies and APIs (and much easier than locating and hiring such vendor-specific super gurus). Over the past several years, we have seen Java/J2EE standards, and the open platforms that support them, enable mainstream developers to easily build code-centric, multitiered Web applications. In a similar fashion, SOA standards, and a Web services platform that supports them, make composite, process-centric applications easier for mainstream developers to implement and allow these systems to evolve and change at the same pace as the business. ©

## About the Author


Dave Shaffer is Oracle's director of product management for Oracle BPEL Process Manager. He has held consulting, product management, and software development roles over the last 15 years at a wide range of technology companies, including Collaxa, Apple Computer, NeXT Software, and Integrated Computer Solutions.

■ ■ ■ [david.shaffer@oracle.com](mailto:david.shaffer@oracle.com)





App Dev: "It must be a database problem."



DBA: "I suspect it's your application server code that's causing the problem."

## Eliminate the finger pointing with WebLOAD Analyzer™.



Get a FREE "No more finger pointing" T-shirt. > >

### Want to accelerate resolution of Web performance problems?

You can with WebLOAD Analyzer—the only load testing tool with root cause analysis for .NET and Microsoft Web applications. WebLOAD Analyzer employs unique "black box" software to capture actual problems at multiple synchronized levels, enabling users to drill down to the individual component. By pinpointing the root cause of all kinds of application problems, WebLOAD Analyzer reduces problem resolution time by 60%—significantly improving application performance and eliminating finger pointing once and for all.

> > Learn more at [www.radview.com/analyze](http://www.radview.com/analyze)



The Smart Choice in Web Application Testing

© RadView Software, Limited. All rights reserved.

# Where HTTP Fails SOAP

Alternative choices for situations when HTTP does not scale sufficiently for enterprise Web service deployments

■ Web services allow for the delivery of SOAP messages over any protocol. A common misconception is that all SOAP messages must be transmitted over HTTP. While that approach is useful in many cases, there are situations where it makes sense to use alternatives. This paper investigates situations where HTTP does not scale sufficiently for enterprise Web service deployments and looks at available alternatives.

## HTTP and Scalability

HTTP was designed for serving Web pages under the assumption that the protocol would only be required to send a request and receive a response. This paradigm has worked very well for the World Wide Web and has been ubiquitously accepted as its standard protocol. When a person makes a request on an interactive Web site, they are typically interacting with an application server tier (J2EE, .NET, or scripting languages such as Python and Perl). Each person is running a browser client and doing only one thing at a time on that particular site. In a typical enterprise, however, an application server fronts a number of back-office systems that provide critical business services. The application server usually supports a number of users concurrently. This implies that it typically needs to make a number of concurrent requests on those back-office systems.

Many people believe that a move to service-oriented architecture (SOA) implies a move to SOAP/HTTP as the ubiquitous protocol throughout the enterprise. What few seem to



WRITTEN BY  
**FRANK  
LYNCH**



WRITTEN BY  
**MARK  
FYNES**

realize, however, is that the SOAP/HTTP approach has inherent scalability limitations under certain circumstances. Simple Web browsers have been the de facto HTTP client to date, and they are in essence single-threaded clients as far as the server is concerned, making only one request at a time over a given connection. This has created a perception that

HTTP can be scaled as needed. To date, it has been scaled only for communication between browsers and application/Web servers, typically through clustering, replication, and the use of hardware load balancers. Unfortunately, scaling communications between an application server and back-office services cannot be

solved satisfactorily using the same techniques.

For example, assume we have an online banking system with support for up to 4000 concurrent users. The Web tier comprises a cluster of application server instances behind a hardware HTTP load balancer. In order to fulfill the online banking business function, there are three Unix-hosted services and a mainframe-hosted service utilized by the application server. In a world where SOAP/HTTP is the only protocol, the application server will have to support an incoming connection from the browser, and one additional connection out to each of the four back-office services *for each concurrent user*. This is because HTTP demands that you wait for a response before you send your next request over that same connection. It has no concept of a request identifier, which is a core requirement to enable connection sharing.

One could of course just serialize the requests, awaiting each response before sending the next request on each given connection. However, this is a waste of resources because the back-office server is not doing anything with this connection until the response is sent back to the client application, and most back-office systems have the capacity to handle multiple concurrent requests.

Interleaving requests over a single connection would be the ideal. It would allow an enterprise to achieve the same level of concurrency while using fewer resources. One would send a number of requests to a server over a single connection and receive responses as they become available. The client can correlate responses based on a request identifier. This would allow responses to be returned as soon

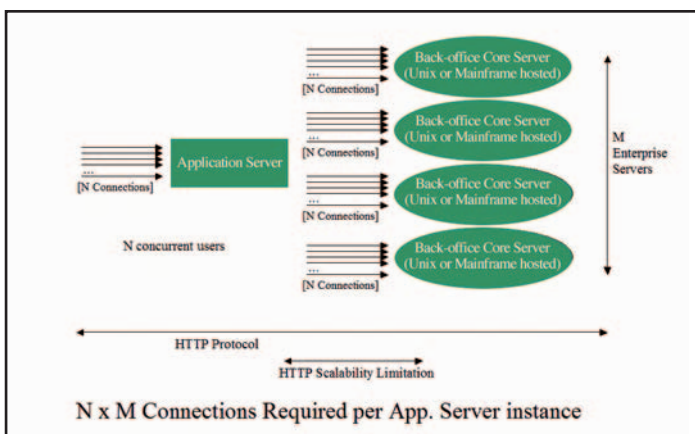


FIGURE 1 This diagram illustrates the significant number of connections required for highly concurrent processing in an HTTP environment



as they are ready (which may differ from the order in which requests were sent). Unfortunately, the HTTP specification forbids such interleaving.

The obvious conclusion is that a standards-based protocol that allows for request interleaving is needed. This would allow the sharing of a single connection between the application server and each back-office system. In the example previously outlined, if the application server had an upper limit of 1000 connections it can open at once (file descriptor limit), in our SOAP/HTTP world, each application server would be limited to concurrently supporting only 1000/5, or 200, clients. A typical workaround for this problem is to add application servers. If enough are added to support 1000 clients, the problems propagate into the back-office servers, which are now maxed out on the number of connections they can keep open. Creating pools of back-office server instances is prohibitively expensive, especially if they are hosted on a mainframe.

This problem has been solved in the past with connection concentrators, but because we cannot interleave HTTP POST requests, *HTTP-based communication cannot be concentrated*. Clearly, HTTP is not capable of scaling in such an environment.

HTTP 1.1 supports a feature known as "request pipelining." Pipelining allows a client to send multiple requests over a given connection without having to wait for each response. However, it is not as useful as interleaving, as the HTTP/1.1 specification (see the first entry in the References section) mandates that: A client that supports persistent connections MAY "pipeline"

its requests (i.e., send multiple requests without waiting for each response). A server MUST send its responses to those requests in the same order that the requests were received."

Pipelining was designed to streamline the downloading of elements within Web pages over the Internet, supporting only HTTP request types that may be reissued without any change to the server state (idempotent requests). The HTTP/1.1 specification is very clear about this point: "Clients SHOULD NOT pipeline requests using non-idempotent methods or non-idempotent sequences of methods (see section 9.1.2)."

Web services, on the other hand, typically use the HTTP POST request type, which can be non-idempotent. Therefore, pipelining cannot be used.

Roy T. Fielding (the primary architect of the HTTP/1.1 protocol) spoke at ApacheCon November 2002, about a new protocol that he is working on called "Waka." In his presentation (see the second entry in the References section) he described Waka as "...a new protocol to solve HTTP's current problems in a generic way."

He went on to mention support for interleaved data and metadata delivery. Waka has not yet been fully specified, so the details on how Waka intends to support interleaving are not yet available. You can track the progress of Waka at the project Web site (see the third entry in the References section). At this time there are no implementations of Waka available.

## Scaling Web Services (SOAP) in the Back Office

Clearly, a protocol is needed that allows the interleaving of requests over a single connection. HTTP could be extended to support request identifiers, but modifications to this standard have taken years to be accepted because of the sheer number of deployments. Solving this problem within the bounds of a SOAP-based specification – WS-ReliableMes-

saging, for example – will always be subject to the limitations that HTTP imposes. A variety of alternatives to HTTP exist. Some are described below.

## MQSeries

MQSeries is a widely deployed enterprise messaging system from IBM. It has been in production for many years and has proven its robustness and scalability in enterprise deployments. It has traditionally been used in single-threaded applications (based on the age of many deployments), but there is no reason why an application could not have multiple threads posting to the same queue, even before responses are read back from a reply queue. This would solve the problem described, but MQSeries is proprietary and expensive. It is much better suited to asynchronous communication, and demands a pair of queues for pseudo-synchronous communication.

## JMS

JMS is a standard messaging interface designed as part of the J2EE specification. It does not specify any details about wire-level implementation, so two separate JMS implementations are unlikely to interoperate. With a JMS-based solution, therefore, all communication must take place using the same implementation; however, the API is widely accepted and adopted, and it's very friendly to the J2EE domain. Just like MQSeries, JMS is asynchronous, thereby allowing interleaving of requests. JMS is completely Java-centric, but many of the core back-office systems in production today are not, which means that successful integration of these systems with JMS can prove to be a challenge.

## IIOP

The Object Management Group (OMG) adopted the Internet Inter-ORB Protocol (IIOP) as part of the CORBA 2.0 specification. A number of groups have adopted IIOP as their standard protocol, not least of which was Sun Microsystems, who adopted it as the standard protocol for Java RMI. Given its CORBA heritage, a number of IIOP implementations exist in a variety of languages. The variety of available IIOP implementations covers the range of commercial, free, and open source software. Most implementations of IIOP have matured to a

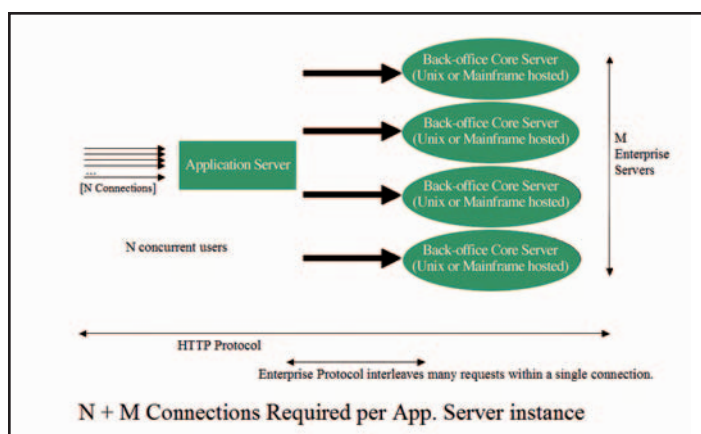


FIGURE 2 This diagram illustrates an environment not bound by the limitations of HTTP (permitting connection sharing in the mid-tier)

point where they interoperate seamlessly with each other, and IIOP has proven itself in some of the most demanding environments such as telecommunications provisioning and network management. IIOP offers support for multiple qualities of service, including optimal delivery of large messages over TCP/IP (which would be ideal for SOAP). Just like HTTP it natively supports a request-reply paradigm, but in addition it allows for the interleaving of requests, replies, and fragments thereof, all over a single connection.

## SOAP over IIOP

IIOP presents a very strong case for adoption as the protocol of choice within the back office. It's one of the few standards-based protocols (thus offering a wire-level interoperable transport) proven to scale in the enterprise. If one wanted to build a Web services framework that supported SOAP messages over IIOP, integrating open-source projects such as Apache AXIS with the JDK IIOP stack might do it.

## The Need for an Enterprise Service Bus

A key difference between a Web services toolkit and an Enterprise Service Bus (ESB) is the ability to switch message format and protocol as necessary. For the sake of this discussion, we are talking only about switching the underlying protocol used to deliver SOAP messages, and we're working under the assumption that there is no need to integrate with legacy systems that expose endpoints with other message formats. The programming model should insulate developers from the protocol and transport, which should instead be a deployment option as opposed to a decision made at development time.

Most ESBs, with minimal additional development effort, allow you to build distributed systems that communicate using any of the following:

- SOAP over HTTP
- SOAP over MQSeries
- SOAP over JMS

However, SOAP is only one among many protocols with which enterprise applications need to deal.

An ESB must also allow you to expose your business logic over more than one protocol/

transport. In particular, it should be capable of exposing the endpoint over an enterprise-strength transport without sacrificing support for SOAP/HTTP. The vast majority of data (in the back office) should be transmitted over an enterprise-strength transport while still allowing for use of SOAP/HTTP where applicable. The forthcoming WS-Addressing standard from W3C will support metadata within its endpoint reference (EPR) construct, thus making it possible to describe and reference endpoints regardless of the protocol and transport used.

In some cases, however, you will not be able to control the volumes of requests coming from SOAP/HTTP-based clients. In such situations the ESB should provide you with a relay that accepts SOAP messages over the HTTP transport and sends them over the enterprise-strength transport to the ESB-enabled back-office server. This is effectively a concentrator implementation, instance-pooling inexpensive relays (typically behind a hardware HTTP load balancer) rather than attempting to create pools of expensive and complex back-office systems.

In the past there was a barrier to accessing back-office systems in the learning curve associated with the middleware technologies involved. With the advent of Web services, this barrier has been lowered substantially; this is a step in right direction and is essential for effective deployment of an SOA. An ESB insulates the developer from the middleware used in deployment. Given this empowerment granted by the ESB, the traditional vendor lock-ins in the back office can be removed. To be truly services oriented you should not be beholden to any individual middleware vendor, regardless of the scalability requirements. Most ESB vendors will provide proprietary alternatives to HTTP where scalability requirements demand it, e.g., IBM is encouraging the use of MQ to deliver SOAP messages. Alas all direct consumers of the service need to have this vendors' technology installed. Your ESB should provide you with a higher quality standards-based alternative, such as SOAP/IIOP.

## Conclusion

SOAP/HTTP has one major thing going for it: its ubiquity and widespread support. While HTTP does a great job at serving Web pages, it is not an enterprise-strength protocol, and does not scale well in the back office. Clearly

an open, interoperable, standards-based, enterprise-strength protocol is needed here. The most widely deployed protocol that fulfills all of these criteria today is the OMG's IIOP.

One thing an ESB allows you to do that a Web services toolkit cannot is adopt an SOA using SOAP without sacrificing the qualities of service required in the back office. In other words, an ESB allows you to apply SOAP where it fits best without forcing you to also apply it to problems for which it's a poor solution. An ESB should allow the developer to preserve the loose coupling that SOAP affords us, while taking advantage of the qualities of service demanded in the back office that are available with IIOP.

## References

- R. T. Fielding, et al. "Hypertext Transfer Protocol -- HTTP/1.1", Internet RFC 2616, June 1999: [www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8.1.2](http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8.1.2)
- R. Fielding, *ApacheCon presentation*, November 2002: [http://gbiv.com/protocols/waka/200211\\_fielding\\_apachecon.ppt](http://gbiv.com/protocols/waka/200211_fielding_apachecon.ppt)
- Waka protocol progress page: [www.apache.org/~fielding/waka/](http://www.apache.org/~fielding/waka/) ©

## About the Authors

Frank Lynch is a principal services engineer with Iona Technologies. He holds B.Sc. in applied computing from Waterford Institute of Technology, Ireland. Frank has 10 years of experience in the software industry, primarily working with distributed systems technologies. He has worked in a wide range of roles including product development, professional services/consulting, sales, and support. He has advised some of the world's largest financial services and telecommunications firms, and spends considerable time with Global 2000 companies as a distributed systems technologist.

■ ■ ■ [frank.lynch@iona.com](mailto:frank.lynch@iona.com)

Mark Fynes is a senior systems engineer at Iona Technologies. Previously he had been a freelance consultant, carrying out a number of roles including development training and implementation of Web-based staff-management systems. Mark secured his Bachelors degrees in both Arts and Engineering from Trinity College Dublin, Ireland with honors. He then joined the Computer Architecture Group at TCD and secured a research-based Masters in Science. During his time as a post-graduate researcher at TCD, he participated in an ESPRIT-funded project to provide video-on-demand with the hotel industry as an early adopter, focusing his work on remote systems management.

■ ■ ■ [mark.fynes@iona.com](mailto:mark.fynes@iona.com)



Offer subject to change without notice

# Learn Web Services. Get a New Job!

## Only \$69.99

1 year (12 issues)\*

\* Newsstand price \$83.88 for 1 year



**Get Up to Speed  
with the Fourth Wave in  
Software Development**

**Subscribe  
ONLINE**  
[www.wsj2.com](http://www.wsj2.com) or  
**CALL**  
**888 303-5252**

Offer subject to change without notice

**The Best  
.NET  
Coverage  
Guaranteed!**

## Subscribe today to the world's leading Web Services resource

- Real-World Web Services: XML's Killer App!
- How to Use SOAP in the Enterprise
- Demystifying ebXML for success
- Authentication, Authorization, and Auditing
- BPM - Business Process Management
- Latest Information on Evolving Standards
- Vital technology insights from the nation's leading Technologists
- Industry Case Studies and Success Stories
- Making the Most of .NET
- Web Services Security
- How to Develop and Market Your Web Services
- EAI and Application Integration Tips
- The Marketplace: Tools, Engines, and Servers
- Integrating XML in a Web Services Environment
- Wireless: Enable Your WAP Projects and Build Wireless Applications with Web Services!
- Real-World UDDI
- Swing-Compliant Web Services
- and much, much more!





# Open Grid Service Architecture (OGSA)

SOA journey: from Web services to grid computing

■ A service-oriented architecture (SOA) is based on services that are self-describing and open components that can be used to build distributed applications. A service is implemented by a software module that responds to queries and commands by performing a specified function. There is a large degree of standardization in the operation of Web services.

In this article I introduce the concept of a service domain (SD) to be used as the major building block for implementing SOA in large computing grids in which tens or hundreds of services are offered to customers. The following are SD-related terms used in this article:

- **Similar:** Two service instances are similar when they implement the same functionality but provide a different API or different service level characteristics.
- **Compatible:** Two service instances are compatible when they implement functionalities that overlap.
- **Related:** Two service instances are related when they implement functionalities in the same category. They may be disjoint and thus augment each other.
- **Virtual Service (VS):** VS is the service offered



WRITTEN BY  
**RAMY  
ABAAS**

by an SD that results from aggregating several service instances.

- **Virtual Port (VP):** VP is the port visible to the SD clients, which is implemented by a mapping to the port of a service instance.

## SOA Challenges

Guarantee of service and integration are considerations that illustrate and assess the challenges involved in using Web services in environments where tens or hundreds of services are offered to customers.

- **Guarantee of Service:** If a service bound to a client fails, there is no easy process for the client to switch to a comparable service. The client must be able to take the following actions.
  1. Query the service registry
  2. Locate an available service

3. Establish a new binding
4. Back out from the point of failure
5. Rerun the service request

It is also likely that there are several service instances from which to choose and the client must include the logic to select an appropriate service instance. The logic becomes complex and must to be included in each client. The situation will become unmanageable if hundreds of services need to be handled in this manner. Although recovery services and highly available platforms are often provided on the service provider end, clients are still required to handle exceptions, because SOA enables dynamic mixing and matching of requestors and providers. If the service providers are independent businesses, the client may view the services that are provided as being less reliable. Availability will continue to be a critical consideration for stand-alone service implementations.

- **Integration:** In today's capricious business environment, Web services are ideal for implementing either internal or external business functions. When businesses are forming partnerships and alliances, the number of suppliers can grow as businesses expand. The characteristics of suppliers can also become more diverse. For example, in a catalog sales scenario, the suppliers could



specialize in different classes and categories of merchandise, carry merchandise of different quality levels, and adhere to different volume-discount agreements. Suppliers may come and go, may fail on their deliveries, and may have relationship constraints, and so on. There is no fast and easy way to manage these changes dynamically in real time. The main business processes can be implemented as workflows, but keeping the processes robust may depend on the performance of specific suppliers. The logic becomes unmanageable when tens and hundreds of suppliers need to be managed. This could be one reason why organizations tend to implement SOA with only a small number of suppliers.

Environments with hundreds of competing or collaborating suppliers will be a common phenomenon in the near future. Today, some companies offer clients tens or hundreds of services, and new techniques to overcome complexity are required in order to achieve the full benefit of an SOA in this type of environment.

#### **Service Domain (SD)**

An SD can overcome these challenges. An SD maps a collection of comparable or related services to a single logical service. A service domain architecture that uses a set of rules for managing the collection of services and that automatically dispatches the “best” service available to satisfy user requests is necessary. The architecture contains built-in autonomic properties in which an SD implementation monitors the events within its control and triggers adjustment actions in its member services, including recovery from service failure and handling of topology changes. The SD node consists of the following:

- A service entry component
- A rule-based policy component
- A service catalog
- A service-rendering component
- An aggregation engine

The SD presents a single Web-services interface that is describable by standard Web services specifications, but it provides a higher-level interface for aggregating and managing a group of Web services as a single virtual ser-

vice. It offers an SOA solution that reduces the complexity of building business applications because the applications need only focus on the services and user interfaces (UIs) specific to the business, while the SD provides most of the enabling logic. A client needs to find and bind once to an SD. The SD hides the complexities involved with using services: discovery, selection, exception handling, etc.

The SD model enhances the Web services concepts of proxy, interceptor, handler, gateway, mediator, and broker by incorporating concepts of grid computing and autonomic computing. Instead of implementing individual functions for each service, such as action selection, routing, failover, and change management, the SD model provides a service grid – a pool of dynamically assembled service instances. The SD automatically dispatches the best service instance corresponding to the

services in the group. It also does not support rule-based automatic service aggregation. The emergence of the Web Services Resource Framework (WSRF) and the associated WS-Notification are bringing together grid computing and Web services.

#### **Open Grid Service Architecture (OGSA)**

The Open Grid Service Architecture (OGSA) defines key building-block services for grid computing. Given the independent nature of its service abstraction, whether it is for a specific IT resource, system service, software solution, or business-application function, the SD model could become a key building block for the OGSA fabric as well.

The SD manages multiple software sites as a grid and is analogous to the concept of grid computing as a hardware resource balancer

“Today, some companies offer clients tens or hundreds of services, and new techniques to overcome complexity are required in order to achieve the full benefit of an SOA in this type of environment”

user request through built-in mechanisms for registering and discovering service instances.

SD presents a Web services collection model that can contain a group of abstract Web services objects. This approach, which provides a convenient way to group Web services, is useful for designing a distributed access-control mechanism because it facilitates the management of authorization specifications and metadata, as well as the composition and specialization of Web services. This model does not support rule-based automatic service aggregation. The Service Group port type is defined as an interface to a collection of grid services. The grouping model does not assume any relationship among the member

and job scheduler. Service providers offer services under terms and conditions, and their services are accessed via application calls and not at the hardware level.

An SD has a set of port types representing the set of distinct services it offers. Each port type corresponds to a collection of similar service instances. Rules are provided to manage and control the behavior of the aggregated services.

Each port type is associated with a set of service level agreements (SLA), each specifying a service level. In order to offer a service to clients, the service provider registers the service with the SD, specifying the port type and an SLA. The SD can then incorporate that service into its

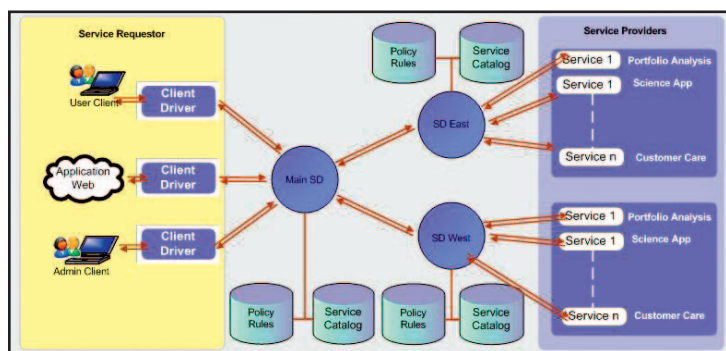


FIGURE 1 SD Topology

own version of service. When client (Requestor) requests a service, the SD matches the client service level with the provider service level and selects the provider. The two service levels are compatible but need not be identical. By using SDs, a service broker is thus established that can manage a heterogeneous group of service suppliers in order to provide virtual and additional value-add services to consumers.

### Service Domain Hierarchy

An SD implementation may have a nested architecture consisting of a hierarchy of SD nodes. In that case, a service requestor interfaces with a single logical SD. The SD implemented as a hierarchy of nested SDs allows the creation of a large virtual business complex referred to as a service grid. The motivation behind the SD concept is to achieve manageability when dealing with a large number of services and service providers. Although we consider how the SD can exploit existing standards, products, and emerging technologies, the main focus is on reducing the complex issues to simple Web service invocations and thus view Web services as the ultimate standard.

Figure 1 shows the topology of a logical SD hierarchy. It is a simple topology that consists of a *main* SD node and two *secondary* SD nodes: SD East and SD West. Service instances of various types, such as portfolio analysis, and customer care applications, are provided through the two secondary SD nodes.

The service instances that are similar are aggregated on the main SD node and presented as virtual ports. Each SD has a set of policy rules that governs its operations and a service catalog (or registry) that stores the information collected about its services. The SD nodes and their services are distributed over a set of hard-

ware resources. The clients interact with the SD through the main SD node using the Web services client interface. Typically, a client driver that includes a Web application server mediates between the user browser and the SD node.

When a client calls the SD for a service, e.g., portfolio analysis, the main SD consults the policy rules and dispatches the request to SD East for processing. SD East will in turn consult its own policy rules to select a service instance to service the request.

The topology of the SD hierarchy is self-configuring because it supports automatic expansion or contraction of the tree, as SD nodes join or leave the SD hierarchy. A service instance can be assigned to an SD node at any level. Services that conform to a given port type interface are logically aggregated into a virtual port type that is surfaced at the main SD node (at the highest nesting level of the topology).

### Architecture of the SD Node

Figure 2 shows the architecture of an SD node. The following are the main components of the SD.

- **Service entry:** The service-entry component provides standard Web services or other means for accepting requests and returning responses. Aside from service requests from clients, the service-entry component also supports administrative operations for managing the SD node. The latter are viewed as operations directed to the “home” port. For example, an administrative client may send a register command in order to register a service instance with the SD node. The information entered is stored into the service-catalog component using the “standard” registry service interfaces.
- **Service rendering:** The service-rendering component provides an attachment interface that supports any WSDL-described Web service offered by a service provider. The attachment interface allows the inclusion of information beyond WSDL definitions;

we refer to this information as augmented information. The augmented information consists of XML-specified service attributes in a format that is consistent with current standards such as the XML-defined nouns and adjectives, and the early grid specifications of service data elements. These service attributes are useful for service discovery and service aggregation. An example of such an attribute is the service “origin” type used to support a variety of Web services—dispatching approaches, including SOAP, WSIF, grid, Microsoft .NET, and document style, for the service instances. It is stored in the service-catalog component when a service instance is registered. The service rendering component uses the attribute to set up the correct Web services client-proxy call to dispatch the service instance.

- **Service catalog:** The service-catalog component provides an internal mechanism to store the WSDL, service attributes, state (operational or down), and status (collected statistics) data in one place. Where it makes sense to have constituent services sharing state and data, SD can share its service catalog with the individual services. The sharing uses the same “home” port operations that SD provides to an administrative client. The service catalog is built on top of a registry such as UDDI. Several SDs can share a service catalog, as illustrated in Figure 2. The SD node stores the (static) WSDL information in the registry. It stores the augmented information in a supplementary data structure in memory, and it stores the dynamic operational data (status) in persistent storage (a file). Administrative clients use the home port as a higher-level interface that eliminates the need to deal with low-level interfaces such as registry APIs and data formats, and enables operations for configuring and managing registries. For example, a client can make a simple call and obtain the list of registered service instances without the need to specify the registry used. If the registry is based on UDDI, this will be shown as an attribute value that indicates the type of registry. Because the service-discovery function of the SD uses real-time data, it is different from conventional registry queries. The discovery can include service features, business relationships, and



performance characteristics. Both static and dynamic information are maintained in its processing storage.

- Policy:** The policy component provides a policy interface that allows the configuration of all operating rules for the SD. A service policy is a set of XML rules that includes service-level definitions and rules for handling security, recovery, events, discovery, service selection and routing, service mapping, and various business considerations. The XML document is constructed and entered through a setPolicy command to the service-entry component. When the command is processed, the policy rules are entered into the policy component using a "standard" policy service interface. Rules are the centerpiece of the SD model. They differentiate the model from other service management middleware such as registry, grouping, proxy, gateway, and intermediaries. These rules are interrelated; for example, the selection rules are related to the service-level definitions and service mapping; selection errors are resolved by the recovery rules; business relationship-based selections depend on other miscellaneous rules.
- Web services stacks and standards (WSSS):** The SD node is implemented as a WSDL-described service. It is built using grid and Web services standards such as WSDL, OGSA, SOAP, XML, UDDI, WSIL, and WSIF. All dependencies on these standards are limited to the WSSS component so that the evolving nature of the standards will not impact other SD node components. The design of WSSS makes the SD node architecture extendable and able to exploit OGSA-defined services and emerging Web services standards as they mature. As Figure 2 illustrates, three of these components provide external interfaces to the SD node: service entry, service rendering, and policy.
- Aggregation engine:** The aggregation engine obtains policy rules from the policy component and uses them to enforce specific behaviors in the various subcomponents on the request-processing path. For example, the lookup subcomponent communicates with the deployment subcomponent to obtain information about registered service instances. The monitor subcomponent collects various metrics and tracks appropriate thresholds

whose values may affect the selection factors used by the selection subcomponent. The aggregation engine also contains subcomponents for handling exceptions. When the ability of the SD to handle additional incoming requests becomes limited, the offload subcomponent can redirect further requests to peer SDs. Similarly, when a service instance fails, the failover subcomponent can dispatch another service instance by calling service rendering. When a shortage of service instances is detected on a virtual port, the SD can query and acquire service instances from other SDs. All of these actions are controlled by policy rules.

The SD architecture does not impose any initial state requirements or other restrictions on the services beyond conformance to WSDL standards. Any initial state requirements are driven by the implementation of the specific services. A stock quotation, for example, does not require the assignment of an initial state. A job-execution service, on the other hand, is assigned an initial state of "ready." The state will be stored in the service catalog.

When the service-entry component receives an incoming request, it uses WSSS to interact with the aggregation engine. Within the aggregation engine the lookup subcomponent is called first to identify service instances available to process the request. Then the selection subcomponent is called to identify the best service instance for this request.

The service-entry component then calls the service-rendering component to invoke the specified service instance. On completion, a response message is returned to the service-entry component, which forwards it to the requestor.

The SD model starts with what is available today, the existing software assets, and builds a service management and brokering middleware solution designed to address the previously mentioned challenges. Figure 3 illustrates the extension of the existing software assets using the Legacy Modernization. Its objective is not to define new application programming interfaces (APIs) or new standards, but to construct from the existing components a new, higher-level structure that can hide complexities from service users, simplify deployment for service suppliers, provide self-managing capabilities, and give administrators a set of tools for managing the IT solution.

The SD, which implements Web services and grid concepts, is extendable to include new grid standards that are still evolving. SDs often

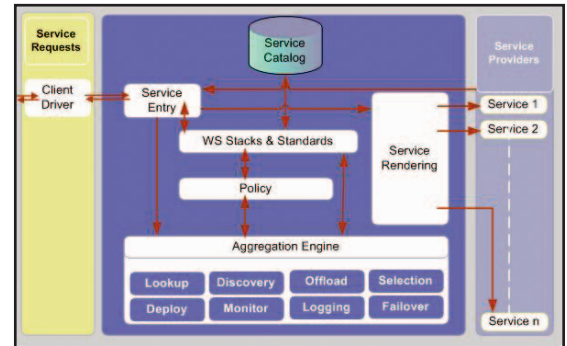


FIGURE 2 SD Architecture

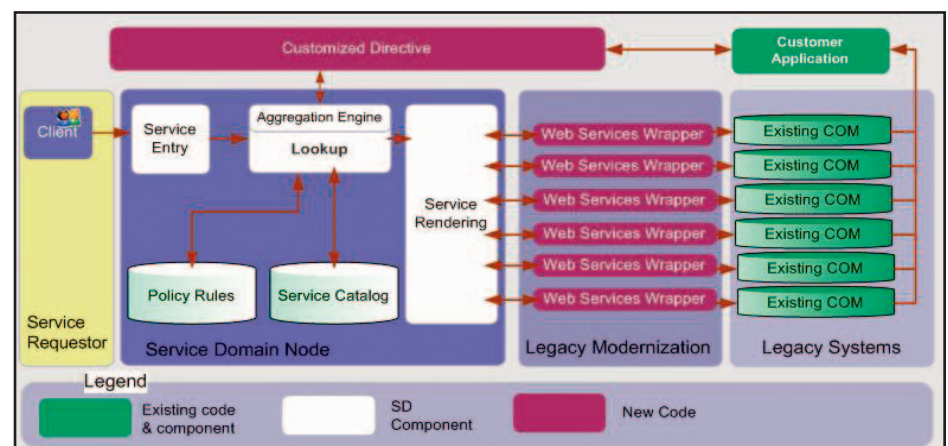


FIGURE 3 Building on the existing assets

can be structured as a logical hierarchy. For example, an SD that provides portfolio management services may direct requests to a descendant (child) SD that provides services for executing the transactions. A further descendant SD may provide resource-allocation services, such as allocation of computing power, storage

Ideally, such a sequence should be handled by a workflow service that the client can invoke and that is able to handle recovery as part of its logic. The recovery can be handled by a rule that listens for a workload service-exception event and then invokes a back-out service to reverse the sequence.

service instances from multiple providers and are not subject to central control.

The providers have their own policies and implementations. By negotiating with the constituent service instances, the SD node produces a service capability that is superior to the mere collection of services. Selecting a service instance to process a service request can be performed in a variety of ways that range from simple to complex, according to rules that can be based on the states of the requestor and the service provider, the request load, the business relationships between participants, and so on. The workflow model should apply Web services to business process management and include a flow definition language and a process choreography engine. The workflow model can be applied to SD nodes directly. In addition the SD model provides simplicity, robustness, and opportunities for lightweight workflow engines. It also allows multiple simple services to be composed into a complex service; in this case, the complex service would be associated with a single port type, and a service requestor would be unaware that the service is actually a composition of simpler services.

SD supports utility computing by generating metering records, enabling usage-based fees to be associated with SLAs, and supporting third-party billing. A hosting manager interface isolates SDs from platform-specific user management, security, instrumentation, and problem-determination subsystems. The interface provides access to local contracting, identity, and metering services.

The SD approach also applies to middleware and system functions. Constituent service instances can use SDs for peer communication and state sharing. Dynamic discovery capability between peer SDs can be implemented by policy rules. The SD hierarchy is a logical concept although it could be built from a physical peer-to-peer network. ©

#### ■ About the Author

Ramy Abaas, enterprise architecture director for Covansys Corporation, a leading global technology services company, holds an MS in CIS as well as multiple Microsoft and IBM Certifications. He has over 16 years of experience designing and developing IT systems for Big Five clients and has served as an IT architect for IBM.

■ ■ ■ [rabaas@covansys.com](mailto:rabaas@covansys.com)

## “ The emergence of the Web Services Resource Framework (WSRF) and the associated WS-Notification are bringing together grid computing and Web services ”

capacity, and other execution-related resources (similar to data and resource management in grid computing). Such an implementation of an SD hierarchy is a service grid.

#### **Stateless and Stateful Web Services**

The SD node maintains a session for each active client. If the service selection rule is set to a “transaction history” directive, the customized directive in Figure 3, the client request is routed to the same service instance. For example, stock quotation requests from a particular user are routed to the financial service that serviced that particular user's previous request for a stock quotation. There are other directives, such as “designated” and “fixed” directives that are associated with the user and extend across sessions. For example, financial requests from a particular user are routed only to those service instances in which that particular user has an account. When a failure occurs, the state can be recovered in a client-transparent way if the failure affected a single invocation of a service, in which case the SD node dispatches another service instance for handling the failed request. If the client invokes a transaction sequence that is not under the control of a transaction manager and a failure occurs, the transaction manager is not able to handle the recovery.

In general, individual service requests can be stateless, but because most services maintain their own states, the result is a stateful application. The SD does not interfere with application-managed state, whether managed by a simple client or by a workflow service. For example, a request for a travel reservation leads to relevant information being saved by the service instance. In a follow-up transaction involving the same service instance, the reservation leads to a purchase.

#### **Summary**

The SD's SLA enforcement of dynamic service-level mapping can be viewed as a self-adapting feature of an autonomic system. Specifically, in service-level mapping, user SLAs are associated with pools of service instances based on the SLA commitments made by the provider at registration time. As runtime information such as the distribution of request types, service availability, and turnaround time becomes available, the pools can be reconfigured based on this information. The Service Grid is a system that coordinates resources that are not subject to central control; it uses standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service. SD satisfies this definition. The resources managed in an SD are WSDL-defined

# Smart Development Environment

## 15th Annual Jolt Product Excellence Award in the Design Tools category

# Winner

*SDE won the 15th Software Development Magazine Jolt Product Excellence Award 2005 in Design Tools category over IBM Rational Software Architect and Borland Together Designer.*



### Visual Paradigm products



### 2004 Quadruple award winning product



Visual Paradigm helps  
**ACCELERATE** the entire  
**MODEL-CODE-DEPLOY**  
process in a **DISCIPLINED** and  
**COLLABORATIVE** way to **EXCEED**  
customers' expectations.

[sales@visual-paradigm.com](mailto:sales@visual-paradigm.com)  
[www.visual-paradigm.com](http://www.visual-paradigm.com)





# Why WSDM Matters

## The role of WSDM in distributed IT management

■ The world of IT management has changed a great deal since the early days of SNMP and network management. IT organizations today are building and deploying a wide range of systems and applications that must be managed in a consistent and reliable way. Applications are being built from the ground up using service-oriented design principles, and an IT manager can no longer look to a single machine to determine the health and availability of the services being delivered. Resources are much more distributed and interconnected, and they are being deployed at an alarming rate. For IT, this poses additional challenges in having to keep track of changes and to build management solutions that can aid in linking business needs to IT.

With the variety of application platforms and technologies in use today, companies demand standardized approaches to managing these heterogeneous platforms. This is where WSDM can play an important role. WSDM, or Web Services Distributed Management, began as an OASIS Technical Committee in February 2003. For the past two years, leading management vendors have been working within the OASIS group to create a set of standards that use Web services technologies for management. OASIS officially approved a set of WSDM standards in March 2005.

In this article we take a much closer look at the various WSDM standards, the capabilities offered by some of these standards, and present some useful case studies that demonstrate the benefits of WSDM. The intention is for software architects and developers who design software that must be managed to gain a much better appreciation for the value of WSDM.



WRITTEN BY  
**CHRIS  
PELTZ**

### WSDM as a Management Standard

To understand WSDM, it's important to make a distinction between two key specifications that were developed in the OASIS WSDM TC.

- The Management Using Web Services (MUWS) specification defines how *any* IT resource can use Web services technologies for exposing manageability interfaces.
- The Management Of Web Services (MOWS) specification builds on MUWS and specifically addresses how a Web service resource can be managed.

In this article we'll focus primarily on MUWS because it is the foundation for managing distributed IT resources, regardless of the underlying

platform or implementation technology. As Figure 1 illustrates, IT management involves both manageable resources and a manageability consumer. A manageability consumer such as an Enterprise Management System (EMS) might need to discover a resource, receive notifications of state changes, or control a resource. WSDM MUWS defines a mechanism for exposing a manageable resource with a Web services endpoint that contains control, attribute, and event capabilities.

Before providing details of the capabilities offered by WSDM, it's important to clarify the relationship to existing management standards and technologies. Because WSDM is based on Web services technologies, it relies on standards such as XML Schema, WSDL, SOAP, and WS-Addressing.

At the same time, WSDM MUWS does not impose any specific management model in its implementation. For example, DMTF's Common Information Model (CIM) provides a common definition of management information for systems, networks, and applications. Work is already underway to map CIM resource models and related operations using WSDM-MUWS (see References section, "Proposal for a CIM mapping to WSDM").

Additionally, we shouldn't think of WSDM as a management instrumentation technology. It is expected that developers would still instrument their application with JMX, WMI, or other management technology. In the case of Java, Apache open source toolkits already exist today that allow a developer to expose Java-based applications with WSDM interfaces (Apache Web Services Muse: <http://ws.apache.org/ws-fx/muse/>). Additionally, the Java community is in the process of creating a Web services connector that supports JMX (JSR 262: Web Services Connector for Java

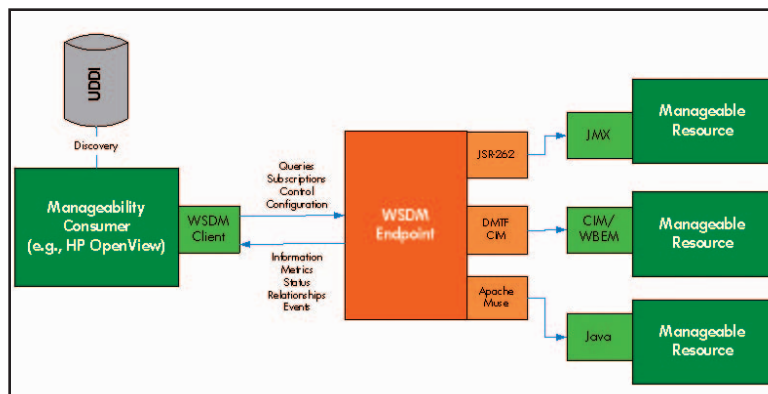


FIGURE 1 Using WSDM MUWS to manage distributed IT resources



CONFERENCE:  
**AUGUST 8 – 11, 2005**

EXPO:  
**AUGUST 9 – 11, 2005**

Moscone Center West  
San Francisco, CA

WHERE **open minds** MEET > >

explore > >   analyze > >   gain > >

LinuxWorld Conference & Expo is the world's leading and most comprehensive event focusing on Linux and Open Source solutions. At LinuxWorld, see and learn how to best leverage the technology for your organization.

- > **Explore** your options on the exhibit hall floor, which features the world's leading hardware and software vendors.
- > **Analyze** the latest Linux and Open Source technology and discover how companies across the globe can show you how to achieve higher profits and increase productivity.
- > **Gain** knowledge about best practices and solutions by attending LinuxWorld's outstanding educational program.

It's the Linux & Open Source event you can't afford to miss!



[linuxworldexpo.com](http://linuxworldexpo.com)



> Register Online With **Priority Code: D0109**

PLATINUM SPONSORS



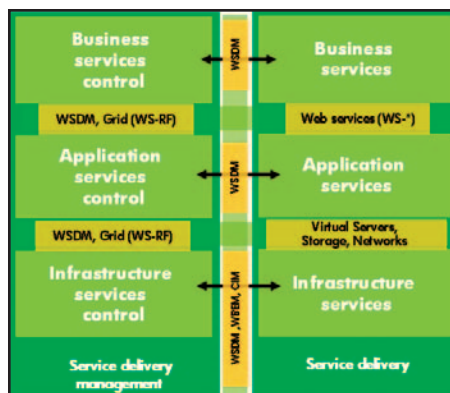


FIGURE 2 The importance of standards

Management Extensions [JMX] Agents: <http://www.jcp.org/en/jsr/detail?id=262>). In either case, the interface one uses for accessing a manageable resource (e.g., WSDM) can be considered separate from the technology used for instrumentation (e.g., JMX, WMI, Java, etc.).

Having these and other adapters in place will help to broaden widespread adoption of the technology. Such adapters will also make it easier for a management system to manage IT resources in a consistent, standards-based manner.

## Why WSDM?

Before we dive into the details, let's better

understand why WSDM is important to IT management. Because WSDM is based on Web services, you can leverage much of the basic infrastructure and tooling support already available. Using WSDM for management can take advantage of improved integration and interoperability, as well as support the basic Web services capabilities such as security, reliability, and transactions.

There are also specific benefits that can be gained for providers and consumers of IT resources.

- For the providers of IT resources (e.g., an infrastructure provider such as a J2EE application server), WSDM MUWS offers a single interface to manageability, regardless of the instrumentation. The same interface can be reused across multiple management system vendors, thus minimizing the number of custom interfaces needed.
- For consumers of IT resources, WSDM MUWS greatly increases the types of IT resources that can be managed. Additionally, discovery of new resources and capabilities can be done automatically at runtime.

Moving forward, we see WSDM as a key enabler to the Adaptive Enterprise. By that we mean the ability for an organization to quickly and easily respond to change through the linkages that are established between business and IT. Accomplishing this requires a cleaner separation between the services being delivered and the management of those services.

Additionally, an adaptive architecture would be one that is modular so that layers of the infrastructure build on each other. As Figure 2 shows, we could achieve this through separation of business services, application, and infrastructure layers.

With this kind of architecture, we can apply WSDM in the interactions between the services and the management control points. Furthermore, we can use WSDM to virtualize up the stack from infrastructure to application to business services. The end result is a more modular, standardized approach to managing IT. With WSDM, an IT organization can begin to more easily drive change from the business down to IT.

## The WSDM Technology Stack

WSDM MUWS outlines several interfaces that can be defined for a resource (see Table 1). We refer to these interfaces as *Manageability Capabilities*. There are capabilities for resource identity, as well as resource correlation to assert uniqueness among resources. Additional capabilities exist for describing relationships among resources and for receiving notifications when resources get created. These capabilities offer a consumer the ability to discover and relate deployed applications and services.

Given a set of discovered resources, a consumer would need to be able to manage and monitor it. MUWS defines capabilities for gather-

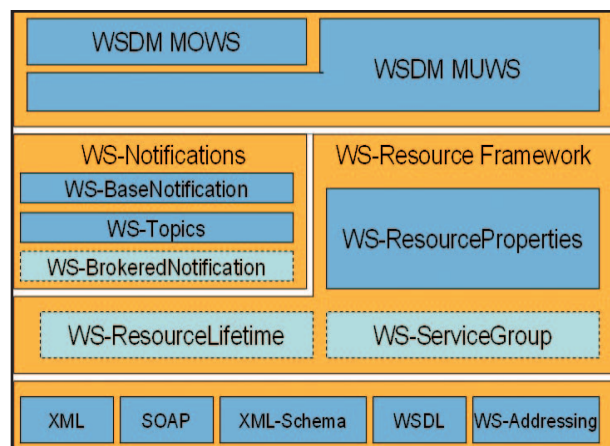


FIGURE 3 The WSDM technology stack

Capability:	Questions Answered:
Manageability Char.	What capabilities are supported?
Identity	How do I uniquely identify a resource?
Description	How do I describe and version the resource?
Correlatable Props	How do I assert equality between resources?
Metrics	How do I get performance and other statistics?
Configuration	How do I monitor and configure a resource?
State	How do I define a state model for a resource?
Operational Status	How do I determine the resource status?
Advertisement	How do I get notified on resource creation?
Relationships	How are resources associated to each other?

TABLE 1 MUWS manageability capabilities



ing performance metrics and statistics, as well as interfaces for configuring a resource. Operations are exposed for defining the state model for a resource and for querying its operational status.

To better understand how these capabilities are addressed, let's examine the specifications upon which WSDM depends. As Figure 3 shows, WSDM MUWS builds on two additional sets of specifications: WS-Resource Framework (WS-RF) and WS-Notifications (WS-N). WS-RF provides a set of mechanisms for accessing stateful resources exposed via Web services. WS-RF uses WS-Addressing to provide Web services endpoints for stateful resources. WS-RF also defines the WS-ResourceProperties specification, which defines a way to represent, advertise, and access the properties that a stateful resource exposes.

The WS-N family of specifications defines a notification model for Web services. As part of this model, WS-N includes the WS-BaseNotification specification. This specification defines the set of messages used to implement a publish-subscribe interaction model for Web services. Additionally, WS-N includes the WS-Topics specification, used to represent and characterize items of interest for notifications. Implementations of WSDM MUWS would leverage WS-BaseNotification, WS-Topics, and WS-ResourceProperties, along with a number of core Web services specifications (XML-S, SOAP, WSDL, etc.)

## A Closer Look Inside WSDM MUWS

Let's now take a closer look at how WSDM can be used to manage an IT resource. We'll specifically look at resource properties, state and metrics, events and notifications, and resource discovery.

### Resource Properties

It's a common requirement to retrieve properties for a manageable resource or to locate a specific resource that matches a given set of criteria. WS-ResourceProperties is the key specification that would be used to read and write properties for a stateful resource. For example, management of a disk drive might require exposing certain characteristics, such as the number of blocks and the block size for the device. This could be modeled in XML as follows:

```
<GenericDiskDriveProp xmlns:tns="http://disk.com/diskDrive" >
```

```
  <tns:NumberOfBlocks>22</tns:
  NumberOfBlocks>
  <tns:BlockSize>1024</tns:BlockSize>
</GenericDiskDriveProp>
```

A consumer that needs to retrieve this property could use the *GetResourceProperty* operation as follows:

```
<wsrp:GetResourceProperty xmlns:dd="http://
disk.com/diskDrive" >
  dd:BlockSize
</wsrp:GetResourceProperty>
```

A consumer could also use the *QueryResourceProperties* operation to locate a disk drive that had more than 20 1K blocks:

```
<wsrp:QueryResourceProperties>
  <wsrp:QueryExpression>
    boolean( /*NumberOfBlocks > 20 and /*/
    BlockSize=1024)
  </wsrp:QueryExpression>
</wsrp:QueryResourceProperties>
```

### State and Metrics

State and metrics are also important for controlling the operational state and determining the overall health of a resource. The WSDM State Capability offers a framework for defining a state model for a resource. A defined model makes it easy to represent transitions between states and to control a resource's behavior. Figure 4 shows a very simple state model that could be created for a given resource.

Additionally, *StateCapability Topics* can be used for sending events when a resource state changes. In this manner, a management system can automatically determine whether a resource is in a healthy, active state.

MUWS metrics can also be useful in monitoring the overall performance and health of a resource. These metrics, represented by a number of different data types, can be collected for a resource at defined time intervals. MUWS provides attributes such as *ResetAt*, *LastUpdated*, and *Duration* for managing the metric collection process. There is also a *CurrentTime* resource property, which can be used for time synchronization during data collection.

### Events and Notifications

Besides querying for and configuring resource-

es, a WSDM consumer can receive event notifications when certain changes occur to a resource. The notification mechanism allows management events to be transported in a consistent manner, relying on the specifications defined within WS-Notifications. The WS-BaseNotification and WS-Topics would be used for defining message patterns and categories for the appropriate publish-subscribe interactions.

A specific event can be described using the *ManagementEvent* tag. Listing 1 shows a simple example of management event where a resource is associated to that event (via the *SourceComponent* element). The *Situation* element is used to describe a specific type of event situation, status, time that the situation occurred, and a relevant

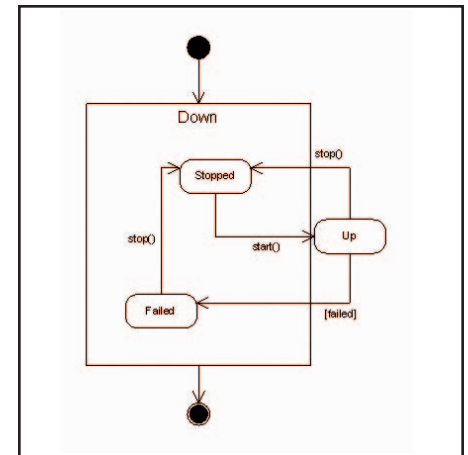


FIGURE 4 Example MUWS state model

event message. Consumers can thus more consistently receive updates of status changes to resources being monitored.

### Resource Discovery

WSDM supports the discovery of resources through Advertisements and Relationships. With WSDM Relationships, you can model the associations that exist between system resources (e.g., a disk drive attached to a host computer). Relationships can be used to auto-discover a set of IT resources, given a root element specified with WS-Addressing. Querying the *Relationship* resource property of a resource identifies known relationships.

A consumer can be configured to receive notifications of new relationships through the *RelationshipCreated* topic. Another capability, WSDM Advertisement, makes it possible to notify a consumer when new resources are created.

With these capabilities, you can keep your inventory of IT resources and relationships up-to-date, thereby enabling real-time configuration management and better insight into the impact of IT on the business. To illustrate this even better, let's turn our attention to two WSDM-based case studies.

## Case Study #1: Determining IT Impact on the Business

Today, there is an ever-growing complexity in managing Web services and business processes. Management needs for this type of system include the need to understand how business processes relate to the underlying IT infrastructure. If a business process is running slow, you may need to trace the root cause to a database problem. You may want to determine what business processes are impacted by infrastructure degradations. Or, you might want to determine what business users are impacted when a network is overutilized.

how we might design a solution that uses WSDM-based technologies. First, we need to make it easy to query for and get updates on the management model exposed by the BPM tool. Specifically, we could build a WSDM gateway that could interact with the native management interfaces and translate them into WSDM-based interfaces. These WSDM-based interfaces would expose a specific management model and provide notifications on changes to that model and events when problems occur.

As Figure 5 shows, an EMS could then consume this WSDM interface via a WSDM-EMS bridge. The management system could consume the BPM model and dynamically update graphical views representing the underlying business services. This dynamic capability would provide an IT operator with a real-time view into the state and health of deployed business processes. Furthermore, since the operator is also monitoring the IT infrastructure, automated correlation could be performed between the business events

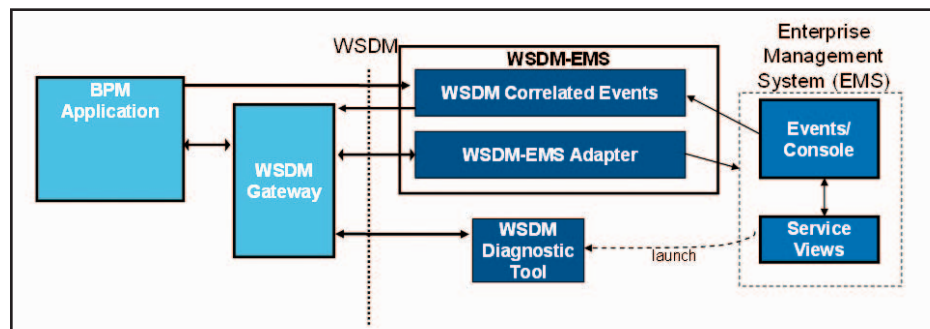


FIGURE 5 WSDM and Business Process Management

In these scenarios, you need to be able to adapt business processes more quickly when there are infrastructure failures, poor performance, etc. WSDM is one technology that can be used to solve this problem. For example, let's say we want to manage a set of business processes that have been developed. In this scenario, the following questions may need to be answered:

- How can we automatically discover changes in the business processes and map them to infrastructure elements?
- How do we allow configuration of this environment from an EMS?
- How do we provide impact analysis information back to the BPM provider (e.g., TIBCO, webMethods, etc.) in a standards-based way?

Given these requirements, let's think about

and other operational events.

With this architecture, problems such as SLA violations could be handled in one of two ways:

1. The management system itself could expose a WSDM interface so the BPM tool could subscribe to certain events. The BPM tool could thus get dynamic feeds from the management system indicating not only that a problem occurred, but also how that problem impacted its business processes. The BPM tool could then make real-time adjustments, such as reconfiguring the business process.
2. With the WSDM interface exposed by the BPM tool, an operator could perform fine-grained configuration on these management resources directly using a diagnostic tool that understands how to query for and configure WSDM resources.

The end result is a flexible, configurable architecture that allows an EMS to interact with a WSDM-based resource as well as provide capabilities for fine-grained configuration and control, event notification and correlation, and dynamic views of the business services and their relationship to IT infrastructure.

## Case Study #2: Managing Service-Oriented Architectures

In the second case study, let's turn our attention to the management of service-oriented architectures (SOAs). An SOA, which consists of loosely coupled services that are discovered, invoked, and composed, is one approach to address the many challenges of changing markets, new customer demand, and emerging technologies. However, organizations wishing to manage an SOA deployment must still address fundamental questions such as:

- *How do I manage the IT infrastructure?*  
Today, many SOAs are built with Web services technologies and it's important to carefully consider the management requirements. For example, you may want to set up SLOs monitoring Web service performance. Or, you may want to provide detailed auditing and historical reports for Sarbanes-Oxley compliance. Securing Web services traffic using XML-DSIG, XML Encryption, and WS-Security may also have to be addressed.
- *How do I leverage an SOA to improve the linkage between business and IT?* Creating the proper linkage between business and IT requires a dynamic model that defines the relationships between business services, their supported software assets, and the virtualized infrastructure. This "SOA services model" could then be leveraged to support life-cycle and automation tasks. For example, deployment of a new service into the infrastructure could be automatically synchronized with a model that represents the available business services. This linkage would serve as a way to handle problem isolation and business impact analysis.
- *How do I integrate an SOA with my existing environment?* An SOA services model offers both a mechanism for virtualization of IT resources and a way to integrate with management consoles, UDDI registries, and the resources that need be managed. WSDM would be seen as a key enabler to this inte-

gration. An open, standards-based integration to the SOA model would make it easier to use management models, and to change the management system over time.

Let's illustrate the value of WSDM here with a simple example. We will assume that we want to monitor the overall application performance for a set of Web services that have been deployed in a Web services container. Monitoring the performance of these services could be accomplished in one of two ways:

1. You could deploy software on the container that monitors these services and reports them to a central management server.
2. You could have an intermediary, such as a broker, that intercepts Web services traffic and keeps track of response times (see Figure 6).

In either case, we can use WSDM to expose the manageability interfaces. WSDM would essentially shield the management system from having to learn about the underlying management technologies or protocols. We could then design an SOA model that uses these WSDM interfaces, thus making it easier to incorporate new types of services into the management ecosystem.

## Let's Get Started

Hopefully, these examples illustrated the value that WSDM can bring to IT management. In the end, use of WSDM will help drive down the costs and complexity in building manageable business services so customers can dynamically manage their heterogeneous IT environments. The good news for WSDM is that a number of vendors, including HP, IBM, and CA that participated in the definition of the WSDM standards are now implementing them in their products. For example:

- HP has released HP OpenView SOA Manager

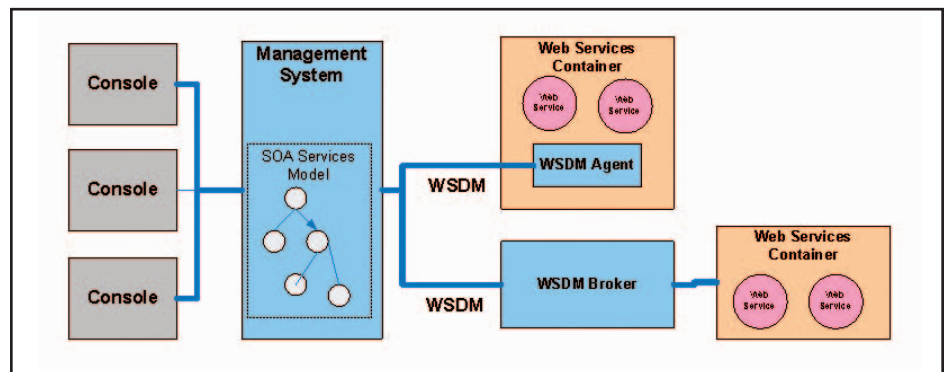


FIGURE 6 SOA management with WSDM technologies

and various HP OpenView Smart Plug-ins based on WSDM technologies

- IBM is working on an Emerging Technology Toolkit that uses WSDM
- CA's Web Services Distributed Management product incorporates aspects of this technology

Your role as an application architect might be in the development of manageable resources that expose WSDM interfaces. To that end, HP has donated a set of open source Java toolkits through Apache. These toolkits support WSDM, WS-RE, and WS-Notification (see Table 2). These efforts were previously under the Apache incubation program, and were recently promoted to full Apache projects.

To learn more about WSDM, you can download and evaluate the Apache WSDM toolkits from <http://ws.apache.org/>. Additionally, if you would like to learn more about management technology, you can take advantage of a number of whitepapers, articles and tips from HP Dev Resource Central (<http://devresource.hp.com>). Included on this site is a whitepaper recently released from HP, IBM, and CA describing a proposed architecture and roadmap for management using Web services technologies (see References section below).

## References

- Cornpropt, J., D. Eckstein, et. al. *Proposal for a CIM mapping to WSDM*. HP Dev Resource Central: [http://devresource.hp.com/drc/specifications/wsdm\\_cim\\_mapping/index.jsp](http://devresource.hp.com/drc/specifications/wsdm_cim_mapping/index.jsp)
- Graham, S., H. Kreger, B. Murray, et. al. *Management using web services: A proposed architecture and roadmap*. HP Dev Resource Central: <http://devresource.hp.com/drc/resources/muwsarch> ©

Listing 1: WSDM management event example

```
<muws-pl-xs:ManagementEvent ...
  <muws-pl-xs:SourceComponent ...
    <muws-pl-xs:ResourceId>urn:machine</...>
  </muws-pl-xs:SourceComponent>
  <muws-p2-xs:Situation>
    <muws-p2-xs:SituationCategory>StartSituation</...>
    <muws-p2-xs:SuccessDisposition>Successful</...>
    <muws-p2-xs:SituationTime>2004-11-11T18:06:00Z</...>
    <muws-p2-xs:Message xml:lang="en">
      Machine: Restart Processing Begun
    </muws-p2-xs:Message>
  </muws-p2-xs:Situation>
</muws-pl-xs:ManagementEvent>
```

## About the Author

Chris Peltz is a software alliances architect for Hewlett-Packard. In this role, Chris helps to drive the adoption of emerging technologies, standards, and architectures with partners looking to integrate with HP management software. Chris has written a number of technical articles on application management solutions and has done extensive research in the areas of Web services and service-oriented architectures.

■ ■ ■ [chris.peltz@hp.com](mailto:chris.peltz@hp.com)

Apache Project	Standard Implementation	Description
WSRF	WS-ResourceFramework	Generic framework for modeling and accessing stateful resources
Pubsub	WS-Notification	Supports the creation and subscription of management events
Muse	WSDM MUWS	Robust Java implementation of management using Web Services

TABLE 2 Apache WSDM toolkits



# Managing Enterprise Data Complexity Using Web Services: Part 2

Developing enterprise digital dashboards using data services architecture

■ Enterprises are increasingly feeling the need for shorter lead time for decision making, the need to extract and present KPI (Key Performance Indicators) to management, and the need for enhanced response capability. These business needs are not in sync with the technological challenges such as the presence of heterogeneous technologies and disparate enterprise systems (e.g., ERP, SCM, CRM, etc.). This situation gets complicated with the increasing number of mergers and acquisitions resulting in the various business units within an enterprise having their own data warehouses. Adding to this is the increasing number of users inside and outside the enterprise who need real-time access to information.

In such a scenario, an EDD (Enterprise Digital Dashboard) would improve the lead time and quality of decision making by extracting and generating KPIs from enterprise software systems. The EDD is in many ways similar to an automotive dashboard, which provides for the driver a single view of the state of the automobile. Development of an EDD involves extracting



WRITTEN BY  
**DR. JAI  
GANESH**



WRITTEN BY  
**DR. SRIRAM  
ANAND**

and generating metrics, indicators, sales figures, production of certain product lines, identifying lead times, inventory levels, and other key organizational data from the enterprise software systems. Dashboards present critical data about the enterprise to the most significant powers who make very strategic decisions based on this information. Therefore, the quality and timeliness of this data cannot be

overemphasized. The most fundamental issue is one of data integration and data rationalization across a variety of applications, databases, and technologies. In this article we present an overview of the architecture for developing enterprise digital dashboards.

## Challenges

EDDs are difficult to implement due to the complexities involved in combining and calculating data from disparate business and enterprise systems such as SAP, i2, etc. Moreover there are associated problems related to the duplication of data that require data synchronization. Multiple systems in different lines of business access data in different ways depending on their needs and specific technologies. There are no consistent practices or techniques in place for the access and update of data. Apart from this, the same data element may be stored and accessed in multiple formats in different databases. The critical factor in the development of an enterprise dashboard is the fact that a dashboard is fundamentally a representation of core business data in multiple formats. These data elements may be rolled up to different levels of granularity depending on the audience in question. Therefore, one of the fundamental issues in the development of a dashboard boils down to the ease of obtaining quality data

from a variety of sources. The key architectural requirements in the development of an enterprise-level dashboard include:

- Guaranteeing data quality given the nature of the audience
- The ability to integrate data sources in a loosely coupled manner
- The ability to provide a unified view of information persisted across varied data sources
- Buying instead of building – use proven commercial/open source products/frameworks instead of building from scratch

### Architectural Options for the EDD: Web Services

Integration solutions from EAI vendors involve large initial investments. Moreover, the EAI products are not very flexible, do not fully support incremental investments, and are of proprietary nature. Furthermore it is not easy to work with EAI systems while integrating with IT systems of different partners with heterogeneous systems. All of the aforementioned problems need to be addressed while implementing a Web services-based EDD solution. This is because the solution should be able to interact with various systems, should be flexible, should support incremental investments, and should be able to interact with the systems of the firm's partners. Web services address most of the problems mentioned above. Web services would support the data coming from various disparate systems to have a single view of the data. Enterprise dashboards need to be able to communicate with multiple business applications in order to display the correct information and apply correct rules in order to display certain data. This calls for a solution that offers a low cost, open standards-based option for executive decision making. Web services are an effective technology to power EDDs. Since Web services are loosely coupled and interoperable, it is easy to expose and extract appropriate data from enterprise systems. The EDD would consume these Web services and processes the data to derive the appropriate metrics, which are then displayed. This could be similar to a portal screen with a high degree of personalization depending upon the type of metrics chosen by the decision maker. This results in shorter lead-time for decision making by mak-

ing available timely information across the supply chain, by accessing data from multiple sources, and bringing in a high degree of interactivity with the information using drill-down features. Let's examine the applicability of Web services as a solution in the context of a specific industry vertical.

### EDD in the Context of the Automotive Industry

We will discuss a Web services-based EDD architecture in the context of the automotive industry. The automotive industry is characterized by global economic slowdown, global overcapacity, decreasing prices and margins, consolidation, etc. Large automotive companies operate across geographies, a phenomena that can be partly attributed to consolidation among car companies, suppliers etc., which has resulted in fewer, larger companies that have more complete product lines. Moreover, consolidation in the automotive industry has resulted in the automotive companies having to deal with disparate systems spread across geographies and supporting differing business models. The automotive companies have large, complex information technology investments. The disparate IT systems in place within the automotive company bring in problems related to information access and decision making. Seamless information sharing across the global operations of the automotive company is necessary to leverage the benefits of consolidation such as cost reduction and effective information access. Appropriate alignment between geographically dispersed business units and functional groups is required to create a unified view of sales, dealers, consumers, products, and services. Currently, each business unit, functional group, and brand operates through independent systems, programs etc. As a result, there is limited synergy across the organization, leading to inefficiencies and lack of coordination. In the following section we illustrate a typical decision-making scenario in an automotive company and explain how the problem can be addressed through Web services-based EDD.

We take the case of a multibillion-dollar automotive company that has business units spread across geographies. The automotive company is structured across geographies as

strategic business units (SBUs). SBUs have a mix of both legacy as well as modern systems (e.g., ERP, SCM, and CRM). The systems of the automotive company across SBUs are as follows:

1. North America
  - Legacy systems (Mainframes)
2. Europe
  - Legacy + SCM
3. Asia
  - ERP + SCM + CRM
4. South America
  - Java based + Oracle RDBMS

The automotive company wants to offer its CEO a top-level view of the performance of its SBUs. The KPIs required are:

1. Revenues by SBU
  - Drill down (break up by product categories e.g., trucks, cars)
  - Drill down (break up by brands)
2. Revenues – target vs. actual, by SBU
  - Drill down (break up by product categories)
  - Drill down (break up by cars)
3. Revenue forecast by SBU
  - Drill down (break-up by product categories)
  - Drill down (break-up by cars)
4. Sales volumes by SBU
  - Drill down (break up by product categories)
  - Drill down (break up by cars)
5. Production volumes by SBU
  - Drill down (break up by manufacturing plants)
  - Drill down (break up by cars)
6. Top dealers by SBU
  - Drill down (break up by product categories)
  - Drill down (break up by cars)
7. Profit margins by SBU
  - Drill down (break up by product categories)

### Current Enterprise Architecture

The current high-level architecture of the systems is illustrated in Figure 1. This diagram highlights the scatter of critical data across geographies. Given the nature of the problem requirements and the diversity of the data sources involved, it seems that the manage-



ment of the data sources and the centralization of common data across lines of business and geographies would prove to be very important in fulfilling the business requirements. Some of the specific issues and challenges in this regard are listed below.

- **Heterogeneous applications:** Core data is required by the dashboard as well as by other enterprise applications. In order to be consistent, the data must be integrated and disseminated using open standards. All consumers of data must obtain data from the

multiple systems and databases will result in business applications having to embed the logic associated with the retrieval/update of multiple data elements residing in different databases.

- **Financial impact:** Depending on the manner in which specific technologies have evolved in a given LOB (Line of Business), the data access techniques may vary between applications. This can lead to higher costs incurred for integration across lines of business due to the difference in technical

line. Latencies in the update of the data that has been sent over the phone may cause incorrect data to be displayed.

- **Resource contention:** This issue can arise due to heavy loads on any of the databases arising during a bulk update situation.
- **Latencies in data retrieval:** The spread of critical data across multiple databases and the associated redundancy may cause additional latencies for certain applications. This can happen due to the bulk update scenario illustrated above.

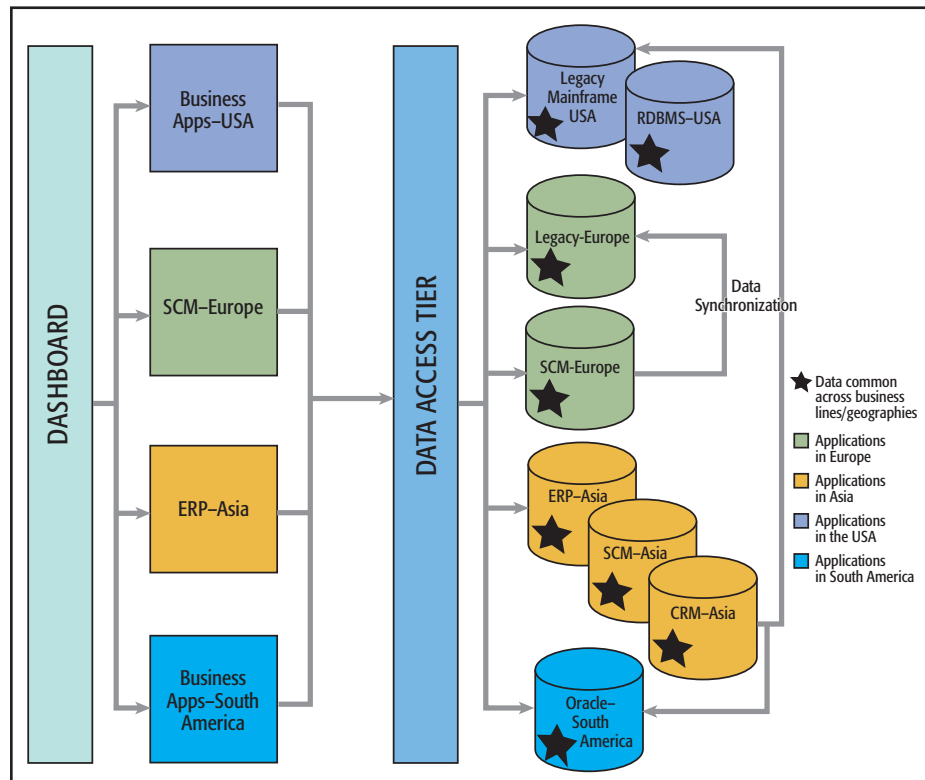


FIGURE 1 Current state of systems and databases

same source. Apart from this, the dashboard would require leveraging logic from several applications that are written using different technologies. These disparate applications must be integrated to present consistent information to the dashboard.

- **Incomplete view of data:** The scatter of critical data, e.g., customer data across multiple databases in various businesses and geographies creates a serious issue with respect to the identification of key customer attributes.
- **Complex systems logic:** The spread of mul-

abilities between lines of business.

- **Problems due to multiple channels of update:** As discussed above, the core business data in a specific database may be updated by a number of mechanisms such as direct update through the business applications dedicated for that line of business, indirect update through synchronization, and update through other channels.
- **User experience issues:** Such a situation may arise when a user requests a change in personal information over the phone or IVR and attempts to look up the same data on-

The proposed architecture for the dashboard consists of two main tiers: the data access tier and the enterprise dashboard tier. The enterprise dashboard provides an aggregated view of the enterprise information collected from varied data sources that are geographically dispersed.

## Data Access Tier

The core data-integration functionality is implemented by the data access tier. This data access tier will focus on the various issues and challenges discussed in detail above. While most conventional data integration solutions result in a number of touch points between the business logic and the integration logic, Web services provide a loosely coupled and extensible solution wherein different types of data sources can be integrated with the EDD without requiring many changes to the existing functionalities. The solution to the aforementioned problems is to provide a consolidated view of data across the enterprise by eliminating multiple channels of update of data and providing a single suite of applications to access/update data. The implementation of this suite of applications as services would eliminate any lock-in on protocols and integrate easily in a heterogeneous environment. Specific aspects of the new architecture would need to address these issues as follows:

- Provide a composite view of data tailored to business processes and usage patterns
- Develop shared data services that can retrieve information for a set of related applications
- Design service contracts based on the needs of individual LOBs/client applications
- Provide information on demand (in re-

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 **\$210<sup>00</sup>**  
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 **\$340<sup>00</sup>**  
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 **\$270<sup>00</sup>**  
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**

## ☐ 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
Int'l - Two Years (24) \$216	You Pay: \$176 /	Save: \$40 + FREE 198 CD
Int'l - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

## ☐ JDJ

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) \$120	You Pay: \$89.99 /	Save: \$40
Int'l - Two Years (24) \$216	You Pay: \$176 /	Save: \$40 + FREE 198 CD
Int'l - One Year (12) \$108	You Pay: \$99.99 /	Save: \$8

## ☐ Web Services 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
Int'l - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE 198 CD
Int'l - 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
Int'l - Two Years (24) \$216	You Pay: \$170 /	Save: \$46 + FREE 198 CD
Int'l - 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: \$79.99 /	Save: \$88 + FREE 198 CD
Can/Mex - One Year (12) \$84	You Pay: \$49.99 /	Save: \$34
Int'l - Two Years (24) \$216	You Pay: \$89.99 /	Save: \$126 + FREE 198 CD
Int'l - One Year (12) \$108	You Pay: \$59.99 /	Save: \$48

## ☐ Wireless Business & Technology

U.S. - (12) Two Years (24) Cover: \$143	You Pay: \$49.00 /	Save: \$71 + FREE 198 CD
U.S. - One Year (6) Cover: \$60	You Pay: \$29.99 /	Save: \$30
Can/Mex - Two Years (12) \$120	You Pay: \$69.99 /	Save: \$51 + FREE 198 CD
Can/Mex - One Year (6) \$60	You Pay: \$49.99 /	Save: \$10
Int'l - Two Years (12) \$120	You Pay: \$99.99 /	Save: \$20 + FREE 198 CD
Int'l - One Year (6) \$72	You Pay: \$69.99 /	Save: \$2

## 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.

## ☐ 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: \$79.99 /	Save: \$88 + FREE 198 CD
Can/Mex - One Year (12) \$84	You Pay: \$49.99 /	Save: \$34
Int'l - Two Years (24) \$216	You Pay: \$89.99 /	Save: \$126 + FREE 198 CD
Int'l - One Year (12) \$108	You Pay: \$59.99 /	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: \$80 + FREE 198 CD
Can/Mex - One Year (12) \$120	You Pay: \$99.99 /	Save: \$20
Int'l - Two Years (24) \$264	You Pay: \$189 /	Save: \$75 + FREE 198 CD
Int'l - 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: \$80 + FREE 198 CD
Can/Mex - One Year (12) \$120	You Pay: \$99.99 /	Save: \$20
Int'l - Two Years (24) \$264	You Pay: \$189.00 /	Save: \$75
Int'l - One Year (12) \$132	You Pay: \$129.99 /	Save: \$2

## ☐ PowerBuilder Developer's Journal

U.S. - Two Years (24) Cover: \$360	You Pay: \$169.99 /	Save: \$190 + FREE 198 CD
U.S. - One Year (12) Cover: \$180	You Pay: \$149 /	Save: \$31
Can/Mex - Two Years (24) \$360	You Pay: \$179.99 /	Save: \$180 + FREE 198 CD
Can/Mex - One Year (12) \$180	You Pay: \$169 /	Save: \$11
Int'l - Two Years (24) \$360	You Pay: \$189.99 /	Save: \$170 + FREE 198 CD
Int'l - One Year (12) \$180	You Pay: \$179 /	Save: \$1

## ☐ WLDJ

U.S. - Four Years (24) Cover: \$240	You Pay: \$99.99 /	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: \$99.99 /	Save: \$140 + FREE 198 CD
Can/Mex - Two Year (12) \$120	You Pay: \$69.99 /	Save: \$50
Int'l - Four Years (24) \$240	You Pay: \$120 /	Save: \$120 + FREE 198 CD
Int'l - 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](http://www.sys-con.com/2001/sub.cfm)

**SYS-CON  
MEDIA**

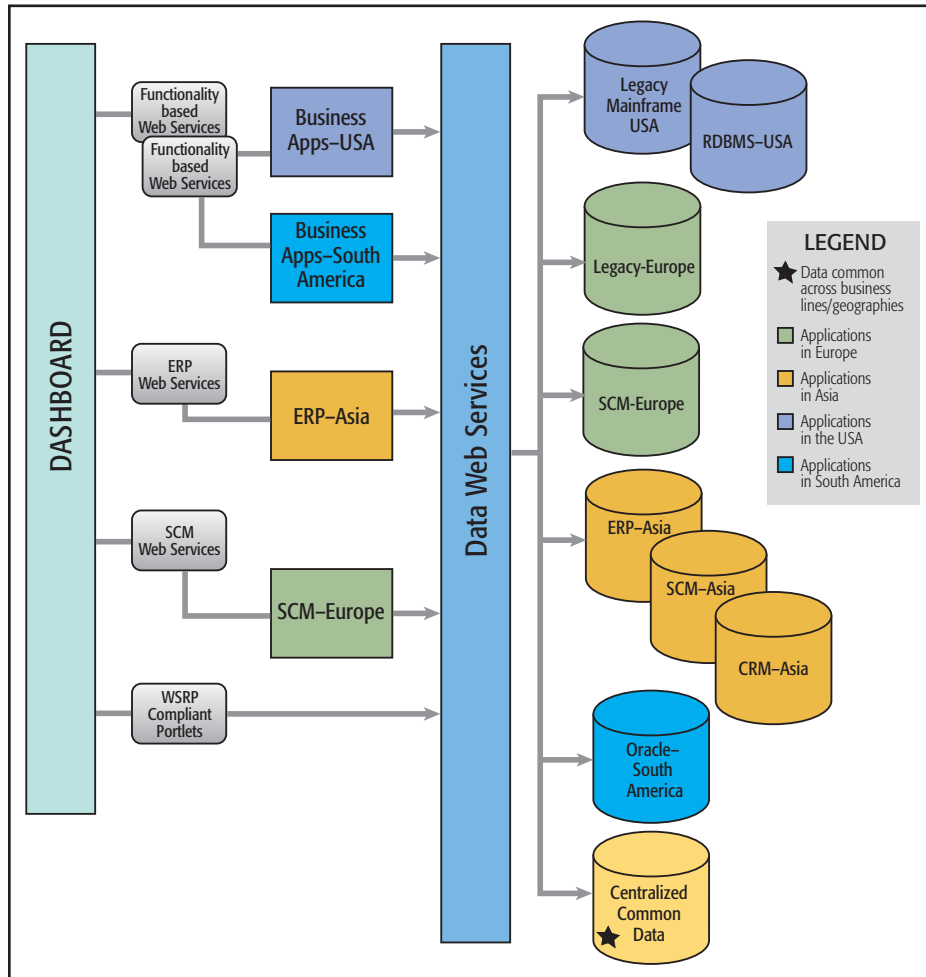


FIGURE 2 Future state EDD architecture based on Web Services

sponse to service requests) by optimizing performance and caching heavily accessed data

- Enforce data security by authenticating invocation clients and protecting against unauthorized data access
- Enforce all updates through the data service layer in order to guarantee data consistency across all systems

### Dashboard Tier

The enterprise dashboard provides the user interface and takes care of other non-functional tasks such as security, internationalization, scalability, availability, and caching. This allows incremental adoption of the Web services strategy, which is a big attraction for organizations that have a huge IT infrastructure to migrate. A Web services-based architecture that leverages the improved data

access mechanism is shown in Figure 2. The specific characteristics of the overall architecture are as follows:

- Web services that are created for the purpose of providing functionality in a loosely coupled, implementation independent manner. These implementations will leverage the core business logic in the existing applications.
- A centralized database that maintains data common to multiple lines of business and geographies.
- A set of services that manage data access/update for all databases; these services will manage all data access/update and will become the de facto data access layer for all applications.
- Depending on the specific line of business and the problems (or lack thereof) associated with the business data, this data will

continue to remain in the existing databases.

- Infrastructure functionality is consolidated and implemented as a Web service to allow additional consumers to leverage the same rules and components. An important benefit is in the area of security, specifically authentication and access control.
- Scalability and availability of various applications can be handled in a streamlined manner by using Web services. Implementing strict service contracts with specific levels of service decouples the service consumers from actual service implementations.

### Benefits

It is important to analyze the actual benefit realized from the migration to Web services that help realize an EDD. This analysis will help in identification of the factors that may be used for the calculation of ROI. The various factors associated with the technical and business viewpoints are:

- **Cost reduction:** In the existing setup, individual lines of business manage various aspects, including infrastructure and common data. With the centralization of common functionality and data and the elimination of multiple redundant data sources and channels of update, a significant portion of this cost may be eliminated. There may be higher upfront costs involved in implementing the Web services, but the long-term costs related to recurring yearly expenses should be lower, at least as far as the common data, processes, and functionality are concerned.
- **Flexible business applications:** Rollout of the Web services with a robust service contract and SLA will allow new applications to be built faster and cheaper. Integration across lines of business will be significantly easier due to the common data being shared and accessed in a standard fashion.
- **Transparency for LOBs:** In the existing model, applications had to be designed to handle issues with respect to heterogeneous systems, data redundancy, and data synchronization. The implementation of Web services will completely insulate LOB applications from redundancies in data storage and issues with data synchronization.
- **Patterns:** As discussed earlier, the dependency on proprietary techniques and frameworks used for component development in



various lines of business may be eliminated by implementing Web services. This will create common patterns for accessing commonly used data and functionality.

## Conclusion

A Web services-based EDD solution has tremendous potential to solve the decision-making problem by enabling information aggregation from multiple disparate systems spread across the enterprise. This approach is a low cost methodology owing to the fact that existing legacy investments are leveraged. The architecture described above can be considered to be a reference architecture for all enterprise-level dashboard applications. This architecture enables a true flexible aggregation of information from the internal systems of the organization using Web services, and offers a single point of contact for decision-making information. Further, this enables existing legacy applications to take part in the overall business architecture. In future work, we will be addressing implementation concerns associated with building shared data services, including performance analysis and migration strategies.

## References

- The role of EII in SOA, Beth Gold-Bernstein, ebizQ white paper, June 2004
- How Do You SOA Enable Your Data Assets? Jim Green, DM Direct Newsletter, October 15, 2004
- 12 Steps to Implementing a Service-Oriented Architecture, David S. Linthicum, White paper – Grand Central Communications, October 2004
- Data Services for Next-Generation SOAs, Christopher Keene, *Web Services Journal*, December 2004
- Architecting Data Assets, Jeff Schulman, Gartner Research, August 2004

## About the Authors

Dr. Jai Ganesh is a research associate in the technology research division of Infosys Technologies Limited. He obtained his PhD in information systems from the Indian Institute of Management Bangalore (IIMB) in 2003 and holds an MBA degree in corporate strategy and marketing. His research focuses on Web services, IT strategy and adaptive enterprises. His research has been published in journals such as *Information and Management*, *Journal of Global Information Management*, *International Journal of Retail and Distribution Management*, and conferences such as AMCIS, ICWS, ICEC, and ICEB. He serves as a reviewer for a number of peer-reviewed journals and conferences, and has consulted for many software firms.

■ ■ ■ [jai\\_ganesh01@infosys.com](mailto:jai_ganesh01@infosys.com)

Dr. Sriram Anand has over 14 years of work experience in industry and research and holds a Bachelor's degree from IIT-Madras and a PhD from SUNY-Buffalo, USA. Sriram is a principal researcher in Software Engineering and Technologies at Infosys Technologies, Bangalore. His career in IT has spanned various roles including component developer, technical lead, senior architect, and development manager. Sriram is experienced in designing enterprise architectural strategy for leading U.S. companies in the financial services, retail, and pharmaceutical domains. He has also published several articles in leading journals and participated in conferences in the engineering and IT domains.

■ ■ ■ [sriram\\_anand@infosys.com](mailto:sriram_anand@infosys.com)

# IN THE NEXT ISSUE OF **WSJ...**

## **FOCUS: Web Services Management**

### **Web Services and Enterprise Content Management: Hype vs. Reality**

New business requirements are leading companies to change the way they deploy Enterprise Content Management (ECM) data and applications. Faced with the limited interoperability and/or scalability of conventional ECM platforms, developers are turning to Web services as a way to realize ECM functionality and real-time content wherever it's needed within an organization. While this approach is still relatively new, and more work remains to be done to improve the effectiveness, it already shows promise as a better way to think about ECM technology.

### **WSRP: Dynamic and Real-Time Integration**

The IT industry has seen various milestones. Some of the major milestones are the introduction of Desktop PCs, Windows platform, C - language, evolution of OOPS and C++, Internet technologies, JAVA, etc. WSRP has all of the necessary characteristics for becoming a member of this elite group. WSRP is an example of real-time integration and it exploits the power of Web services and portlet technologies.

### **Describing Web Services in 2005**

For years, WSDL has been synonymous with describing interoperable Web services. However as the now venerable Web Services Description Language (WSDL) 1.1 heads towards its fifth birthday, it finds itself beset with challenges and opportunities. Currently in its second incarnation and heading for its third, WSDL has been the definitive gold standard for providing descriptions of Web services.

### **10 Things to Think About When Building the Perfect SOA**

Right now the implementation of SOAs seems to involve much more hype than actual work. However, there are some patterns beginning to emerge, or, procedures the implementers are doing right to insure success. These patterns are not always obvious, so perhaps this is a good time to learn through the successes of others and do our own homework before we spend millions on moving to an SOA.

# Designing Services for Performance – Part II

Continued discussion of the secrets of building and operating a realistic SOA

■ As we discussed last month, performance is often an afterthought when building new systems, including SOAs. We're finding that services and SOAs fall victim to this oversight as well. Indeed, there is a right way and a wrong way to design a service and an SOA. Also, there are things that are out of your control that you must consider during your design.

This month let's continue our discussion with some important performance concepts, including how to create a performance model, as well as more service and SOA design tips when considering performance.

## Creation of a Performance Model

SOAs are not unlike any other distributed computing systems, and thus designing a performance model should be nothing too new. At this point we understand exactly how each service behaves under an increasing load, and we have enough data to plug into a model. Now, it's just a matter of building a model.

There are very expensive performance monitoring and simulation tools that are for sale in the market, but sometimes the least expensive and most simple tools work best...in many cases, just a spreadsheet will do. For our purposes, we need to consider both information and behavior in the context of performance, as well as core features of an SOA.

**Information Movement Modeling**, typically asynchronous in nature, means we're attempting to simulate how information



WRITTEN BY  
**DAVID  
LINTHICUM**

moves from point to point, point to many points, or many points to many points. Based on the information we accumulated we know the:

- Information production rate from a service
- Information consumption rate from a service

For example, an instance of a service is able to produce 52 messages (or similar groupings of in-

formation) per second...the source service. An instance of a service is able to consume 34 messages per second...the target service. This is a simple point-to-point relationship, but keep in mind that multi-points to multi-points are always possible, and those are a bit more complex to model since you have to determine patterns of movement between multiple points vs. all messages produced by a single service that are consumed by another.

Moreover, keep in mind transformation and routing latency is typically an issue here as well, and needs to be modeled along with consumption and production. You should have test data from these services, but the performance of transformation and routing services will be largely dependent upon the complexities of the transformations and

logic associated with the routing. What many do when creating performance models is to model very complex, complex, and simple transformation scenarios, and the percentages of each.

**Service Invocation Modeling**, typically more synchronous in nature, means we're attempting to determine the number of times a service is able to provide a behavior (application function) in an instance of time, typically a second.

For instance, you may have a service that provides a risk calculation for the insurance business, and is perhaps abstracted into several different applications (composites). We know through testing that each composite can invoke the service up to 100 times a second before it hits a saturation point, meaning the performance of the service quickly diminishes as additional load is placed upon it. This saturation number plugs into the model, as well as the number of applications that are abstracting this service. You have to model all of these services in the same way.

Models are important because they allow you to predict performance under changing needs without having to actually build and test the system. Models, of course, are not perfect, and you must constantly adjust assumptions and modeling information as you learn more about the behavior of the architecture.

## Designing for Performance, Monitoring, and Optimizing

So, now that we know how to diagnose the performance of an SOA, as well as model for it to determine how it will behave in a changing environment, how do we design a service and SOA with optimized performance? Here are a few tips.

- The more processing you can place at the origin of the service, the better your SOA will perform. In many SOAs, the architects abstract the services to a single server, and performance can be somewhat problematic in larger implementations.
- Many services are built on top of more traditional legacy APIs, and as such the translations between legacy APIs to expose them as services may cause performance problems. The ability to leverage existing legacy systems as services is a powerful notion. However, you must be careful in

selecting the proper enabling technology to do this. Service invocations that take a second or more to produce behavior, or information bound to behavior, will cause big problems when you align them with hundreds of other services that are doing the same thing.

- The use of too many fine-grained services may cause performance problems. Indeed, you should not be afraid to leverage fine-grained services within your SOA. However, you need to understand the performance issues associated with doing so, and take the network bandwidth and how other applications leverage the services into careful consideration.
- Make sure to consider performance when selecting your orchestration layer. Many BPEL engines are notoriously poor performers, and can become the bottleneck for the SOA.
- Understand the basic rule that, while the

value of an SOA is the ability to leverage many remote services, the more services you leverage, the more problematic your SOA will become.

### Core Issues

Making solutions scale is nothing new. However, the SOA technology and approaches recently employed are largely untested with higher application and information and service management traffic loads. SOA implementers were happy to get their solutions up and running, yet, in many cases, scalability is simply not a consideration within the SOA, nor was load testing or other performance fundamentals. We are seeing the results of this neglect now that SOA problem domains are exceeding the capacity of their architectures and the technology. It does not have to be this way.

What is more, many SOA technology vendors have not focused on scalability

within their solutions. Instead, feature/function enhancements are the rule of the day. Architects feel it's more important to add orchestration features and more adapters to their solution than to figure out how to reliably pump more information, and manage more services, with their product. It's time for that focus to change. ©

### About the Author

David Linthicum is the CTO at Grand Central Communications ([www.grandcentral.com](http://www.grandcentral.com)), and a leading expert in the application integration and open standards areas. He has held key technology management roles with a number of organizations, including CTO of both Mercator and SAGA software. David has authored or coauthored 10 books, including the groundbreaking and best-selling *Enterprise Application Integration* released in 1998. His latest book is *Next Generation Application Integration, From Simple Information to Web Services*. Dave does a Podcast, "SOA Expert." The feed is at: [http://soaexpert.podblaze.com/skin\\_soaexpert.xml](http://soaexpert.podblaze.com/skin_soaexpert.xml).

■ ■ ■ [dlinthicum@grandcentral.com](mailto:dlinthicum@grandcentral.com)

ONCE YOU'RE IN IT...

ONCE YOU'RE IN IT...



- WebServices
- Storage & Security
- IT Solutions Guide
- PowerBuilder
- ColdFusion
- WebSphere
- LinuxWorld
- Wireless
- WLDJ
- NET
- JDJ
- MX

...REPRINT IT



...REPRINT IT

Contact Dorothy Gil  
201.802.3024  
[dorothy@sys-con.com](mailto:dorothy@sys-con.com)

**SYS-CON**  
MEDIA

**RePrints**



# Altova Enterprise Suite 2005

Effective, powerful, and useful



■ XML development is a term that can mean many things to many different people. For some, it is the modeling and creation of XML Schemas and instance documents used to exchange data. Others see it as a part of the larger effort of developing Web services. Still others view it as a means to transform and integrate systems whose data structures and definitions are proprietary. All of these types of development require various skills and tools.

Altova's suite of XML tools provides functionality for essentially all forms of XML development. The suite includes core Schema and instance document development support, Web services development support, data transformation and integration support, and finally, presentation support. The products available in the suite include:

- XMLSpy
- MapForce
- StyleVision
- SchemaAgent
- Authentic
- DiffDog

This review will discuss XMLSpy, MapForce, and StyleVision.

## XMLSpy

XMLSpy is the core of the enterprise suite from Altova. It provides all of the tools necessary to design, develop, edit, and transform XML documents and schemas. It also supports XML database technologies, Web services, and provides code generation capabilities for the Java, C#, and C++ programming languages.



WRITTEN BY  
**BRIAN  
BARBASH**

The first things developers will notice in this iteration of XMLSpy are the simple but effective updates to the UI. For example, the text mode of the application is now collapsible by element in order to more easily manage large documents. Another nice change is the addition of an auto-hide capability to the Project, Info, Elements, Attributes, and Entities panels, providing a more ef-

ficient use of screen real estate.

## Modeling and Editing XML and XML Schema

XML Schema modeling in XMLSpy is mainly done in a graphical editor as shown in Figure 1. In this view, schema elements are represented as nodes of a tree structure. In the example shown, the PropertiesConfig-

Type is a global Complex Type structure that contains a sequence of repeating Property nodes. The graphical editor also supports visualization of advanced schema objects such as keys (shown), unique constraints, and key refs.

The Details panel in Figure 1 provides access to the lower-level properties of the selected node such as type, cardinality, and reference information, among other characteristics.

Once the XML Schema is modeled, the document may be validated against the latest W3C XML Schema specification simply by clicking the validate button on the toolbar. Documents are also validated when saved. If any errors exist, they are presented to the developer to correct.

XML instance documents are modified in the Text and Grid views. The Text view is the standard text editor within XMLSpy, providing text folding capabilities, autocomplete, and syntax highlighting.

The Grid view is useful for working with documents that contain repeating elements, such as the Properties node in the example above. In this view, XML is rendered in a table format that allows developers to view documents in rows and columns.

Instance documents are similar to XML Schemas and may be validated if a schema is defined in the schemaLocation attribute of the root element, or if the DTD is defined in the DOCTYPE node. Developers may then validate the document either by clicking the validate

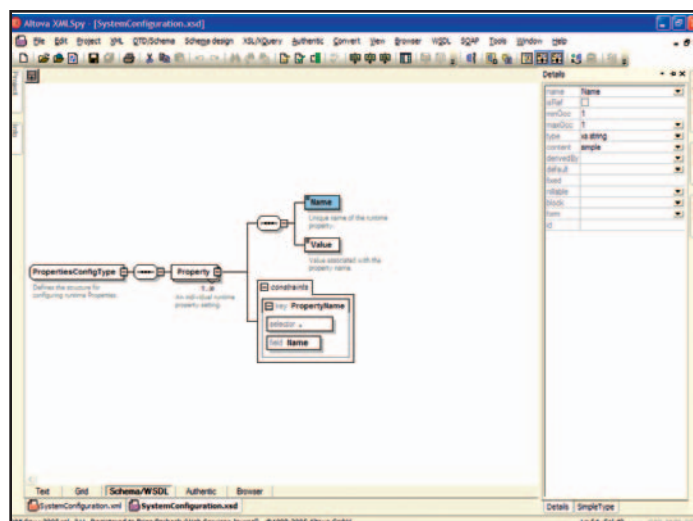


FIGURE 1 XML Schema graphical editor

# Subscribe Today!

— INCLUDES —  
**FREE**  
DIGITAL EDITION!  
(WITH PAID 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*

## Editorial Mission

Greater Collaboration and Efficiency Through Education

- ✓ **ISSJ's** editorial mission is to showcase proven solutions that will guide, motivate, and inspire senior IT and business management leaders in the planning, development, deployment, and management of successful enterprise-wide security and storage solutions.
- ✓ **ISSJ** brings together all key elements of data storage and protection, and presents compelling insight into the benefits, efficiencies, and effectiveness gained by focusing on these two critical areas of IT simultaneously.
- ✓ **ISSJ** is an objective, critical source of information that helps storage and security managers make informed management decisions about what is working today, and what they need to plan for tomorrow, and is the only publication that focuses exclusively on the needs of IT professionals who are driving the enterprise storage architecture/infrastructure while pursuing and incorporating the latest security technologies.
- ✓ **ISSJ** achieves our mission by delivering in-depth features, practical "how-to" solutions, authoritative commentary, hard-hitting product reviews, comprehensive real-world case studies, and successful business models, written by and for professional IT storage and security practitioners.

# SAVE 50% OFF!

(REGULAR NEWSSTAND PRICE)

## Only \$39<sup>99</sup>

ONE YEAR  
12 ISSUES

[www.ISSJournal.com](http://www.ISSJournal.com)

or

1-888-303-5282

**SYS-CON  
MEDIA**

The World's Leading IT Technology Publisher



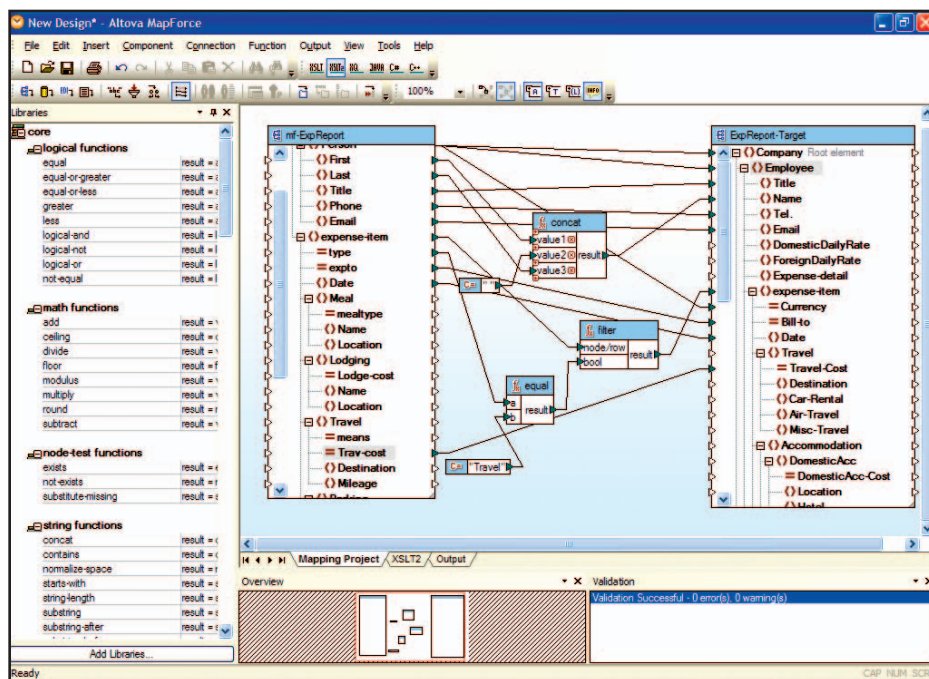


FIGURE 2 MapForce graphical editor

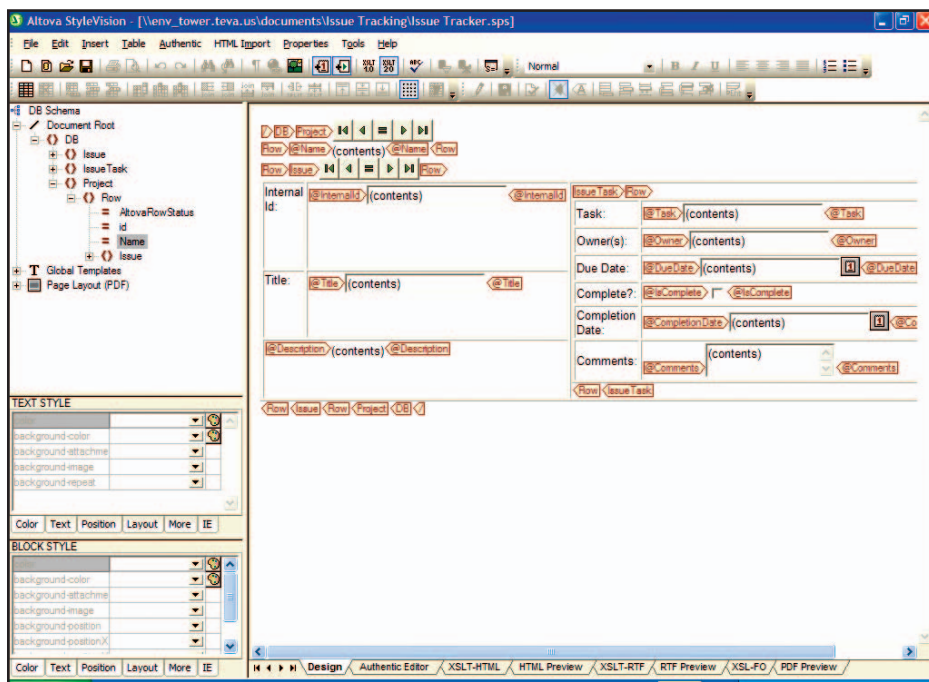


FIGURE 3 StyleVision editor

button in the toolbar or saving the instance document. Any errors are presented on screen.

### XMLSpy and Web Services

XMLSpy is also an effective tool for the Web services developer. It provides full support for the latest WSDL and SOAP specifications,

allowing it to both model and invoke Web services.

Modeling Web services, as with schemas, is mainly done in a graphical editor. The editor is logically broken into the main components of a WSDL document: Operations, PortTypes, Bindings, and Services. Each component

allows a developer to graphically define and update the relevant properties. For example, adding input, output, and fault messages to an operation is as simple as selecting from a context menu and defining the message parts.

XMLSpy may also serve as both a SOAP client and debugger. To act as a client, the WSDL of the service to be called is simply identified and a SOAP document is generated. Once the parameters of the call are set, the document may be sent to the service and the results are displayed. This capability is extremely useful for testing and debugging Web service applications.

When acting as a SOAP debugger, XMLSpy becomes a proxy between the Web service client and the Web service itself. It provides the capability of establishing breakpoints within a SOAP document based on function calls or XPath query statements. Intercepted documents may be inspected and modified before they are forwarded to the service. Service results are then passed back to XMLSpy for inspection and/or modification before reaching the client.

## MapForce

Altova's MapForce product is a data mapping and integration tool that provides transformation capabilities for a number of different formats, including XML, flat file, database, and EDI. Like its XMLSpy counterpart, it supports the latest working drafts of XSLT 2.0 and XPath 2.0. The tool may be used to create one-time data conversion mappings, or it can output Java, C#, and C++ code for integration into standard data exchange interfaces.

### Building Transforms for XML

The MapForce graphical editor is a simple yet extremely powerful environment in which to transform documents. Shown in Figure 2 with the mapping exercise from the tutorial, it uses a standard drag-and-drop interface to build the transformation logic. When working with XML, source and target schemas are identified and represented as side-by-side tree structures. Source nodes may be dragged to the relevant target nodes. If working with higher-level-containing structures, MapForce will automatically map child nodes based on a set of configurable criteria. If the transformation method is XSLT, XSLT2, or XQuery, a third tab in the



editor is present that contains the relevant source code. As development progresses, MapForce updates the Output tab with the results of the transformation.

As mentioned previously, MapForce supports the latest specifications of XSLT and XQuery, as well as providing the ability to generate code in Java, C#, or C++. At any time during development, the method of transformation may be changed to any of the other supported methods with the simple click of a button. With each method, the output and relevant source code tabs (if visible) are updated in real time.

#### Working with Databases

Out of the box, MapForce supports Microsoft Access, Microsoft SQL Server, Oracle, MySQL, Sybase, and IBM DB2 natively, along with any other ADO- and ODBC-compatible database. Generally speaking, databases are represented as simple XML documents, with each table acting as a container node for its columns. Mapping rules are defined in the same way as XML transformations with the exception that only Java, C#, and C++ transformation methods are supported.

#### Other Data Sources

This new version of MapForce provides support for EDI documents and flat files (delimited and fixed width). EDI support includes the UN/EDIFACT directory 0.4B and version 5012 of the ANSI X12 specification.

The flat file definition process is similar to that of Microsoft Excel import utility. Developers select the type of file – fixed or delimited, then proceed by defining the data types and other details of the data to be imported. An option to use the first row of the file as the field names is also available. When completed and imported, the flat file appears as any other transformation source or target in the MapForce mapping window.

#### StyleVision

StyleVision, as its name implies, is a graphical stylesheet editor. The tool supports XSLT and XSL:FO transformations to HTML, PDF, RTF/Microsoft Word, and Altova's Authentic formats. Like its suite companions, it also provides standard support for XSLT 2.0 and XPath 2.0. Similar to MapForce, the

output of this tool is XSLT. However the primary purpose of the XSLT in this case is for presentation and user interaction, instead of data exchange and integration.

#### Working with Stylesheets

As with MapForce, stylesheets are built directly from the schema or DTD that defines the structure of a document, or directly from database tables. Figure 3 illustrates a simple database stylesheet for a custom issue-tracking system.

The left side of the screen represents the schema of the database that supports the issue-tracking system. The right side of the screen is the graphical structure of the output. Along the bottom are the various output options for rendering the stylesheet.

As seen in the figure, content may be arranged and formatted in any layout supported by HTML. For this example, content will be created for the Authentic Editor – Altova's graphical environment for editing XML content. Since this stylesheet is backed by a database, any element may be represented as an editable field. In the Authentic Editor, changes to the field are then stored back into the database (note: Authentic is the only output option that supports database updates). This functionality provides a fast and easy mechanism to create and deploy simple database applications.

#### Summary

As XML development grows in complexity and capability, effective tools will always be required to produce high-quality applications. Altova's suite of products is not only effective, but extremely powerful and useful as well. It addresses XML development, data transformation and integration, Web services, and presentation, among a number of other functions. XMLSpy, along with its siblings in the suite, shines, and they are worthy of any XML development effort. ☺

#### About the Author

Brian R. Barbash is the product review editor for Web Services Journal. He is a senior consultant and technical architect for the Envision Consulting Group, a management consulting company focusing on contracting, pricing, and account management in the pharmaceutical industry.

■ ■ ■ bbarbash@sys-con.com

THREE REASONS TO

**blog-n-play.com**

**1 Get instantly published to 2 million+ readers per month!**

*blog-n-play™* is the only **FREE** custom blog address you can own that comes with instant access to the entire i-technology community. Have your blog read alongside the world's leading authorities, makers and shakers of the industry, including well-known and highly respected i-technology writers and editors.

**2 Own a most prestigious blog address!**

*blog-n-play™* gives you the most prestigious blog address. There is no other blog community in the world that offers such a targeted address, and comes with an instant targeted readership.

**3 Best blog engine in the world...**

*blog-n-play™* is powered by *Blog-City™*, the most feature rich and bleeding-edge blog engine in the world, designed by Alan Williamson, the legendary editor of *JDJ*. Alan kept the i-technology community bloggers' demanding needs in mind and integrated your blog page to your favorite magazine's Web site.

 **www.TAMI.linuxworld.com**  
"Many blogs to choose from"

**PICK YOUR MOST PRESTIGIOUS ADDRESS**

IT Solutions Guide	MX Dev. Journal
Storage+Security Journal	ColdFusion Dev. Journal
JDJ: Java	XML-Journal
Web Services Journal	Wireless Business & Tech.
.NET Dev. Journal	WebSphere Journal
LinuxWorld Magazine	WLDJ: WebLogic
LinuxBusinessWeek	PowerBuilder Dev. Journal
Eclipse Dev. Journal	

**3 MINUTE SETUP**

**Sign up for your FREE blog Today!**

**blog-n-play.com™**  
i-Technology Blogs Read by Millions *beta*

— This site will go beta February 15, 2005!

# XML Binding Frameworks in the Context of Service-Oriented Architecture

Make an informed choice about a binding framework for your SOA needs

■ This article critically evaluates the role of XML binding frameworks play in the context of service-oriented architecture (SOA) platforms, and it also provides an objective evaluation of the popular XML binding frameworks in a J2EE environment.

**X**ML binding refers to the mapping of XML documents to/from any suitable internal representation (e.g., object-based representation) that is understandable by the underlying system, and in the process facilitating easy and intuitive access to the data in XML documents. In a J2EE context, this translates to an easier and logically meaningful way of accessing the data in XML documents, rather than using the low-level DOM/SAX parsers.

To illustrate, in Listing 1, an order-processing application would find it easy to access *Order*, *Item*, and *Customer* objects rather than using the XML-specific data elements listing

WRITTEN BY  
**BIJOY MAJUMDAR,  
DR. SRINIVAS PADMANABHUNI,  
UJVAL MYSORE,  
& VIKRAM SITARAM**

each element, its text, and its attributes as in a DOM/SAX approach. Due to this abstraction, the applications can directly deal with the business entities as part of the business functionality.

XML binding also abstracts away many low-level document details, and it requires less memory than a document model-based approach (such as DOM or JDOM).

Listing 1: An XML listing of data

```
<Order ordNo="0123">
  <Customer custNo="C123">
    <CustName>ABC</CustName>
    <Street>123 Main St. </Street>
    <City>Hyd</City>
    <State>AP</State>
    <PostCode>50003</PostCode>
  </Customer>
  <Items>
    <Item ItemNo="1">
      <Model>M12</Model>
      <Qty>10</Qty>
    </Item>
    <Item ItemNo="2">
      <Model>M10</Model>
      <Qty>5</Qty>
    </Item>
  </Items>
</Order>
```

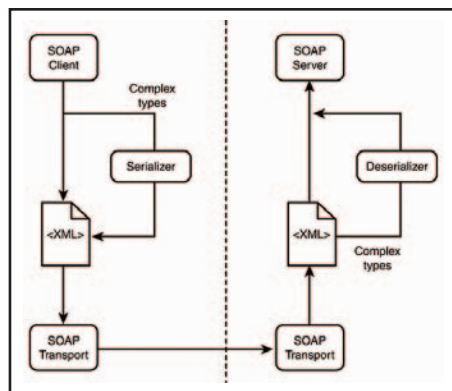


FIGURE 1 XML serialization and deserialization in SOAP

## The Role of XML Binding in SOA

In the context of SOA, XML is of vital importance and has become the lingua franca of communication. We will first outline the importance of XML binding in SOA platforms. Some of the key advantages of using XML binding in the context of an SOA are described below.

### *Serialization/deserialization of SOAP payloads in Web services platforms with increased interoperability*

Web services represent the most popular SOA formalism, and are based on strict XML-based standards for communication (XML Schema for data, and SOAP for wire-level messaging). Web services rely on SOAP marshalling and demarshalling capabilities on the service provider and consumer side (See Figure 1). The marshalling and demarshalling is required to and from the XML document passed over the wire to the native format. Typically the XML payload of a SOAP message needs to be serialized at the requestor end, and deserialized at the provider end (or vice versa). Thus, XML binding frameworks are crucial to handling the serialization and deserialization requirements in SOAP invocations.

Typically SOAP servers have specific custom serialization/deserialization mechanisms for XML data in SOAP payloads. In J2EE, JAXB is a core Java-based standard for XML binding. A JAXB-compliant application is able to utilize different JAXB implementations by just regenerating the schema classes without any change to the application code.

Without relying on an underlying SOAP implementation where there is variance in serialization/de-serialization mechanisms, a J2EE Web Service application can use a JAXB based XML binding framework for the payload marshalling and de-marshalling. This decoupling of XML binding from a SOAP provider increases reuse and interoperability. Further, the same XML Binding framework can be used for Web Service requests alongside applications like configuration reader etc.

*Working with business entities is eased, and reuse is made possible*

The binding frameworks build an ab-

straction layer, thereby hiding the long parsing algorithms and complexity of the DOM/SAX API. Developers need work only with business logic and not with the finer details of data binding and transformation. The business entities represented by mapped object versions of XML documents/parts of documents can be reused across multiple services. On a related front, there is ample scope for improvement of the performance of SOA-based applications by caching at the mapped object level.

#### **Configuration and customization becomes flexible and parameterized**

SOA-based applications can leverage XML to provide dynamic configuration, deployment, and customization. Since configuration/deployment files leverage XML format, the use of XML binding frameworks significantly eases dynamic configurability and customization. In the long term, it is envisaged that metadata based on XML will be the foundation of service mediation technologies to enable typical SOA needs like service versioning, service personalization, and dynamic service provisioning. In view of this, XML binding frameworks will be key constituents of SOA life-cycle management frameworks.

#### **Semantic Interoperability among stakeholders in SOA**

Semantic Interoperability reflects the need for different stakeholders in SOA (service requestors and service consumers) to share a common understanding of the

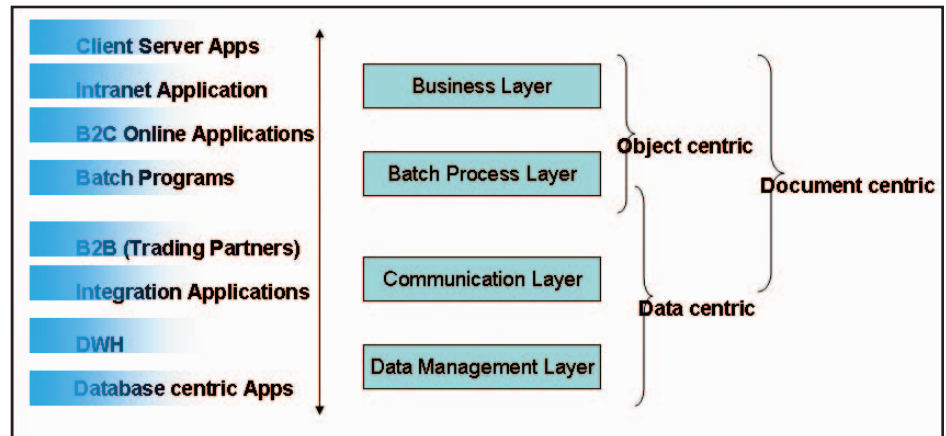


FIGURE 2 Potential usages of XML binding frameworks

documents/data items interchanged. In the recent past, a lot of business communities have come together to define vertical-specific semantic vocabularies in XML abstracting for all of the typical objects and documents of that vertical, e.g., XBRL is an XML standard for financial reporting. To enable service requestors and service providers to process these XML standards-based documents, XML binding frameworks are key to providing a standardized set of IT functionalities in any vertical, especially in those domains where such standards are mainstream.

#### **XML-based data sharing and persistence**

In certain situations, data in the form of XML needs to be shared among multiple participants in an SOA. For example, the data from an XML database may need to be accessed by multiple services. XML binding frameworks will enable an easier way for

the services, especially those hosted on a similar platform, to access the data. In any situation in which an XML-based Common Information Model of an enterprise needs to be enforced, XML binding frameworks can significantly ease the deployment by providing an easier way to use the API.

#### **Shared Data Services**

Shared Data Services (SDS) enable SOA at the data layer by providing a layer of abstraction on top of various data repositories that allow client applications to access these data repositories. Since it is a shared service, it allows reusability and interoperability.

One of the key data formats for working with SDS is XML Schema. The underlying database serving an SDS platform is either a native XML database or a relational database. An XML binding can be used to push data into/from a native XML database or to pull/push data from a relational database accompanied with mapping to/from an XML schema

#### **A Characterization of XML Binding Frameworks**

Having established the vitality of XML binding frameworks in an SOA context, we'll first dissect the typical XML binding frameworks.

Many XML binding frameworks are focused on code generation from XML Schema. Using this approach, the starting point is a schema grammar for the documents to be processed, and then you use

“ XML binding also abstracts away many low-level document details, and it requires less memory than a document model-based approach (such as DOM or JDOM) ”



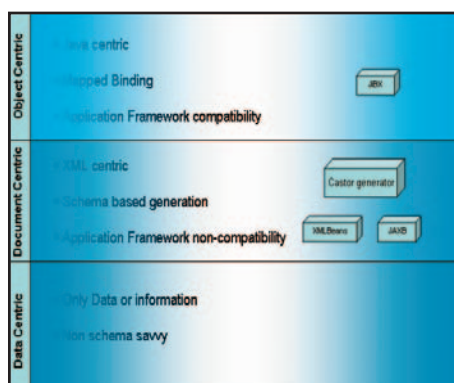


FIGURE 3 Characteristics of the different approaches

the binding framework to generate the target source code. The object model or document constructed from a schema can provide a fast way to start working with documents. This can be classified as a *document-centric approach*.

An alternative approach would be to use the mapped binding in a target language. It works with classes you define in your application in the target language rather than using a set of generated classes based on a

schema. This is more of an *object-centric approach*. This is convenient for the developers because of the closeness of the programming component to the target language.

Most of the conventional applications however stress working with the data, with XML only as a potential form for enabling data to be available to services or applications. These applications are bothered about the consistency of the data rather than the structure of the schema. Such usages of XML binding frameworks can be termed as *data-centric*.

Figure 2 illustrates a schematic mapping of potential usages of these categories in the context of different SOA applications. Figure 3 shows a summary of the characteristics of the approaches.

Data warehouses and database applications in SOA, e.g., Shared Data Services, can be highlighted as constituting the data management layer, and hence tend to concentrate on a *data-centric* use of XML binding frameworks. Similarly B2C, intranet, and Internet applications that stress business logic that deals with business entities,

constitute the business layer. Typically the business layer applications can be thought of as using an *object-centric* approach to access the entities, though in some cases a document-centric approach can be used.

Integration and B2B trading applications in the context of SOA that need to exchange contracts/schemas will likely use the *document-centric* approach.

## Some Key XML Binding Frameworks

We will be evaluating some key J2EE XML binding frameworks. A rough categorization of these frameworks is also shown in Figure 3. The frameworks are as follows.

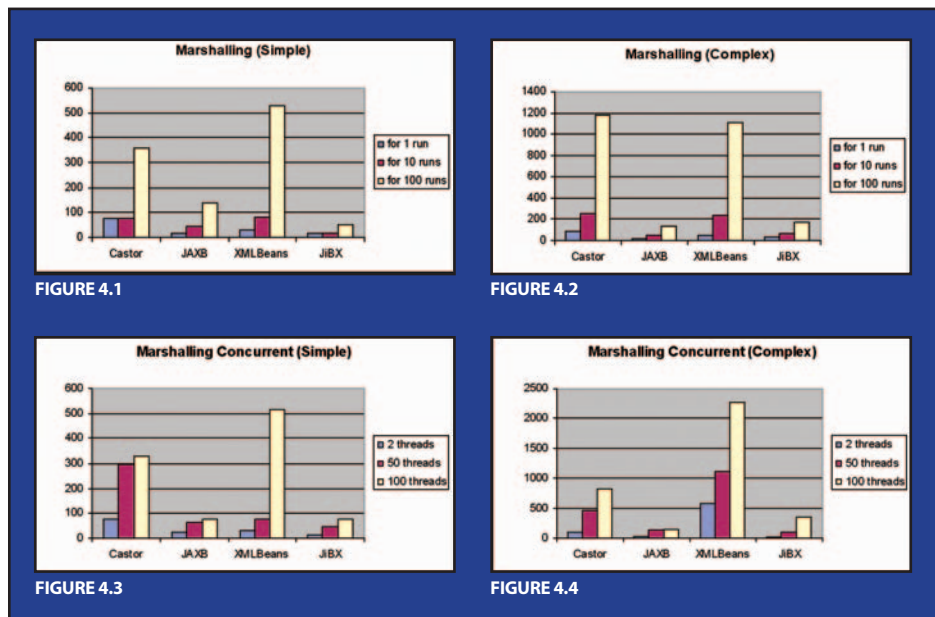
**Castor** ([www.castor.org](http://www.castor.org)): Castor XML can marshal almost any “bean-like” Java object to and from XML. The marshalling framework uses a set of descriptors to describe how an object should be marshaled and demarshaled from XML. Castor performs marshalling and demarshalling using Source Generator, which creates a Java object model and provides the binding to marshal, demarshall, and validate instances of XML schema. The source code generator takes as input XML schema document and produces the Java object model pertaining to the specifications of that schema.

**JiBX** ([www.jibx.org](http://www.jibx.org)): JiBX provides binding from XML data to Java objects. The binding is specified using a definition document, and JiBX uses a binding compiler to compile the definitions into Java byte code for efficiency. JiBX is designed for high performance.

Once the definition is ready, the binding compiler enhances binary class files produced by the Java compiler, adding code to handle converting instances of the classes to or from XML. The enhanced class files generated use a runtime component both for demarshalling and marshalling. The runtime uses a parser that implements the XML Pull API.

JiBX uses the XML Pull parsing technique. Instead of the parser calling methods in the handler to report document parts, one calls the parser to get each component in turn, easing the maintenance of the document state.

**XMLBeans** ([xmlbeans.apache.org](http://xmlbeans.apache.org)): XMLBeans uses XML Schema to compile to Java interfaces and classes that allow modification of XML instance data. XMLBeans provides setter and getter methods like JavaBeans. XMLBeans



XML nature	Serial Runs	Concurrent Runs
Simple	1	2
Complex	10	50
	100	100

TABLE 1 Test conditions

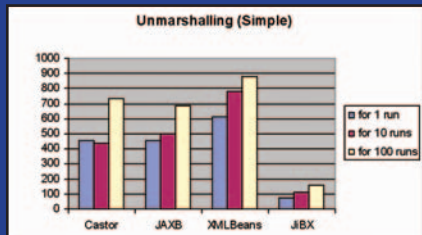


FIGURE 4.5

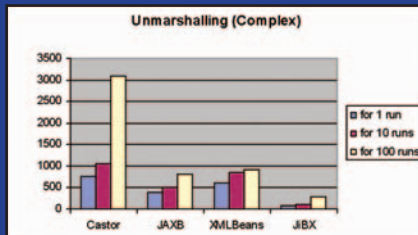


FIGURE 4.6

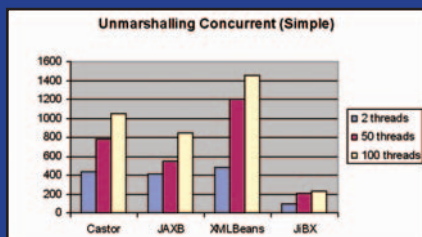


FIGURE 4.7

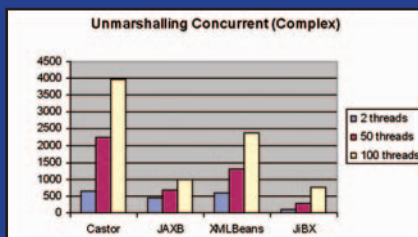


FIGURE 4.8

distinguishes itself by fully supporting XML Schema. During demarshalling an XML instance the full XML infoset is kept and is available to the developer. This is critical because a subset of XML is not easily represented in Java, e.g., the order of elements or comments might be needed in an application.

**JAXB** ([www.java.sun.com/xml/jaxb](http://www.java.sun.com/xml/jaxb)): JAXB is a standard for XML binding and consists of two parts. First, JAXB contains a compiler that reads a schema and produces the equivalent Java object model. This generated object model captures the structure of XML better than general-purpose APIs like DOM or SAX, making it a lot easier to manipulate XML content.

The second part is an API, through which applications communicate with generated code. The JAXB implementation from SUN JWSDP1.5 is used.

## Quantitative Evaluation of the Frameworks

We have quantitatively evaluated the XML binding frameworks on the following parameters.

**1. Marshalling Time** – Marshalling is the process of generating an XML representation for an object in memory. As with Java object serialization, the representation needs to include all dependent objects: objects referenced by our main object, objects referenced by those objects, and so on.

**2. Demarshalling Time** – Demarshalling is the reverse process of marshalling; it builds an object (and potentially a graph of linked objects) in memory from an XML representation.

**3. Memory Load** – JVM Heap memory used for the whole process of marshalling and demarshalling using the respective XML binding frameworks.

## Performance Tests

The test run used three samples of XML varied by its nature (segmentation factor). The test cases had to execute the selected binding frameworks on the sample XML files serially some x number of times. In addition to these, x number of threads have been executed concurrently. The combination of these test scenarios is an indicator of the memory load on a platform and time statistics

of each of these utilities (see Table 1).

## Test Suite

The JProbe profiler and Memory Debugger have been used to capture the performance, memory, and threading coverage issues of the respective binding frameworks. Each test case is run as a stand-alone session to have snapshots of the performance captured for each program execution. The tests are performed on a system with the following configuration:

- Pentium IV 1.8 GHz
- 512 MB RAM
- Virtual Memory: 768 MB

## Quantitative Results

### Marshalling Time

The graphs in Figures 4.1-4.4 represent the marshalling performance for simple and complex XML respectively. Higher performance from JIBX is due to the byte code enhancement done at the compile time. It avoids introspection at runtime.

## WSJ Advertiser Index

Advertiser	URL	Phone	Page
Blog-N-Play	<a href="http://www.blog-n-play.com">www.blog-n-play.com</a>	201-802-3000	43
Forum Systems	<a href="http://www.forumsys.com">www.forumsys.com</a>	801-313-4400	3
Fusionware	<a href="http://www.fusionware.net">www.fusionware.net</a>	801-313-4400	13
Gartner	<a href="http://gartner.com/us/adea">gartner.com/us/adea</a>	800-778-1997	9
ISSJ	<a href="http://www.ISSJournal.com">www.ISSJournal.com</a>	888-303-5282	41
KaPow Technologies	<a href="http://www.kapowtech.com">www.kapowtech.com</a>	800-805-0828	Cover III
LinuxWorld Expo	<a href="http://www.linuxworldexpo.com">www.linuxworldexpo.com</a>	800-657-1474	27
Mindreef	<a href="http://www.mindreef.com/tryout">www.mindreef.com/tryout</a>	603-465-2204	6
Parasoft	<a href="http://www.parasoft.com/products">www.parasoft.com/products</a>	888-305-0041	Cover II
Radview	<a href="http://www.radview.com/analyze">www.radview.com/analyze</a>	888-RADVIEW	15
Secrets of the XML Masters	<a href="http://www.sys-con.com/freecd">www.sys-con.com/freecd</a>	888-303-5282	53
Skyway Software	<a href="http://www.skywaysoftware.com">www.skywaysoftware.com</a>	813-288-9355	Cover IV
SYS-CON Publications	<a href="http://www.sys-con.com/2001/sub.cfm">www.sys-con.com/2001/sub.cfm</a>	888-303-5282	35
SYS-CON Reprints	<a href="http://www.sys-con.com">www.sys-con.com</a>	201-802-3024	39
Visual Paradigm	<a href="http://www.visualparadigm.com">www.visualparadigm.com</a>	852-2744-8722	25
Web Services Journal	<a href="http://www.wsj2.com">www.wsj2.com</a>	888-303-5252	19
WebRenderer	<a href="http://www.webrenderer.com">www.webrenderer.com</a>	613 6226 6274	5

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.

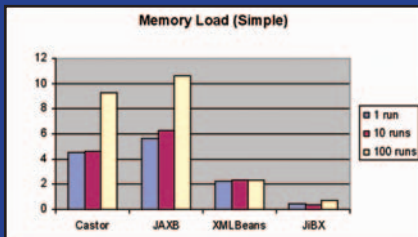


FIGURE 4.9

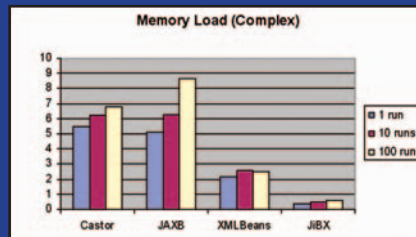


FIGURE 4.10

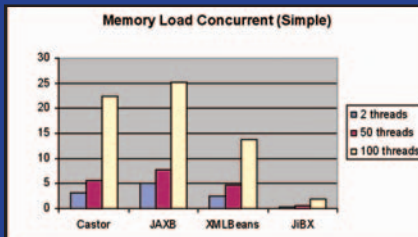


FIGURE 4.11

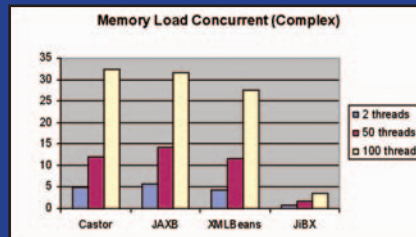


FIGURE 4.12

As seen from Figures 4.3 and 4.4, for multiple concurrent threads the results for XMLBeans worsen, whereas JiBX on the other hand has the same pattern.

#### Demarshalling Time

Refer to Figures 4.5 and Fig 4.6. Castor provides good support for data binding using code generation, but demarshalling performance is weak compared to others. The graph pattern is similar in the case of concurrent runs.

Castor uses the SAX2 parser and parses

the XML file in an event-driven manner, thus reducing its speed and performance as compared to JiBX.

JiBX uses XMLPull parsing technique, unlike the push parsing techniques such as SAX. Instead of the parser calling methods in the handler to report document components, one calls the parser to get each component in turn.

#### Memory Load

Memory load over the entire run (composed of demarshalling marshalling) is low for JiBX as it uses the XMLPull parsing technique during demarshalling, which doesn't require the parser to maintain the state in the document. Castor uses the SAX2 parser, which requires the parser to maintain the state in the document. JAXB is relatively inefficient in terms of memory usage.

#### Inferences

From the performance analysis of the four data-binding frameworks, a SWOT analysis is presented in Figure 5. Based on information from

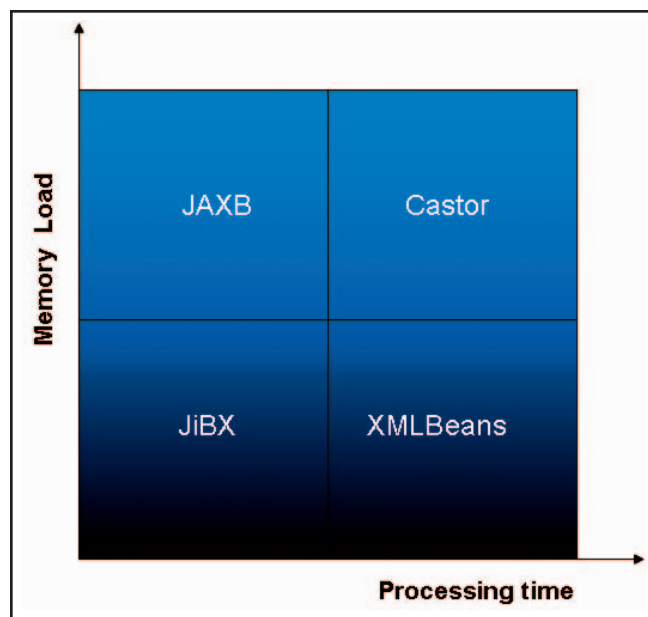


FIGURE 5 | SWOT analysis of the XML binding frameworks

Figure 5 and from the performance charts seen earlier, JiBX outperforms all of the other binding frameworks by a wide margin on both time and memory usage. JAXB scores well on marshalling and demarshalling time, but its memory usage is higher than the others. XMLBeans has a major performance hit but is more memory-efficient than Castor and JAXB. Depending on the specific requirement of an application, a suitable XML binding framework can be chosen.

#### Summary

We have illustrated in this article the pivotal role XML binding frameworks can play in context of SOA. To enable practitioners to make an informed choice about the appropriate XML binding platform, we have carried out a detailed quantitative analysis of some of the popular XML binding frameworks in the J2EE context, and presented the results. ©

#### References

- <http://castor.codehaus.org>
- [www.jibx.org](http://www.jibx.org)
- [www-106.ibm.com/developerworks/library/x-databd3](http://www-106.ibm.com/developerworks/library/x-databd3)
- <http://java.sun.com/xml/jaxb/>
- <http://java.sun.com/developer/technicalArticles/WebServices/jaxb/>
- <http://xmlbeans.apache.org>
- [www.xml.com/lpt/a/2003/09/03/binding.html](http://www.xml.com/lpt/a/2003/09/03/binding.html)
- <https://bindmark.dev.java.net/>
- *Professional XML, 2<sup>nd</sup> edition*. Wrox Press Ltd.
- Niranjana V., Anand S., and Krishnendu K. *Shared Data Services: An Architectural Approach*. Accepted in the International Conference of Web Services 2005, Florida, USA.

#### About the Authors

The authors are members of the Web Services COE (Center of Excellence) for Infosys Technologies, a global IT consulting firm, and have substantial experience in publishing papers, presenting papers at conferences, and defining standards for SOA and Web services. The Web Services COE specializes in SOA, Web services, and other related technologies. Dr. Srinivas Padmanabhuni heads the Web Services COE.

- ■ ■ [Bijoy\\_Majumdar@infosys.com](mailto:Bijoy_Majumdar@infosys.com)
- ■ ■ [Srinivas\\_P@infosys.com](mailto:Srinivas_P@infosys.com)
- ■ ■ [Ujval\\_Mysore@infosys.com](mailto:Ujval_Mysore@infosys.com)
- ■ ■ [Wikram\\_Sitaram@infosys.com](mailto:Wikram_Sitaram@infosys.com)





## This Month

### An Easy Introduction to XML Publishing – Part 3 of a Five-Part Series

PG Bartlett

In Part 1 of this series we discussed some of the key problems of capturing and sharing information and in Part 2 we looked at the critical components of a solution: modularization, automation, and XML. In part 3, we start getting technical – but in a nontechnical way. We examine the essential parts of building a solution, including developing data models (which are either DTDs or Schemas), designing stylesheets, and integrating various components of the solution.

### Exploring XML Schema Using JAXB in Enterprise Applications

Pushkar Varma

Code-generation tools are capable of significantly impacting productivity and are now taking on a more essential role as part of a developer's tool set. There are two categories of transformation tools: those whose output is used by software components, and those whose output is used by developers. With tools that generate output for developers, it becomes important to properly design the input to these tools so that the output is comprehensible and usable, and the input is maintainable by developers. Java Architect for XML Binding (JAXB) is one such tool that takes an XML Schema file as input, and transforms this input into a Java class model that is usable directly by a developer. Since an essential part of any application is its information model, it becomes important to properly design the XML Schema so that the generated class model is comprehensible and usable, and the schema itself is maintainable and reusable.

## An Easy Introduction to XML Publishing – Part 3 of a Five-Part Series

Developing a new publishing system

## 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

### XML Labs:

Product reviews, book reviews, tutorials, and standards analysis



# An Easy Introduction to XML Publishing – Part 3 of a Five-Part Series

Developing a new publishing system



WRITTEN BY  
PG Bartlett

**I**n Part 1 of this series we discussed some of the key problems of capturing and sharing information and in Part 2 we looked at the critical components of a solution: modularization, automation, and XML.

In part 3, we start getting technical – but in a nontechnical way. We examine the essential parts of building a solution, including developing data models (which are either DTDs or Schemas), designing stylesheets, and integrating various components of the solution.

## Data Models? Why Would I Care About Data Models?

We're fond of history lessons in this column, so let's go back to the year 1900. At that time, the average worker in the United States earned about \$500 a year and a bicycle cost \$600. No wonder the men on old-fashioned bicycles wore a top hat and tails – you had to be rich to own one!

The reason for the high cost of bicycles is that they were handmade. They were built one at a time and required a tremendous amount of labor to handcraft the parts and painstakingly adjust them until they fit together. Repairs were expensive and time-consuming as well, since the parts had to be made and fitted by hand.

Twenty-five years later, a Model T from Ford cost \$260 and median annual incomes had risen to over \$1500. Within a single generation, manufactured transportation had gone from an impossible luxury to widely affordable.

Several manufacturing innovations were behind this remarkable change, including the moving assembly line, specialization of workers, and interchangeable parts. Later, automated machinery would replace the tedious, dangerous work that humans performed, further reducing costs, increasing quality, and expanding variety.

Applying these same principles – automation, specialization, and interchangeable parts – to the creation and sharing of information delivers the same kind of benefits. Whether you're plan-

ning to automate manufacturing or publishing, one of the keys is to design interchangeable parts that you can be confident will fit together easily when the time comes to assemble them.

That's where XML and data models come in. Whether a data model is based on a DTD (which stands for Document Type Definition) or a Schema (invented more recently than DTDs; see the article at [www.arbortext.com/resources/xpn\\_june\\_03.html#tech](http://www.arbortext.com/resources/xpn_june_03.html#tech) for a comparison between DTDs and Schemas), the data model does for documents what design drawings do for interchangeable parts.

The data model describes all of the parts of a document along with the rules for how those parts may be combined. By following these rules when you create documents, software programs can automatically manipulate the documents later. Data models serve as the foundation of XML-based applications. All of the functionality in an XML publishing system rests on the data model. In most cases, if the data model changes, something else has to change as well. If you look at a data model in its raw form, whether it's a DTD or Schema, it looks scary. So we won't look. Instead, let's consider an abstract and highly simplified view in Figure 1.

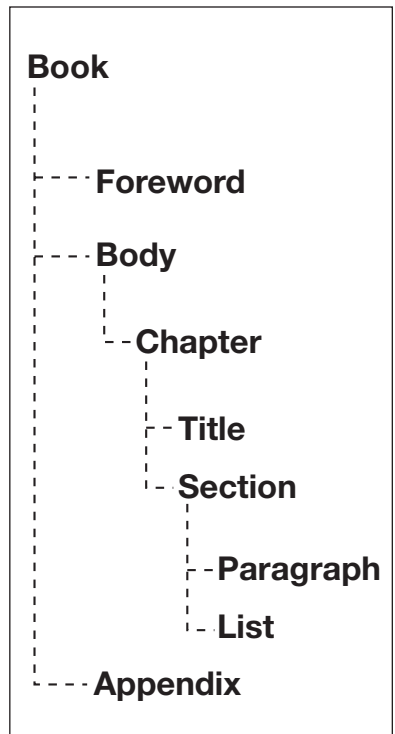


Figure 1 • Simplified view of a data model

You can see that a data model is like an organizational chart. The data model describes not only the parts (which we call “elements”) of a document, such as **chapter** and **section**, but also the hierarchy (for example, a **section** always comes within a **chapter**).

You can also see that the data model prescribes the order of the elements. In the example above, the main parts of a Document are **Foreword**, **Body**, and **Appendix**, and they must come in that order.

Data models also prescribe *how many* of an element can appear, such as “exactly one,” which you would want for the title of a **chapter**; “at least two,” which you would want for the items in a **list**; or “any number,” which you would want for **paragraphs**.

Gee, this seems pretty easy, doesn’t it? That’s only because we left out a lot of detail. So far, we showed examples of the organizational elements of a document, such as **chapter** and **section**. There are many reasons for capturing information into separate elements, such as:

- **Organization** (**chapter** and **section**) – We have already seen elements of this type, which prescribe the basic structure of the document. Documents may be books, articles, catalogs, datasheets, and so on, and each of these typically has some unique characteristics in its structure. For example, a book may have **chapter** elements while a catalog may have price elements.
- **Formatting** (**emphasis**) – Many elements exist only to make sure they have different formatting. For example, because emphasized words usually appear in italics, we designate an element for them such as **emphasis**. To provide a contrasting example, in virtually all cases we do not capture nouns or verbs as separate elements because we do not do anything different with them – they look the same as any other word in a sentence.

Even though XML separates content from formatting so that your information exists independent of the way it’s presented, you must consider your formatting goals while you design your data model. Too many times, we have seen organizations finalize their data models only to find that their stylesheets become very complex and expensive to develop and maintain, or they have to spend more time and money to revise their data models later, or they fail to meet all of their formatting design objectives.

- **Reuse** (**topic**) – One of the most important benefits to be gained from an XML publishing system is the capability to reuse information in multiple documents. Approaches to reuse have varied widely, but one emerging best practice is to reuse information at a “topic” level rather than at a chapter or section level. Whether this works for you depends heavily on the specifics of your application, so this is a prime area for expert assistance.
- **Personalization** – To enable the assembly of information for different audiences, you will want to use components for assembly, which can range in size from a cell in a table to a chapter. While some of those components may already exist as separate elements, you may have to create others explicitly for the purpose of including them in or omitting them from a specific rendition of the document.
- **Validation** (**partnum**) – In some cases, you may want to validate the content of an element against a database or other rules. For example, you may want to ensure that a Social Security number always be in the form of 3 digits, dash, 2 digits, dash, and 4 digits. Schemas are better at data validation than DTDs, but in either case

you will need to put the information to validate in a separate element.

- **Searching** (**definition**) – One of the potential benefits of XML is to improve the searchability of information. For example, you could improve the relevance of a search by allowing users to search for service information only within **service procedures** instead of searching across all document types.

“Whether you’re planning to automate manufacturing or publishing, one of the keys is to design interchangeable parts that you can be confident will fit together easily when the time comes to assemble them”

- **Translation** (**productname**) – To simplify or automate translation to other languages, you may want to designate terms such as product name or company name that are translated through a database or some other method instead of being translated manually.
- **Behavior** (**link**) – Cross-references and hyperlinks not only look different than other elements, but they also have specific behaviors associated with them. You could also implement more elaborate behavior for an element, such as making footnotes on Web pages pop up when the user’s mouse hovers over them. Regardless, you must specifically identify as elements those parts of a document to which you want to assign specific behaviors.
- **Other automation** – For anything you need to automate, you may need to create separate elements so that you can write software to find and manipulate them.

Even though we started with the data model in this article, in reality you must consider your system architecture and specific choices of products as you create your data model to make sure you can accomplish all of your goals. Whole books have been written about the process of creating a data model. It takes days of concentrated discussion plus weeks of analysis to come up with a data model or a set of data models appropriate to your business. You will find more information about creating DTDs, including links to even more resources at [www.arbortext.com/resources/xpn\\_july\\_03.html#tech\\_](http://www.arbortext.com/resources/xpn_july_03.html#tech_).

Once you create a description of your data model, encoding it as a DTD or Schema is the easy part. However for creating your data models in the first place, we recommend that you enlist the aid of an expert as well as plan to invest some time in it yourself. It’s just as easy to overdo a data model and make it too complex as it is to omit elements that you’ll wish for later.



## Thank Goodness Data Modeling Is Over! What Comes Next?

There are several major areas of design and development in the creation of your XML publishing system. The following paragraphs highlight each of these areas.

- **Conversion** – You will probably want to convert some of your existing word processing and desktop publishing files to XML. Because conversion tends to be expensive, you should only convert documents that you're continuing to improve or that you're planning to reuse/repurpose for new documents.

Conversion is expensive because it's complex: you're converting documents with a flat structure that are filled with inconsistencies to files with a hierarchical structure and absolute consistency.

Some documents are so complex and inconsistent that re-keying is the best approach. Arbortext partners such as Data Conversion Labs ([www.dclab.com/](http://www.dclab.com/)) offer such services. On the other hand, your documents may be sufficiently consistent and simple that you can use software to convert automatically or semi-automatically, most likely requiring manual cleanup afterwards. Setting up this software requires creating a "map" between styles and tags. Arbortext, our partners, and other companies offer software and setup services, which the next article in this series will describe in more detail.

- **Stylesheets** – Because an XML document contains no instructions for formatting its contents for viewing or printing, those instructions have to exist somewhere. That's where stylesheets come in. Stylesheets contain instructions for how to display or print each element (you can learn a lot more at [www.arbortext.com/resources/xpn\\_sep\\_03.html#tech](http://www.arbortext.com/resources/xpn_sep_03.html#tech)).

You'll need stylesheets both for editing and for publishing. Although you could ask your authors to create and edit XML without a stylesheet, they will put up stiff resistance because they will intensely dislike the loss of on-screen formatting because they have grown accustomed to seeing visual cues about the meaning of each element. In other words, they'll want to see the contents of an emphasis tag displayed on the screen in italics instead of just seeing the emphasis tag.

Without a Stylesheet	With a Stylesheet
This is <emphasis>very</emphasis> easy!	This is <i>very</i> easy!

- You may need many different stylesheets. You will need one for each different XML editing tool that you use, for each type of document you produce, for each medium to which you publish that document, and for each format variation. For example, if you are producing both print and Web versions of a product catalog and a technical manual, and if you have to produce the print version on both 8 1/2 x 11 and A4 paper, you would need the following stylesheets:
  - Editor stylesheet for catalog
  - Editor stylesheet for technical manual
  - Web stylesheet for catalog
  - Web stylesheet for technical manual
  - Print stylesheet for catalog for 8 1/2 x 11
  - Print stylesheet for catalog for A4

- Print stylesheet for technical manual for 8 1/2 x 11
- Print stylesheet for technical manual for A4

You may be able to consolidate some of these stylesheets. For example, you may be able to use the print stylesheets for editing, and you may be able to combine the different paper sizes into the same stylesheet.

- **Translation** – Many companies who implement XML publishing systems find that the cost savings on translation alone more than justify their investment in a new system.

To achieve these cost savings, there are several steps. First, to reduce the total amount of information that you send to a translation house, you would break up your information into smaller modules. Often, the size of information modules that you choose for reusing information works well for reusing translations too.

Second, to simplify the translation process, you may wish to simplify the terminology and sentence structure that your authors use. Several companies provide software to enforce simplified English, such as Smart Communications ([www.smartny.com/](http://www.smartny.com/)).

Third, because you will be managing many components in multiple languages, you may want to implement a content management system from one of our repository partners ([www.arbortext.com/partners/index.asp](http://www.arbortext.com/partners/index.asp)).

Finally, you will want to obtain translation software from a company such as Trados or work with a company that specializes in translating XML such as OmniLingua.

- **Publishing** – One of the primary benefits of XML publishing is the capability to publish automatically to multiple types of media for multiple audiences. Much of the work is in setting up the stylesheets, described earlier, because they contain the instructions for formatting your documents for each type of media.

Automated publishing software applies a stylesheet to your content to produce a Web, print, PDF, or other version of your information. Multiple technologies are required to support multiple types of media, although a few vendors such as Arbortext bring these together under a single architecture.

To produce documents automatically, your requirements may extend beyond the formatting capabilities of the technology you're using. For example, you may want to generate content automatically, either by deriving it from your document to produce, say, a list of figures, or by extracting it from a database, such as pulling part descriptions based on part numbers.

Delivering information in multiple languages adds another dimension of complexity. You need to be sure not only that the publishing software can support the composition of all of the languages you need, but that it also provides other capabilities in support of those languages such as hyphenation and index sorting.

To provide different information based on the audience, we could spend an entire article – and recently did ([www.arbortext.com/resources/xpn\\_oct\\_03.htm](http://www.arbortext.com/resources/xpn_oct_03.htm)) – discussing the ways to approach building a system to provide customized information on demand. The main point of the article was that you could

either use a “master document” approach or an “assembly” approach to creating customized views of your information.

Since the master document approach is easier to implement, let’s consider it in more detail. From a master document, you can publish subsets of that document for different audiences. For example, if you make products that come with a variety of options, you could tailor each customer’s owner’s manual to match the specific combination of options that the customer selected.

To do this, you first add information to your document about the audience for each part of that document. You can customize information all the way down to a word in a sentence or a cell in a table. Then, when you publish the document, you select the audience and the publishing software uses your selection plus the additional audience information you entered in the document to decide which information to omit from the document for publishing.

- **Library Services (CMS)** – The primary capabilities of a content management system (CMS) are “library services,” which include file storage and retrieval, version control, and access control; most CMSs provide workflow as well. Arbortext does not provide library services or workflow; instead, we work with a number of repository partners who provide these capabilities.

The value of such services increases with the amount of information you’re managing, the number of people who contribute or edit your information, the geographical dispersion of

your people, and the complexity of your processes. About two-thirds of Arbortext’s customers have implemented a CMS along with their implementation of Arbortext.

Considerable effort is involved in setting up a CMS, such as assigning users and permissions, setting up directories, designing workflows, and implementing other automation. You may also want to customize the metadata to suit your specific needs.

- **Integration** – Finally, you have to tie together all of the various products and technologies in your system so that your users can easily learn and operate the system.

One of the main issues to consider is whether your organization or your vendors are responsible for updating your integration as software versions change.

In Part 4, we’ll provide an overview of the XML publishing architecture and describe the tools available for creating content. ☛

## AUTHOR BIO

PQ Bartlett is vice president of product marketing at Arbortext, where he is responsible for corporate positioning, marketing strategy, and product direction. Bartlett joined Arbortext in 1994, bringing more than 18 years of experience in both technical and marketing positions at leading-edge high technology companies. He is a frequent presenter at major industry events and has been invited to speak and chair sessions at Comdex, Seybold Seminars, XML conferences, AIIM conferences, and others.

pgb@arbortext.com

# 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 400 exclusive *XML-J* articles!

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

**ORDER AT [WWW.SYS-CON.COM/FREECD](http://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.

**SYS-CON  
EVENTS**

*Only from the World's Leading i-Technology Publisher*



WRITTEN BY PUSHKAR VARMA

# Exploring XML Schema Styles Using JAXB in Enterprise Applications

## Choose the right XML Schema for your enterprise applications

**C**ode-generation tools are capable of significantly impacting productivity and are now an essential part of a developer's tool set. There are two categories of transformation tools: those whose output is used by software components, and those whose output is used by developers. With tools that generate output for developers, it becomes important to properly design the input to these tools so that the output is comprehensible and usable, and the input is maintainable by developers. Java Architecture for XML Binding (JAXB) is one such tool that takes an XML Schema file as input, and transforms this input into a Java class model that is usable directly by a developer. Since an essential part of any application is its information model, it becomes important to properly design the XML Schema so that the generated class model is comprehensible and usable, and the schema itself is maintainable and reusable.

This article explores various styles for designing XML Schemas and assesses each style in terms of certain quality attributes. Figure 1 shows how an information model is transformed to a class model using JAXB, and shows which qualities of the artifacts should be assessed based on a set of quality attributes.

There are four XML Schema styles based on various combinations of the global and local scope of elements and complex types. The styles are assessed by evaluating how the JAXB output satisfies comprehensibility and usability concerns, and how well the XML Schema satisfies modifiability and reusability concerns. The JAXB component's quality is not assessed since its implementation is based on a standard and there is little variability in its behavior; once a developer learns to

use it, its model behavior is predictable.

### The Design Context

When designing an XML-centric application, two concerns need to be addressed:

1. *Functional Concerns* - How to represent an information model, i.e., a data model or a UML class model, in XML using XML Schema so that any subset of the information model can be produced and consumed in a flexible and extensible manner, and the information model can be correctly represented in XML.

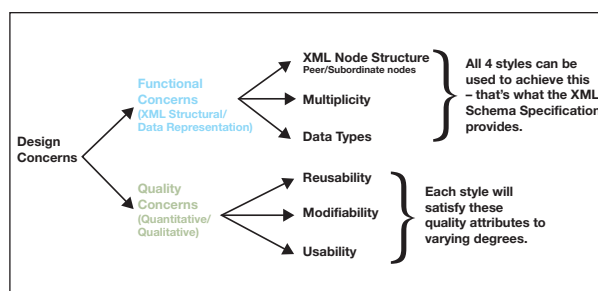


Figure 2 • Design concern tree

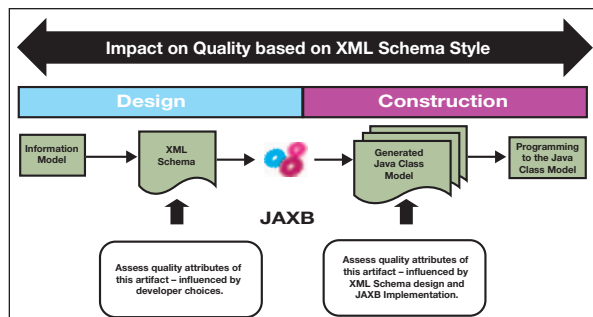


Figure 1 • Quality attributes during development

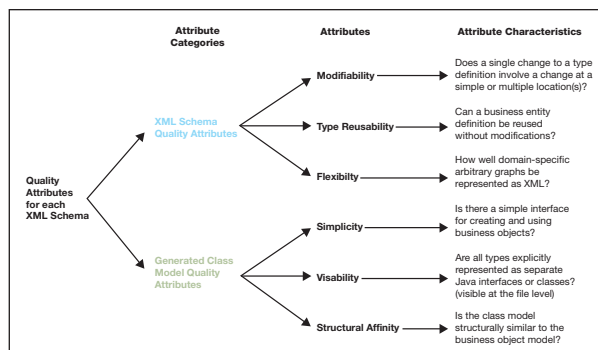


Figure 3 • Quality attribute tree

### AUTHOR BIO

Pushkar Varma is an IT architect with IBM Global Services whose interests include application architectures, architecture assessment methods, Web application development using Web services and J2EE, and rich clients using Eclipse.



2. *Quality Concerns* – How to design an XML Schema that is maintainable and reusable, and allows processing by a tool (JAXB) so that the generated output from the tool is comprehensible and usable by a developer.

The Quality concerns are the primary focus of this article, since the design of the XML Schema will be a significant contributing factor to how well quality concerns are satisfied. Figure 2 shows a design concern tree with the two main design concern paths and further subconcerns for each path. The functional concerns are primarily satisfied by the XML Schema specification, while the Quality concerns will be satisfied to varying degrees based on the XML-to-Java binding framework, and the design of the XML Schema used as input to the framework.

### XML Schema Styles

XML Schema styles are based on various combinations of global and local scopes for XML Schema *Element* and *Complex Type* declarations. Table 1 shows the four styles.

Why is one style chosen over another? Normally, a developer may choose to use *Style 1 <L, L>* over *Style 3 <L, G>* simply due to level of knowledge and experience, lack of time, or simply a preference. However, each style will have different degrees of influence on the qualities of the XML Schema, and the resulting generated Java code. For example, maintainability of an XML Schema file will be significantly impacted based on what style is used, and comprehensibility of the resulting Java code will be sensitive to each style as well.

### Quality Attributes for Evaluating XML Schema Styles

Figure 3 shows a tree view of quality attributes that will be used to assess the quality of the XML Schema and the generated class model for each XML schema style. This list is not comprehensive, but is only meant to be a starting point for introducing quality assessment of XML Schema styles in the context of code generation tools. Each quality attribute will be assessed simply as High or Low, so as not to get overly complex. Table 2 shows what High or Low means for each quality attribute.

When creating an XML Schema, element specifications and type specifications are normally defined separately since type specifica-

XML Schema Constructs		Elements (non-root nodes)	
		Local	Global
Complex Type	Local	<b>Style 1 &lt;L, L&gt;</b> Example: <Element name="Root"> <Element name="E1"> <ComplexType name="C1"/> <ComplexType name="C2"/> </Element> <Element name="E2"> <ComplexType name="C1"/> </Element> </Element>	<b>Style 3 &lt;L, G&gt;</b> Example: <Element name="Root"> <Element ref="E1"/> <Element ref="E2"/> </Element> <Element name="E1"> <ComplexType name="C1"/> <ComplexType name="C2"/> </Element> <Element name="E2"> <ComplexType name="C1"/> </Element>
	Global	<b>Style 2 &lt;G, L&gt;</b> Example: <Element name="Root"> <Element name="E1" type="C1"/> <Element name="E2" type="C2"/> </Element> <ComplexType name="C1"/> <ComplexType name="C2"/>	<b>Style 4 &lt;G, G&gt;</b> Example: <Element name="Root"> <Element ref="E1"/> <Element ref="E2"/> </Element> <Element name="E1" type="C1"/> <Element name="E2" type="C2"/> <ComplexType name="C1"/> <ComplexType name="C2"/>

Table 1 • XML Schema styles

Quality Attributes	Meaning of Scale	
	High	Low
XML Schema Quality Attributes		
Modifiability	Easy to change a type definition at a single location	Change to a type definition requires changes at multiple locations.
Type Reusability	A type definition is specified so that it can be easily referenced for use by another type specification.	A type definition is specified so that it cannot be easily referenced for use by another type specification without significant refactoring – may require elimination of duplication.
Flexibility	An arbitrary subset of an object graph made of domain objects can be represented as XML.	Object graphs are constrained to specific domain types as root XML elements. An arbitrary subset of an object graph cannot be represented separately in XML.
Class Model Quality Attributes		
Simplicity	The API is simple to use for creating objects and using object accessors.	The API is difficult to use for creating objects, and not easily understood.
Visibility	All types can be identified easily through a visual inspection of the Java files.	All types are not represented as individual Java class files. It is difficult to determine by generated files what types are represented without examination of the inner types.
Structural Affinity	The class model more closely resembles the business object model.	The class model does not closely resemble the business object model. Operations are not provided on the core business entity type, but are on a supporting type.

Table 2 • High and Low assessments of quality attributes

tions represent pure business domain entities, and element specifications represent roles of their underlying type.

### Example

First, a business object model (BOM) is described that shows the key business entities and relationships. Then the BOM is specified in XML Schema using each of the four styles discussed earlier. Each XML Schema is then input

into JAXB to generate a class model, whose UML representation is presented as well. Each style's XML Schema and generated class model are assessed using the quality attributes.

### Business Object Model

Figure 4 shows an example business object model. This model shows simple aggregate associations, e.g., Customer to Order, and multiple associations between two entities with

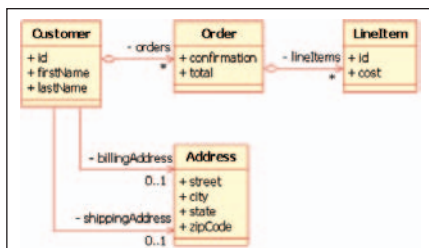


Figure 4 • Business Object Model

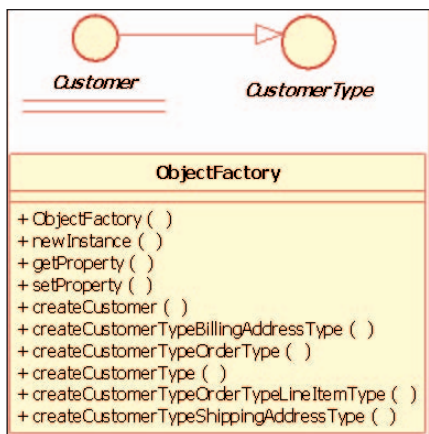


Figure 5 • Style 1 generated class model

different roles, e.g., Customer to Address. In order to understand the dynamics among the XML Schema, JAXB transformation, and

generated class model, the model was kept simple, hence it leaves out self-relationships and inheritance.

Listings 1 to 4 and Figures 5 to 8 show a representation of the object model in XML Schema using the four styles, and their corresponding generated JAXB class models, respectively.

### Assessing the Styles

Based on the example analyzed above, quality assessment of each style is summarized in Tables 3 and 4. For the XML Schema quality, styles 2 and 4, the ones with global complex types, offer the highest modifiability and reusability, and style 3 and 4, the ones with global complex types, have the highest flexibility. For the generated class model, style 2 has the highest simplicity, while any style with at least a global element or complex types has a high structural affinity.

### Recommendations

Table 5 summarizes the assessment of quality attributes for the XML Schema and the generated class model based on the observations above. The use of a particular style will be based on requirements, skills of the developer, and development time constraints. However, in general, it can be observed that styles 2 and

4 seem to have the most assessments of High quality for both the XML Schema and the generated class model. Both of these styles have global complex types, which leads to a schema with highly flexible and reusable type specifications regardless of its element specifications. These two styles only differ in the scope of their element specifications, but based on needs, it may be desirable to use one style over the other. In general, based on this assessment, XML schemas should be defined with global type specifications. Styles 2 and 4 would be more appropriate for a formalized development environment with large XML Schemas, whereas styles 1 and 3 would be acceptable for small XML Schemas.

This approach of assessing the quality of an XML Schema and a generated class model, within the context of a tool such as JAXB, should provide value in determining which style to use.

### Summary

This article has discussed the four styles that are possible based on the various combinations of scopes of complex type and element specifications. Taxonomy of quality attributes was presented, which is to be used for quality assessment of each of the four styles. An

Quality Attributes	XML Schema Styles			
	Style 1<L,L>	Style 2<G,L>	Style 3<L,G>	Style 4<G,G>
<b>Modifiability</b>	<ol style="list-style-type: none"> <li>1. All types are defined within one root element.</li> <li>2. Type definitions with different roles are specified multiple times.</li> <li>3. A sequence of relationships becomes extremely nested.</li> <li>4. Changes to type specifications that are used multiple times will require duplicate changes such as the Shipping Address and Billing Address complex types. Both are addresses, however, defining local types results in duplication of specifications.</li> <li>5. The schema gets more nested and confusing as more types are added.</li> <li>6. Changes to elements specifications and type specifications cannot occur separately, since types are defined inline to the element specifications.</li> </ol>	<ol style="list-style-type: none"> <li>1. Type specifications are defined and named separately, which will allow each type to be modified easily and only once; there is no duplication.</li> <li>2. It is easy to understand the types and their relationships to other types through local element specifications.</li> <li>3. A role (element) or the type in a relationship can be changed separately.</li> <li>4. Type specification changes are more granular with this Schema style, and do not require duplication of effort.</li> </ol>	<ol style="list-style-type: none"> <li>1. This has a similar problem of duplicate specifications as Style 1. Changes to an address specification will require a change to the Billing Address and a change to the Shipping Address.</li> <li>2. Elements and types within a relationship can still be separately modified similar to Style 2.</li> <li>3. Type and element specifications are not separated, as opposed to Style 2.</li> <li>4. Changes to type specifications will require duplicate changes, such as the BillingAddress and ShippingAddress types.</li> </ol>	<ol style="list-style-type: none"> <li>1. This style provides the most separation between type and element specifications, which allows a type specification to be modified independently of element specifications.</li> <li>2. Proliferation of global complex type specifications within a schema may create unwanted flexibility that will allow unneeded root nodes to be defined in XML.</li> <li>3. This also creates more confusion since it becomes difficult to determine what element should be the root nodes.</li> </ol>
<b>Type Reusability</b>	It is not possible to reuse types in this schema style since all elements and types are defined inline as children of the root element.	Type specifications can be reused simply by reference. For example, the Address type is referenced by the Customer type with two different roles, ShippingAddress and BillingAddress. However, the same Address type can be used for another schema with different roles, such as Personal or Business Address.	Since types are defined as local elements, it is not as easy to reuse a type without explicitly getting its element specification.	This style also provides the greatest element and type reusability since element specifications can be reused by other future types without modifications.
<b>Flexibility</b>	From all of the four styles discussed in this article, this style results in the least flexible schema. For this specific example, only the Customer domain object with its related objects can be represented in XML; no other subset of this schema can be generated separately. For instance, the Shipping Address, or any other type under Customer, cannot be generated as a separate XML document.	This style, with its local element specifications, does not provide much flexibility for an arbitrary subset of an object graph to be represented as an XML document.	This style is very flexible since arbitrary subsets of an object graph can be represented as XML documents. However, due to local complex types, reusability and modifiability is greatly reduced.	This style provides flexibility with its global element and complex type specifications. Any arbitrary subset of an object graph can be represented as an XML document.

Table 3 • Observations of XML Schema for each style

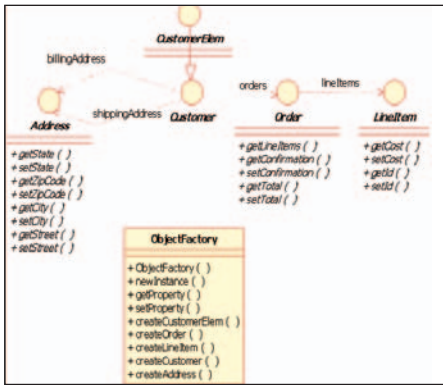


Figure 6 • Style 2 generated class model

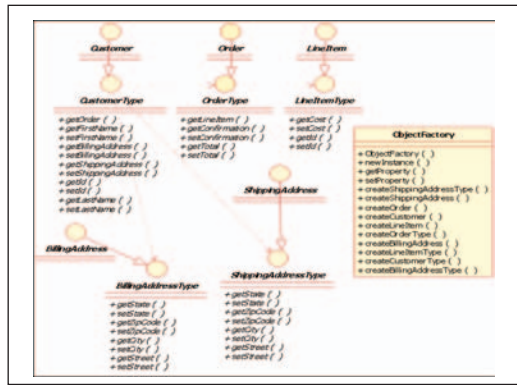


Figure 7 • Style 3 generated class model



Figure 8 • Style 4 generated class model

example business object model was used and represented in XML Schema form for each of the four styles. Finally, a recommendation was provided about when each XML Schema style should be used.

## Resources

- Check out View the Quality Measures Taxonomy ([www.sei.cmu.edu/str/taxonomies/view](http://www.sei.cmu.edu/str/taxonomies/view))

[qm.html](http://qm.html)) at Carnegie-Mellon University's Software Engineering Institute (SEI): [www.sei.cmu.edu/](http://www.sei.cmu.edu/)

- The concept of a quality tree was adopted from the CMU/SEI Technical Report, Quality Attributes: [ftp://ftp.sei.cmu.edu/pub/documents/95.reports/pdf/tr021.pdf](http://ftp.sei.cmu.edu/pub/documents/95.reports/pdf/tr021.pdf)
- Read the W3C's ([www.w3.org](http://www.w3.org)) XML Schema 1.0 for more understanding about XML Schemas:

[www.w3.org/XML/Schema#dev](http://www.w3.org/XML/Schema#dev)

- Download the Java Web Services Developer Pack 1.4 (<http://java.sun.com/webservices/jwsdp/index.jsp>) at Sun's Java site: <http://java.sun.com/>
- Check out Rational XDE for creating UML diagrams: [www-306.ibm.com/software/awdtools/developer/rosexde/](http://www-306.ibm.com/software/awdtools/developer/rosexde/)

pvarma@us.ibm.com

Quality Attributes	XML Schema Styles			
	Style 1<L,L>	Style 2<G,L>	Style 3<L,G>	Style 4<G,G>
<b>Simplicity</b>	The object factory generated by JAXB, used to create all types defined in the XML Schema, shows all of the factory methods, but the factory method names are very long, and inclusive of all type names in a relationship sequence. This results in an API that is difficult to understand and use.	The object factory's interface is quite easy to understand, and the developer is able to determine how to create each type independently of the other.	1. This class model is similar to Style 2's class model, however, with added complexity. 2. Each local type has a separate interface, and each element is also represented as a separate Type interface. 3. The Object Factory is a bit more complex since it contains factory methods for all interfaces relating to local types, and interfaces relating to global elements. This results in factory methods that can only be distinguished by the word "Type" as a suffix, which only leads to confusion about which method is used to create a business entity and which is used to create an element. For example, createOrder and createOrderType are similar in name.	Due to a clear separation of type and element specifications, the object factory has factory methods for the types and elements. This is programmatically more confusing since it is more natural to create a type and its relationships through type references, and have the XML elements generated in the JAXB implementation. With factory methods for elements, it requires more effort to determine the underlying types and how the types and elements, once created, interplay to create the desired object graph.
<b>Visibility</b>	There are only two interfaces generated in the class model, the Customer business entity, and CustomerType. Where are all of the other types? All of the inline elements and types defined earlier are not identifiable at the Java file level as separate manageable files, but instead are inner types that are available through the Customer interface.	Unlike Style 1's class model, all types are separate interfaces in this class model.	All types and elements are separate interfaces in the class model.	All types and elements are separate interfaces in the class model.
<b>Structural Affinity</b>	1. All associations to Customer are inner types, which is semantically correct since they are composition relationships. 2. Each separate role for a single type, e.g., Address, is treated as a separate type, e.g., Billing and Shipping address types.	1. This style closely represents the object model. Different roles on a single type are treated as a single type. 2. The only single root element, CustomerElem, can be separated modeled from the types. 3. The separate roles will be realized in the generated XML.	All types are shown in this class model; both types and elements are represented as interface files. All of these interfaces provide flexibility, but proliferation of interfaces contributes to reducing simplicity.	This style allows for the most structural similarity to the original business object model. For example, the AddressType is a separate type, with the ShippingAddress and BillingAddress as empty subtypes.

Table 4 • Observations of Class Model Quality for each style

XML Schema Style	Schema Quality Attributes			Generated Class Model Quality Attributes		
	Modifiability	Type Reusability	Flexibility	Simplicity	Visibility	Structural Affinity
Style 1<L,L>	Low	Low	Low	Low	Low	Low
Style 2<G,L>	High	High	Low	High	High	High
Style 3<L,G>	Low	Low	High	Low	High	Low
Style 4<G,G>	High	High	High	Low	High	High

Table 5 • Summary of quality assessment



### Listing 1: Style 1 Schema

```
<xsd:element name="Customer">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Order" minOccurs="0"
maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="LineItem" minOccurs="0"
maxOccurs="unbounded">
              <xsd:complexType>
                <xsd:attribute name="id" type="xsd:string"/>
xsd:attribute
                <xsd:attribute name="cost" type="xsd:
float"/>
</xsd:complexType>
              </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="confirmation" type="xsd:
int"/>
</xsd:complexType>
            <xsd:attribute name="total" type="xsd:float"/>
</xsd:complexType>
          </xsd:element>
        <xsd:element name="ShippingAddress">
          <xsd:complexType>
            <xsd:attribute name="street" type="xsd:string"/>
xsd:attribute
            <xsd:attribute name="city" type="xsd:string"/>
xsd:attribute
            <xsd:attribute name="state" type="xsd:string"/>
xsd:attribute
            <xsd:attribute name="zipCode" type="xsd:int"/>
xsd:attribute
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="BillingAddress">
          <xsd:complexType>
            <xsd:attribute name="street" type="xsd:string"/>
xsd:attribute
            <xsd:attribute name="city" type="xsd:string"/>
xsd:attribute
            <xsd:attribute name="state" type="xsd:string"/>
xsd:attribute
            <xsd:attribute name="zipCode" type="xsd:int"/>
xsd:attribute
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:int"/>
</xsd:complexType>
    <xsd:attribute name="firstName" type="xsd:string"/>
</xsd:complexType>
    <xsd:attribute name="lastName" type="xsd:string"/>
</xsd:complexType>
  </xsd:element>
```

### Listing 2: Style 2 Schema

```
<xsd:complexType name="Customer">
  <xsd:sequence>
    <xsd:element name="Orders" type="Order" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="ShippingAddress" type="Address"
minOccurs="0" maxOccurs="1"/>
    <xsd:element name="BillingAddress" type="Address"
minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:int"/>
  <xsd:attribute name="firstName" type="xsd:string"/>
  <xsd:attribute name="lastName" type="xsd:string"/>
</xsd:complexType>

<xsd:element name="CustomerElem" type="Customer"/>
</xsd:element>
<xsd:complexType name="Order">
  <xsd:sequence>
    <xsd:element name="LineItems" type="LineItem" minOc-
curs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="confirmation" type="xsd:int"/>
  <xsd:attribute name="total" type="xsd:float"/>
</xsd:complexType>
```

```
<xsd:complexType name="LineItem">
  <xsd:attribute name="id" type="xsd:string"/>
  <xsd:attribute name="cost" type="xsd:float"/>
</xsd:complexType>
<xsd:complexType name="Address">
  <xsd:attribute name="street" type="xsd:string"/>
  <xsd:attribute name="city" type="xsd:string"/>
  <xsd:attribute name="state" type="xsd:string"/>
  <xsd:attribute name="zipCode" type="xsd:int"/>
</xsd:complexType>
```

### Listing 3: Style 3 Schema

```
<xsd:element name="Customer">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Order" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element ref="ShippingAddress" minOccurs="0"
maxOccurs="1"/>
      <xsd:element ref="BillingAddress" minOccurs="0"
maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:int"/>
    <xsd:attribute name="firstName" type="xsd:string"/>
    <xsd:attribute name="lastName" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="Order">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="LineItem" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="confirmation" type="xsd:int"/>
    <xsd:attribute name="total" type="xsd:float"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="LineItem">
  <xsd:complexType>
    <xsd:attribute name="id" type="xsd:string"/>
    <xsd:attribute name="cost" type="xsd:float"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="ShippingAddress">
  <xsd:complexType>
    <xsd:attribute name="street" type="xsd:string"/>
    <xsd:attribute name="city" type="xsd:string"/>
    <xsd:attribute name="state" type="xsd:string"/>
    <xsd:attribute name="zipCode" type="xsd:int"/>
  </xsd:complexType>
</xsd:element>

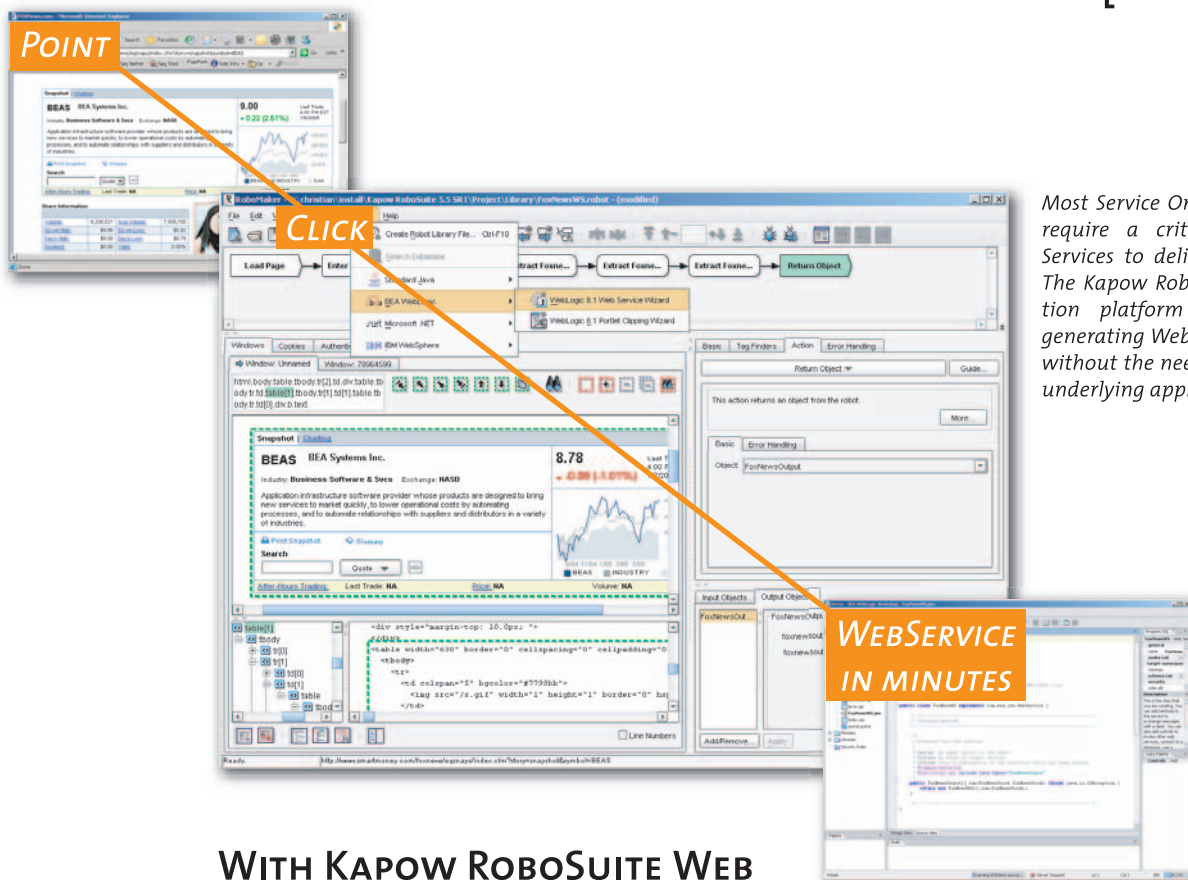
<xsd:element name="BillingAddress">
  <xsd:complexType>
    <xsd:attribute name="street" type="xsd:string"/>
    <xsd:attribute name="city" type="xsd:string"/>
    <xsd:attribute name="state" type="xsd:string"/>
    <xsd:attribute name="zipCode" type="xsd:int"/>
  </xsd:complexType>
</xsd:element>
```

To view Listing 4, please view this article online at <http://webservices.sys-con.com/read/issue/archives/> (Vol: 5 Iss: 7).

# LEVERAGE EXISTING IT TO GENERATE WEB SERVICES

[ in minutes ]

For  
free online  
demonstration,  
Gartner "Cool Vendor"  
release, and software trial:  
[www.kapowtech.com/wsprint](http://www.kapowtech.com/wsprint)



Most Service Oriented Architectures require a critical mass of Web Services to deliver tangible results. The Kapow RoboSuite Web Integration platform supports this by generating Web Services in minutes without the need to re-program the underlying application.

## WITH KAPOW ROBOSUITE WEB INTEGRATION PLATFORM YOU GET

- Fast generation of Web Services of any web application – simple or complex – with visual point and click
- Non-intrusive solution using the browser front-end to access any web-application
- Wizards enable quick, low cost deployment in SOA frameworks for production-ready SOA

[kapowtech.com](http://kapowtech.com)

Kapow Technologies is a leader in Web Integration – a new integration paradigm using the broadly available web front-end. The Kapow RoboSuite platform uniquely enables flexible and fast integration of content, data and applications from any source available through a browser into portals, content management systems, applications, databases or web services.

**kapow**  
TECHNOLOGIES

# WANTED: SOA NO EXPERIENCE NECESSARY

## BUILD, GOVERN, & *DEPLOY* Business Level Solutions, Processes, and Services



## CALL NOW For Case Study Details on How to Implement SOA Quickly and Easily

"From a support and knowledge standpoint, Skyway is untouchable. What's immeasurably beneficial is how quickly Skyway responds; it is truly phenomenal. A lot of people only say they can do what Skyway delivers. With Skyway you will do more with less." — Jim Garcia, CIO, Enporion

"Developing with Skyway Software gives us a much more rapid response time to our business community, it has a lower TCO, and it accelerates SOA and Web Service adoption throughout BAT."  
— Kevin Poulter, British American Tobacco, a \$30 billion powerhouse

White Paper — Case Study — FREE Evaluation Download

[www.skywaysoftware.com](http://www.skywaysoftware.com)



(813) 288-9355

Copyright 2005 Skyway Software, Inc. All Rights Reserved. All logos and company names are trademarks or service marks of their respective companies.



# Web Services

**.NET J2EE XML JOURNAL**

July 2005 Volume 5 Issue 7

## Open Grid Service Architecture (OGSA)

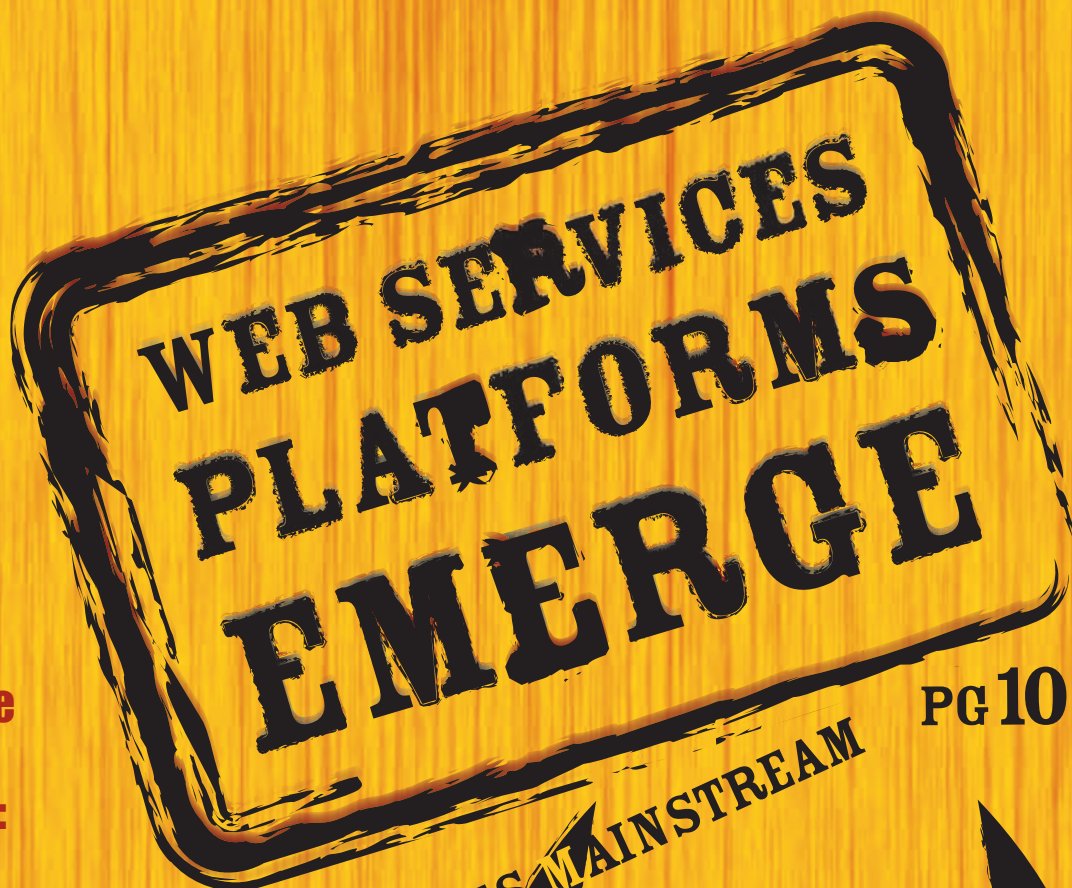
SOA Journey:  
From Web services  
to grid computing  
pg.20

## Why WSDM Matters

The role of WSDM  
in distributed  
IT management  
pg.26

## Managing Enterprise Data Complexity Using Web Services: Part 2

Developing enterprise  
digital dashboards using  
data services architecture  
pg.32



SOA GOES MAINSTREAM PG 10

