

SOA

Web Services

.NET J2EE XML

JOURNAL

October 2005 Volume 5 Issue 10

Best Practices and Solutions for Managing Versioning of Web Services

Patterns and developer
techniques for service
versioning
pg.10

Web Services Assurance for Insurance

Emerging test
automation solutions
are key to a successful
Web services strategy
pg.20

How H&R Block Has Maximized Its Return on SOA

Counting on management
infrastructure
pg.28

DOES YOUR SOA

ACHIEVE AGILITY?

**SOA actualization:
enterprise agility**

PG. 32

RETAILERS PLEASE DISPLAY
UNTIL DECEMBER 31, 2005

\$6.99US \$7.99CAN

10>



71486 03420 9



Ensure Interoperability.

Validate functionality.

Eliminate security vulnerabilities.

Test performance & scalability.

Confirm compliance.

Collaborate thru reuse.

Ensure Secure, Reliable Compliant Web Services

 **PARASOFT.**

SOA Test™

As enterprises adopt Service Oriented Architectures (SOA) to deliver business critical data, ensuring the functionality and performance of Web services becomes crucial. Complex web services implementations require the means to thoroughly validate and test them to assure they are truly production ready.

Parasoft SOA Test is a comprehensive, automated tool suite for testing web services and complex Service Oriented Architecture (SOA) solutions to ensure they meet the reliability, security and performance demands of your business. SOA Test provides a total and holistic testing strategy for your SOA implementations including automated unit testing, graphical scenario testing, scriptless performance/load testing, security penetration testing, standards validation, message authentication, and more.

If you are building serious web services, you need SOA Test.

Download a copy of SOA Test for a free evaluation today at www.parasoft.com/SOA_WSJ

Parasoft SOA Test clients include: Yahoo!, Sabre Holdings, Lexis Nexis, IBM, Cisco & more.

 **PARASOFT®**

We make software work.™

Automated Software Error Prevention™

Parasoft Corporation, 101 E. Huntington Dr., Monrovia, CA 91016. For information, call 888-305-0041 (select option "1"). Copyright ©2005 Parasoft Corporation. All rights reserved. All Parasoft product names are trademarks or registered trademarks of Parasoft Corporation in the United States and other countries. All other marks are the property of their respective owners.



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 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

Does Your SOA Achieve Agility?

32

SOA actualization: enterprise agility

By Jeff Schneider

Best Practices and Solutions for Managing Versioning of Web Services

10

Patterns and developer techniques for service versioning

By Sriram Anand et al.

Web Services Assurance for Insurance

20

Emerging test automation solutions are key to a successful Web services strategy

By Linda Hayes

How H&R Block Has Maximized Its Return on SOA

28

Counting on management infrastructure

By Fred Carter

From the Editor Death to the Browser

Sean Rhody 7

Industry Commentary SOA Makes for a Strange Bedfellow

Ajit Sagar 8

WSJ: Product Review BPM Suite from Bluespring Software

Strong Web services capabilities and an intuitive user environment

Brian Barbash 26

WSJ: Industry The Momentary Enterprise

Thinking outside the VC box

John Webster 36

WSJ: SOA Does Your SOA Include a Persistence Strategy?

You need to consider the nonvolatile along with the volatile

David Linthicum 40

WSJ: Architecture Contract-First Web Services

Six reasons to start with WSDL and Schema

Steve Close 42

WSJ: Integration Leveraging Your Host Systems with Web Services

Transforming legacy applications into SOA-based composite applications

Farshid Ketabchi 46



Transforming Large XML Documents

INDRONIEL DEB ROY

52

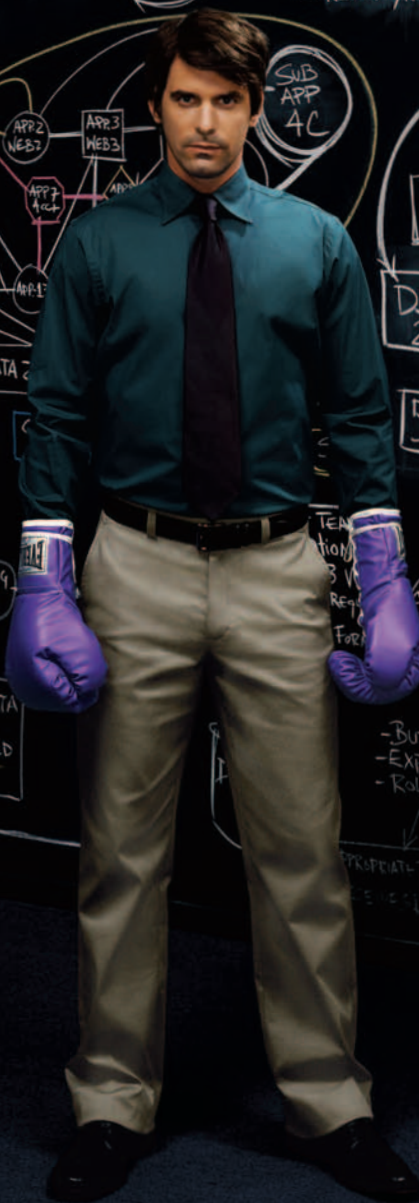
Preventing XML Problems

DR. ADAM KOLAWA

56

WebSphere

IBM



IBM WEBSHERE PRESENTS

YOU vs REWRITE, REVISE RE-EVERYTHING

INFLEXIBILITY: MEET SOA

**PLUS: TANGIBLE BUSINESS BENEFITS ★ BEST PRACTICES FOR BEST RESULTS
OVER 10 YEARS OF WORLD-CLASS INTEGRATION EXPERTISE**

FEATURING:

**A FASTER, EASIER WAY TO
IMPLEMENT TRUE SOA**

**SUPPORT FOR OVER
80 O.S. CONFIGURATIONS**

**UNPARALLELED INDUSTRY
KNOWLEDGE & PROCESS SKILL**

IBM MIDDLEWARE. POWERFUL. PROVEN.

FIGHT BACK AT WWW.IBM.COM/MIDDLEWARE/SOA. THIS IS A RIP-AND-REPLACE-FREE EVENT.

IBM, the IBM logo and WebSphere are registered trademarks or trademarks of International Business Machines Corporation in the United States and/or other countries. ©2005 IBM Corporation. All rights reserved.



Drown your competition with Intermedia.NET

Looking for a hosting company with the most cutting-edge technology? With **Intermedia.NET** you get much more than today's hottest web & Exchange hosting tools – you get a decade of experience and an unmatched reputation for customer satisfaction.

Thousands of companies across the globe count on us for reliable, secure hosting solutions...and so can you.

In celebration of 10 successful years in the business, we're offering a promotional plan for just \$1. Visit our website www.intermedia.net to find out more.

Our premier hosting services include:

- Windows 2003 with ASP.NET
- ColdFusion MX with Security sandboxes
- Linux with MySQL databases
- E-Commerce with Miva Merchant Store Builder
- Beneficial Reseller Programs
- ...and much more!

Unprecedented power, unmatched reputation...
Intermedia.NET is your hosting solution.



INTERMEDIA.NET

Call us at: **1.888.379.7729**

e-mail us at: **sales@intermedia.NET**

Visit us at: **www.intermedia.NET**

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

EXECUTIVE 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

Sriram Anand, Brian Barbash, Fred Carter, Mohit Chawla,
Steve Close, Linda Hayes, Farshid Ketabchi, Adam Kolawa,
Krishnendu Kunti, David Linthicum, Akhil Marwah, Sean Rhody,
Indronil Deb Roy, Ajit Sagar, Jeff Schneider, John Webster

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.



Death to the Browser

To paraphrase, "I come not to praise the Browser, but to bury it." Because the cold hard fact of application development is that the browser needs to die. Immediately. It's already caused more than enough damage.

This may seem to be a harsh statement. After all, the browser was responsible for the explosion of the Internet. It serves many useful purposes and people do billions of dollars worth of business through it every year. Seemingly, I should be praising the browser, not calling for its execution.

Nevertheless, the browser needs to go, and we all know it. It's the dirty secret of the IT world, one we never like to talk about – as a mechanism for delivering a GUI, the browser stinks.

Stinks isn't even a strong enough word. The browser was intended to deliver text across the Internet, and it's good at that. So good that people began to piggyback other things onto their HTML code in order to try to exploit a mechanism of enormous popularity to deliver applications. That's where the problems began.

In one sense, it is HTML and HTTP themselves that have let us down. They stopped evolving, stopped trying to grow – and have been coasting, resting on their laurels for years. By now HTML should have evolved a cross-platform mechanism for designing rich controls and multiwindow applications. It should have moved beyond request-response and standardized a bidirectional communication mechanism so that only data need be transmitted. The overwhelming popularity of software such as Instant Messenger and Napster prove that bidirectional communication is possible, and very desirable. Instead, we have frames and a refresh tag.

I've gone on record before regarding the last mile of Web services and SOA – namely the delivery of complex services and user interfaces to end users. This is where HTML should be – it should have evolved as a mechanism to allow us not to just post text content, but to describe application function as it relates to presentation.



WRITTEN BY
SEAN RHODY

Admittedly, this is a complex area, one where others have tried and failed or at best partially succeeded in driving a common understanding. Nevertheless, rather than writing application code in the form of applets or ActiveX controls, would it not be easier to describe behaviors in XML and allow the next incarnation of the Browser to render application displays? If the capability existed, the tools to make application design feasible and simple would soon follow.

Instead, the browser is brain dead. Plug-ins and controls don't help, because for the most part, even though they may be high quality, they are provided by a single vendor and don't have the force and impact of an industry standard. Also, it's too much work to make the browser look like an application, and in the end, you still have to write the entire application in a way that gives developers fits – because of the constraints of the browser.

What is needed is the Post Browser, the Next Browser, whatever name you want to give to it. Sure, it can still run HTML (the old stuff), in a container that is essentially the same as today's browser. However it should be capable of complete look-and-feel customization via a standard markup language. It should provide a rich set of custom controls and be able to access the desktop (with appropriate security, of course). It should have a native, secure, bi-directional mechanism, and one that supports multiple connections so that we can access services from multiple sources in a composite application. It should also have extensible controls so that we can extend and improve the behavior of controls and applications as needed. Furthermore those extensions should become part of the next release of the standard, which shouldn't take years to come forward.

So I say "Death to the Browser" – bring on a real application platform. ©

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

Kerry Mealia kerry@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

Vincent Santaiti vincent@sys-con.com

Shawn Slaney shawn@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@edithrom.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.



SOA Makes for a Strange Bedfellow

Over the last few years, Web services and SOA have made a lot of inroads into not only the IT departments of large enterprises, but also into the minds of the business owners of different LOBs (Lines of Business). SOA is more than Web services; it is the mantra for bridging the gaps and walls between IT and business. One of the main reasons behind this is that when the SOA paradigm is mentioned, it automatically translates into an "interoperable," "technology-neutral" nirvana. Of course, any mature organization accepts the fact that there are many baby steps before getting from here to there, but the promise of success is believable—at least in theory. Nowadays you can hear the words "component factory" and "business factory" in the same sentence.

While organizations are embracing the promise of SOA, vendors are presenting a united front by touting their ability to expose whatever plumbing they have behind the covers as ubiquitous services that are available to any client. When Web services started gathering some momentum, one of the main concerns from the industry was the ability of product vendors to be able to work together to make the promise a reality. CORBA had come and gone (after all, in many ways, SOA is just another euphemism for distributed computing). In order for the whole picture to come together, the mainstream platforms, namely .NET and Java, had to adhere to the standards and deliver products that would pragmatically facilitate service enablement. The industry has come a long way, and it does seem that vendors are providing the united front that is needed, whatever their product offerings may be. Their focus is now more on how to differentiate via the tools and standard APIs that will enable the components to become "SOA compliant."

As the industry moves forward, there is more cross-platform support between vendors. Case in point, the recent agreement between JBoss and Microsoft that will have increased benefits for JBoss users who are developing Web services on .NET platforms. Five years or so ago, you wouldn't have thought of these types of partnerships. JBoss is very much a prodigal child of the Java platform. The technology engagement between the two companies is expected to include technical assistance and architectural guidance on the following features:

- **Microsoft Active Directory:** Integrated sign-on and federated identity



WRITTEN BY
AJIT SAGAR

- **Web Services:** Interoperability using Web services architecture
- **Management:** A Management Pack for Microsoft Operations Manager
- **SQL Server:** Optimized performance for users of Hibernate, JBoss's object/relational mapping technology, and Enterprise JavaBeans 3.0

JBoss is in an interesting space. At JavaOne this year, I had the chance to interview Pierre Fricke, director of product development at JBoss (the interview is available online on SYS-CON.TV – <http://java.sys-con.com/read/109191.htm>), and he mentioned that JBoss is building a professional services unit that will assist customers in deploying JBoss products. It will be interesting to see how fast the company can scale to address the needs of the customers not only in the product deployment phase, but also in planning how to take advantage of the integrated offerings for Web service enablement. I think that level of scalability is only achievable through partnerships with System Integrator companies.

Last month Microsoft released Web Services Enhancement (WSE) 3.0 Beta for Windows .NET, which enhances the development and deployment of secure Web services. As far as tools go, Microsoft has always maintained the lead in tools that automate development and deployment. The Web services arena has been no exception. The challenge for .NET has been, and will continue to be, acceptance for enterprise applications in large organizations. Web services definitely constitute a paradigm that facilitates the adoption of .NET into large organizations because the interoperability provided by the paradigm enables IT to adopt .NET into parts of the solution while retaining other technologies for the high-volume applications. Partnerships such as the recent one between JBoss and Microsoft will help to further the cause. ☺

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



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

© 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.

Best Practices and Solutions for Managing Versioning of Web Services

Patterns and developer techniques for service versioning

■ Service-oriented architecture (SOA) and Web services are being critically considered by most organizations today in some form or another. The adoption of SOA and Web services has gained momentum after the standardization of various aspects such as security, business process coordination, transaction management, communication protocol, registration and discovery, etc. However, one notable and practical aspect of designing, implementing, and managing services has not been tackled at a specification level. This aspect is related to the management of change and interface versions.

Real-world business and infrastructure services will be dynamic and subject to change due to various issues that may range from a business rules change to service-level

improvement. It is a challenging problem to manage services whose interfaces are changing, especially in the situation where the number of consumers for the service is significant. In this article we present sample scenarios explaining where this problem can occur and we'll discuss detailed options with code samples to manage service versions and service changes. The options provided range from design-level techniques to specific implementation-level

techniques based on emerging standards such as WS-Addressing.

Keywords

SOA, Web services, version control, standards, change management

Problem Background

A service represents a unit of functionality that is described by well-defined interfaces and can be discovered by interested consumers and invoked remotely without any knowledge of the technology or location of the components that realize the specific service. SOA represents an enterprise architecture that is based on loosely coupled services that communicate with each other through well-defined interfaces through a communication medium. Web services represent the bulk of SOA implementations in the industry today. The issue of version management is an important one in the SOA context independent of the mechanism of service implementation, i.e., Web services or another approach. Therefore, let's examine some possible scenarios where this problem will be relevant.

Consider the situation in Figure 1 where we have components that perform compliance checking implemented using Web services. The business context is one that represents a typical financial services organization.

WRITTEN BY



**SRIRAM
ANAND**



**KRISHNENDU
KUNTI**



**MOHIT
CHAWLA**



**AKHIL
MARWAH**



XML 2005

The reliable source for everything XML.

XML 2005 is the largest independent conference for professionals who want to use XML and related technologies to **produce higher-value information, increase production speed, and save money.**

Experience the **entire universe of XML-based technologies**, including web services, publishing, data integration, information management, and evolving applications.

- Nearly two-dozen tutorials on everything from XML fundamentals to applications development to web services
- 120 technical sessions presented by the top technical minds, technology implementers, and industry insiders
- Keynotes from industry, academic and consultancy experts worldwide
- Exhibition of products and services from leading vendors

Register now for discounts!

XML 2005 • November 14-18, 2005 • Hilton Atlanta, Atlanta, Ga., USA

<http://2005.xmlconference.org>
registrar@idealliance.org

tel: +1 703 837 1070 fax: +1 703 837 1072

Produced by:



Sponsored by:



Media sponsor:



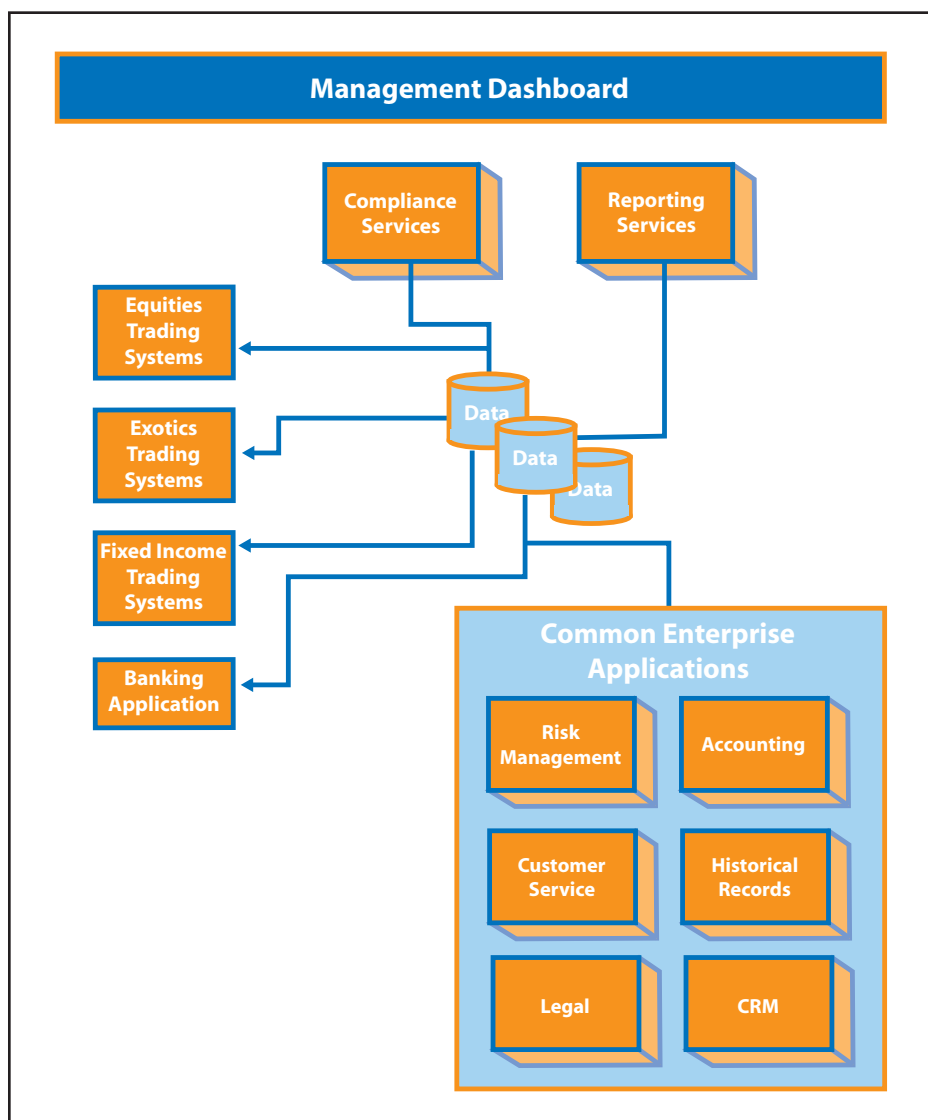


FIGURE 1 Financial services systems context

In this context, we have multiple front-, middle-, and back-office applications that are pertinent to the following lines of business: equities trading, fixed income trading, and banking. Data from these applications flow to a set of databases associated with each business application.

Back-office applications such as risk management, CRM, accounting, etc., have a critical dependency on the data from the business fulfillment applications. Compliance is a critical requirement of all financial organizations. Some of the areas in which financial services companies have to enforce compliance rules are: Sarbanes-Oxley, USA

Patriot Act, and BASEL II. IT applications that implement the rules for these compliance requirements can be ideal candidates to be Web services. Implementing compliance through the use of services will allow organizations to be flexible in adapting to changes required for regulatory compliance. Apart from this benefit, on the technical side it will provide a standard way for core financial applications that are written using multiple technologies to use the compliance services (see the first entry in the References section). This briefly establishes the background for using SOA in a compliance setting. A similar rationale may be developed for reporting applications

that will provide rolled-up information to management. We will not delve into the details of the applicability of SOA for these two situations in this article. We will assume that Web services have been chosen to implement these applications. Consider a situation with regards to the service for performing compliance checking.

Assume that a portType exists for performing compliance checking with respect to the USA Patriot Act. For the sake of simplicity, let's assume that this is a coarse-grained interface that is responsible for verifying the compliance of the data associated with a specific customer. Consider an interface named "checkCustomerCompliance" that accepts a customer ID as input, retrieves all appropriate data related to that customer, performs the necessary compliance checks, and returns a status back to the consumer. Given this WSDL and considering potential mechanisms by which this Web service may be changed, we can come up with the following possibilities:

- Correcting a service implementation, e.g., a bug fix that does not involve any change to the service contract.
- Interface modification due to addition or deletion of parameters to support an implementation change or through service enhancement. Fundamentally, this does not involve a change to the semantic contract of the old interface.
- Service implementation change (with no change to service interface) due to change in the business rules. A specific example would be the modification of a service used for compliance checking based on a change to the compliance rules/laws.

These scenarios are discussed in more detail below with specific solution options to handle each type of change.

Options for Handling Service Changes

Implementation Change

As mentioned above, this type of change is strictly related to the rectification of errors in a service implementation that does not include a change to the external service interface. Therefore, in principle, consumers of this service would be completely unaware

of any change to the service whatsoever. This type of change is fairly easy to manage given the fact that the consumer, in principle, is getting a service that is closer to the contract that has been specified. The options to manage this type of a change are therefore quite simple and can be detailed as below.

Silent

One fairly trivial option that can be adopted is to simply deploy the corrected service implementation and decommission the original service implementation. The modified implementation would then be bound to the original service-interface definition. Obviously there are other considerations that accompany this, such as performing the deployment without impacting users who are currently accessing the service during the time of this deployment, but we will not address those issues in this article.

Version Number

The second and more involved approach to handle this situation is using a change identifier. This approach may be realized by using the identifier in the service interface headers. This identifier can provide a pre-defined measure of the change in a specific interface. By predefined we mean that the service documentation will enumerate the different values that the identifier can take. Consumers of the service should be capable of obtaining the interface change details. The identifier can be valid for a specific period of time after which its value will be reset, thus completing the process of notification

Option	Pro	Con
Silent	Minimum additional work to be performed	May violate SLA in terms of keeping consumers informed of changes
Version Number	Changes are communicated to service consumers	Overhead of an additional output parameter
Modified Implementation	Allows multiple versions of the implementation to be running at the same time	<ul style="list-style-type: none"> • Overhead of maintaining multiple versions • Overhead on the part of consumers to interpret the additional variable

TABLE 1 Summary of options for managing change in a Web service

of consumers. To provide a specific example, consider a service named “getUSSales” that is responsible, as the name indicates, to provide a rolled-up report of all US sales by polling the appropriate databases. Assume that the query that was used in the service implementation to query one of the databases was wrong and was subsequently corrected. As a part of the service definition, a parameter may be defined as follows:

```
<message name="USSalesResponse">
    <part name="change_detail"
type="xsd:string"/>
...other parameters...
</message>
```

In this example, the parameter “change_detail” may be a concatenation of two numbers, one indicating whether the service has changed or not, and the other indicating the change reference number (like a bug-tracking number for example). The WSDL and SOAP (generated from apache AXIS) shown in Listings 1 and 2 contain a part-in-message

element to indicate version number and the response generated from such a service.

Modified Implementation Version

The third option that is available is to maintain the old interface and define a new implementation. The interface can be bound to the modified implementation as well and the version information may be provided via the SOAP headers. It would become the responsibility of the calling applications to interpret the version number. The calling application needs to include a header element in the SOAP request that indicates the required version number of service implementation. In case the requested implementation is not the most recent one, the service includes a SOAP header element that indicates the most recent version available.

This method will require multiple implementations of the interface to be active at the same time. The behavior of the service that depends on the header can be described in human-readable format using the documentation feature in WSDL. The polymorphic behavior can be achieved either by having more than one implementation of the interface or by implementing a new service and diverting the requests based on SOAP headers.

The first option method, exposed as a Web service internally, uses a factory method to get multiple implementations of an interface, and hence a polymorphic behavior. An implementation is selected based on the SOAP headers.

The second option can be implemented by having Web service implementations of both old and new services. The Web service consumer can continue to direct requests to the old service and the requests may be diverted to the new service by using the

“ It is a challenging problem to manage services whose interfaces are changing, especially in the situation where the number of consumers for the service is significant

”

service-level handler based on SOAP headers (see Listings 3 and 4). The multiple options are summarized in Table 1.

Recommended Option

Based on our experience, we recommend the option with the version number that will provide the information about the change to the service consumers. The service can specify a period beyond which the change identifier will not have a relevant value so that this output parameter may be used continuously for multiple changes. This option is recommended because of the level of additional work that is required balanced against

vice such as the aforementioned “checkCustomerCompliance” service. In the event of a legislative change in compliance rules, this service would require modification that may precipitate changes to both interface as well as to implementation. As a specific example, assume that the aforementioned service is responsible for retrieving a customer data and then running specific checks based on the regulations under the Patriot Act entitled “Money Laundering.” Assume that at a later point in time, the regulations are modified in some way that requires the compliance checking to be provided with additional information. This situation would result in

that we are considering a situation with the WSDL snippet (sample WSDL for sales) provided below.

```
<message name="checkCorporateSalesResponse">
  <part name="line_of_business_id" type="xsd:string"/>
  ...other parameters...
</message>
```

Let's assume that due to reorganization within the company, this service is responsible for only returning the sales associated with a particular geographic region. One option to align the service with the requirement change is to add a parameter that specifies the geographical unit for which the sales are required. However, this may need to be implemented carefully because existing consumers of the service would get incorrect information if the implementation were performed in a transparent manner. The existing interface will need to be retired to prevent consumers from getting outdated functionality. This may be accomplished by throwing a fault on the old interface that redirects the user to the updated WSDL. In addition, the change identifier concept mentioned above can be used with a special value that indicates a “retired” interface. The service implementation can return the change identifier for a specific period of time after which the service, including both interface and implementation, may be decommissioned.

Stateful Web Services

Consider the situation of a service that is stateful and that is implemented in a manner that the service implementation is aware of the consumer. This type of implementation is rare and this option has been discussed here for the sake of completeness. For this type of a service there is the option to track information provided to a service consumer. Therefore, an approach similar to the change-identifier method may be used along with the implementation of a new interface. In this method, the service output may be modified to add an element that contains information on the change. The consumer-calling interface is also augmented with an element that

“ The service implementation can return the change identifier for a specific period of time after which the service, including both interface and implementation, may be decommissioned ”

the need to communicate relevant changes to service consumers.

Interface Modification

Now let's look at situations that may warrant a change in the actual service that must be communicated to the consumer, with the possibility of a change in the WSDL (specifically in the “portType,” “message,” and “binding” elements of a typical WSDL). The important characteristic of this type of modification is that the change needs to be communicated to all of the consumers of the question without fail in order to ensure compliance with the SLA. This change may arise due to one of a few different reasons such as correcting an implementation as mentioned above or making a planned enhancement to the service based on requests from consumers simply to support modified business rules. Consider a compliance ser-

vice change to the service as discussed above. There are several approaches to handle a service interface that is subject to this type of a change, and these options are discussed below. These options may be broadly classified into two groups. One set of options uses version management techniques without requiring any additional technology implementations. The other set has a dependency on UDDI and the implementation of a UDDI directory. The options are discussed in additional detail below.

New Interface

The first and most obvious option that comes to mind is the creation of a brand new interface, regardless of whether the old interface may be reused or not. Note that under certain circumstances the old interface may be reused, especially if existing parameters are being deprecated. For example, assume

provides an acknowledgement to the service implementation about the status of the message received. This mechanism can be used as a programmatic method of communication between a service producer and service consumer on issues that are extraneous to the actual functional implementation. Once a service consumer acknowledges the receipt of a change, the data may be used to track the number of consumers who have been made aware of a specific interface change, and subsequently the interface may be retired. Once a specific consumer is informed of a change, no additional information about this specific change needs to be sent.

XML Namespace

The concept of XML namespaces may be used to provide a different mechanism for handling interface change. Recall that the “targetNamespace” attribute of a schema is used to provide a scoping mechanism in XML; therefore, this attribute may be used to identify changed interfaces. The convention used for naming the namespace may be predefined and communicated to consumers through the service documentation. One simple approach may be to add a version number to the original namespace URL. Alternatively, a timestamp may be used to augment the URL. Take a look at the WSDL snippet shown in Listing 5. Consider a change to this WSDL similar to the aforementioned change to add the geography as an additional filter for getting the sales pertinent to specific locations. In this case, we may indicate the modification by changing the namespace attribute to something as follows:

```
targetNamespace=http://www.mydo-  
main.com/webservices/salesreport/  
CorporateSales_v1.1.wsdl
```

or

```
targetNamespace=http://www.mydomain.  
com/webservices/2005/01/05/salesre-  
port/CorporateSales.wsdl
```

In the second example, the versioning is accomplished by placing the WSDL scoped by the timestamp of the change. Requests associated with the previous version of the

WSDL, i.e., those that are associated with the older namespace, may be handled by a couple of different options. One straightforward option is to generate an error on the server and provide data to the user on the new implementation. This approach would result in service consumers having at least one guaranteed failure per service change during service invocation. Therefore, if a service undergoes multiple modifications within a short time frame, this can lead to poor quality of service delivered to the consumer. This scenario is quite realistic in the case of service implementations for common functionality, e.g., data access in a large company. Typically these services are designed to accommodate all consumers, but evolving needs of multiple areas of the company may require frequent modification until the services are “stabilized” and adopted throughout the company. In this situation, a more graceful approach may be to provide an intermediate Web service that can accept requests and then make a decision to forward the request to the correct implementation based on the timestamp rules. This would not add a significant amount of overhead to the existing implementation and it may be the preferred option given the expected frequency of service changes.

Complex Type (SOAP Response)

An additional option for communicating a specific change to a service consumer is the use of a SOAP response element. In such a scenario some predetermined tags in the SOAP response are used to convey the information. The SOAP response should be some complex type that contains version information along with response parameters. The analog to a complex type can be a class in the Java programming language, for example. This type can be composed of any constituent elements. Therefore, we could conceivably have an element that is dedicated to storing the version of a particular interface. Consumers of the service may retrieve the data in the complex type and interpret it for detailed information on the characteristics and “age” of the particular service. This element is returned instead of an actual response in case the version of a

service has changed. This approach allows for significant flexibility in communicating changes to a service consumer. Listing 6 shows a snippet from a WSDL to help illustrate this better.

Here we note that a complex type of name “VersionDetails” has been created for communicating the service version information to service consumers. The ID will have a predefined format to indicate the service version. Note that adopting this approach would only serve to provide the basis for transferring information to the service consumers. A policy needs to be adopted for managing the implementations of the previous versions of the interface. One approach is to combine this method with the method of identifying versions by namespace. The version ID of the most current interface may be provided to the consumer. Subsequently, the consumer may use this version to use the correct namespace and invoke the most up-to-date WSDL.

UDDI-Based Options Specifically for Web Services

The options discussed above primarily deal with version management in the situation where we are dealing with services that are accessed by consumers directly. This can be done either by providing endpoints to interested consumers or through some manual mechanism that involves a priori knowledge of a particular service endpoint.

The recommended best practice for Web services is to use UDDI directories (see the fourth entry in the References section). This will allow the dynamic lookup of service interfaces and implementations that are tied to specific business areas or other criteria that the service consumer can specify. Specifications for UDDI are maturing and there are some sample UDDI directories available on the Internet, for example, the public directories from IBM or Microsoft. However adoption of UDDI has not come anywhere close to the rates of adoption of Web services. Below we discuss some options for versioning services using UDDI, along with the merits and demerits of those options (see the second entry in the References section).

The UDDI data structure that is used for

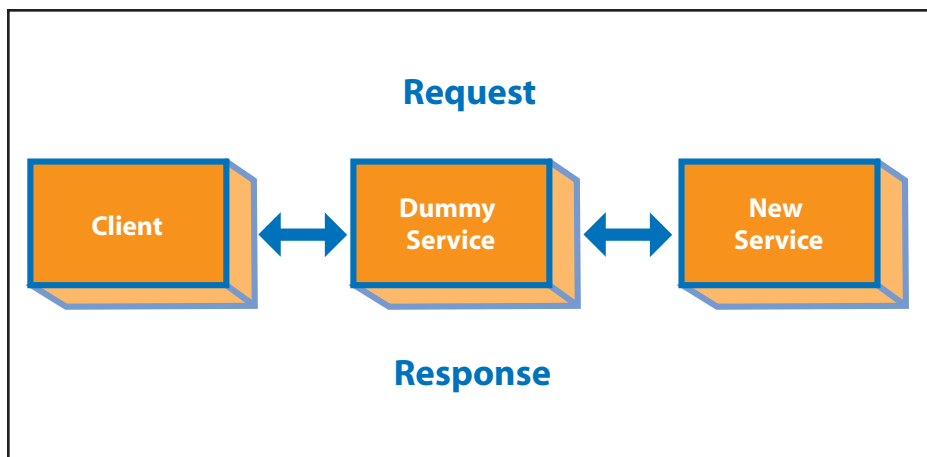


FIGURE 2 Using the traditional approach to divert requests

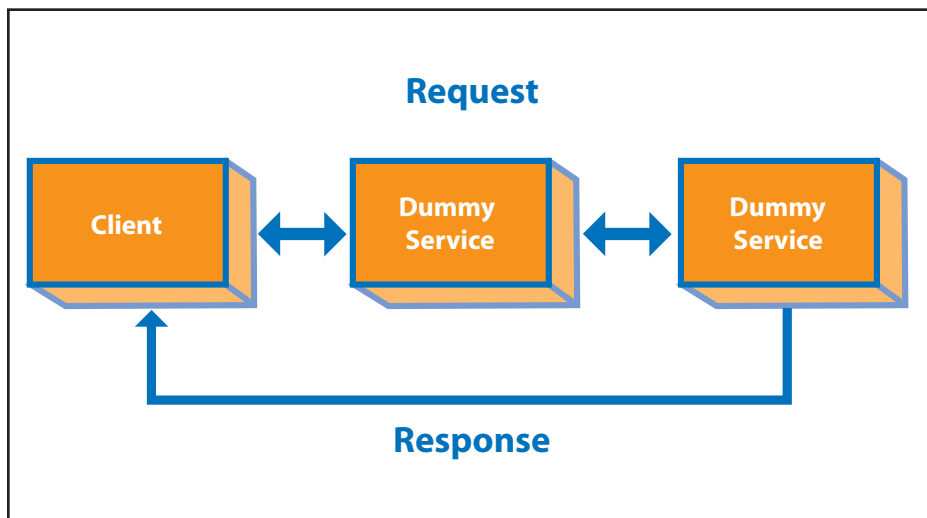


FIGURE 3 Using WS-Addressing for diverting requests

publishing information on organizations and the services they provide is called a tModel.

A tModel represents general concepts that many organizations can refer to in their UDDI registrations. A tModel is identified by a unique key known as the tModelKey. Apart from this, a tModel data structure may contain other elements that are discussed below:

- **authorizedName:** Represents the name of the person publishing the tModel
- **operator:** Name of the registry operator
- **name:** Name of the tModel
- **description:** Description of the tModel
- **overviewDoc:** References to remote descriptions associated with the tModel, e.g., the URL for a WSDL document residing on a different server

- **identifierBag:** Name-value pairs to hold identification criteria
- **categoryBag:** Name-value pairs that are used to associate the tModel with a specific taxonomy, e.g., an industry, line of business, or geography

Note that the tModel's overviewDoc field is used to reference a WSDL that conforms to the tModel in question.

A UDDI registry would have a number of tModels available depending on the type of service/business. This can be queried by Web service consumers. The programmatic query would return a collection of tModelInstanceDetails that represents all tModels that match the criteria provided (see the first entry in the References

section). Let's examine how we might leverage tModels to support version management.

Option 1

We can create a different tModel for every available service version. This would generate a unique tModelKey for every possible "flavor" associated with a specific Web service. Considering that the industry taxonomy classification would be the same, all these service versions would be retrieved from a UDDI registry. Using this approach, consumers could select the WSDL that is associated with the version that they need.

Option 2

A tModel is associated with a data structure called tModelInstanceInfo that contains some fields pertinent to a specific tModel. This data structure has a field named "instanceDetails" that may be used to store a version number for a specific Web service implementation. However, using this approach for version management does not constitute a best practice and defeats the association between a WSDL and a tModel. UDDI may also be used to deprecate obsolete versions of a service by providing appropriate details in the "identifierBag" field mentioned above.

However, all the methods identified above are by no means very elegant. The first option is reasonably stable, but suffers from a bloat in the number of tModels for dynamic services that have multiple versions. The other approaches represent workarounds at best that attempt to compensate for the gap in the standards.

WS-Addressing for Version Management

The WS-Addressing specification (see the fifth entry in the References section) can be used to provide an efficient solution for tackling issues related to versioning in Web services. As mentioned in the problem background section, three broad scenarios have been identified in relation to version management of services, namely implementation change, implementation correction, and interface change. WS-Addressing can also be used to address scenarios in which the endpoint URL of a Web service changes.

The challenge faced in all three of these scenarios is to make the underlying users of the services aware of the changes made in the service. WS-Addressing can be used to provide an optimized and efficient solution to address the aforementioned issue. In the following section we'll discuss each individual scenario.

Implementation Correction or Implementation Change

In this scenario the implementation or the functionality of the service is changed or corrected. In such a case the client will continue using the service without any knowledge that the service has changed. One of the options is to include a parameter in the method invocation signature (Reference section: option for handling service changes) that denotes the version, but the stated option couples version information with business functionality, which is not an elegant solution. WS-addressing can be used to solve the issue by adding a custom tag to indicate any change in implementation, for instance in Listing 7 a tag named <implementation> has been included in a response message to indicate the change in the implementation logic. Whenever the implementation changes the information pertaining to the specific change is included in this tag for a certain period of time. When there has been no change performed to the implementation of the service, an empty implementation tag is provided. Similarly, the client can include an implementation tag that indicates the last-accessed implementation version, so if the implementation has changed since the last client request the service can send the required information.

Note: It needs to be pointed out that the implementation is a custom-defined tag and hence the usage of this tag must be done only after an agreement between the client and the service.

Interface Change

In this scenario, recall that the interface, i.e., the number of parameters or the type of parameters associated with the interface, changes. In this case an error message needs to be communicated to the client that should also contain information communicating the details of the change. In case a client request

does not match the required interface definition, WS-Addressing can be used to tackle the change by including some custom header tags that indicate interface change.

For instance, in Listing 8 a custom tag called <interface> is used to communicate the information related to the change to the client.

Endpoint Change

In this scenario the endpoint of a service is changed and the responsibility of forwarding the request to the new endpoint is delegated to some dummy service for a period of time. In such a scenario the requirement is to communicate the new URL to the existing users.

Figures 2 and 3 show such a scenario. In a traditional approach the dummy service takes the responsibility of invoking the new service and sending the response back to the client. There is no way the new service can send the response directly to the client. Moreover, custom headers need to be added in order to communicate the changed location to the client. Use of WS-Addressing headers can solve the aforementioned issues, and the <from> tag in the response effectively communicates the new URL to the client. Furthermore, the service can send the response directly to the client based on the <ReplyTo> tag of the request message.

Summary

We have examined some options for the version control and management of change in Web service implementations. We have examined options that utilize design patterns, best practices, and that also leverage existing technologies such as UDDI. It is perceived that emerging standards in the Web services space such as WS-Addressing may provide some standardized mechanisms for handling service versioning. Until such standards are in place and adopted by popular vendors, some of the options discussed above may need to be used for large-scale service deployments. The emergence and maturity of Web service management platforms may provide some robust functionality in this regard as well. Until the convergence of standards and the stable functionality provided by these products, Web service implementers may be forced to rely on unorthodox mechanisms for change management.

References

- Kenyon, J. (2003). "Web Service Versioning and Deprecation." *Web Services Journal*. SYS-CON Publications, Inc. February.
- Brown, K., and Ellis, M. (2004). "Best Practices for Web Services versioning." *IBM Developer Works*. January.
- *UDDI Specification*: www.uddi.org/specification.html
- *WSDL Specification*: www.w3.org/TR/wsdl
- *WS-Addressing*: www.w3.org/Submission/ws-addressing/ ©

About the Authors

Dr. Sriram Anand is a principal researcher at Infosys Technologies, Bangalore. Prior to joining Infosys he worked in IT consulting as well as product engineering in the US for over 12 years. His interests include enterprise architecture, service-oriented architecture, and legacy integration and software engineering methodologies. Dr. Sriram is experienced in designing enterprise architectural strategy for leading U.S. companies in the financial services, retail, and pharmaceutical domains. He holds a Bachelor's degree from IIT-Madras with a PhD from SUNY-Buffalo, USA.

■ ■ ■ sriram_anand@infosys.com

Krishnendu Kunti is a senior technical specialist with the Web Services Center of Excellence at Infosys Technologies, Hyderabad. He has contributed to architecting, design, and testing of Syndeo, an in-house service-oriented platform for enterprise-grade deployment of Web services. His areas of interest are SOA and business execution languages. Currently he is working as a technical lead at a leading financial services company developing data services using SOA.

■ ■ ■ krishnendu_kunti@infosys.com

Mohit Chawla is a software engineer with the Web Services Center of Excellence at Infosys Technologies, Hyderabad. His primary area of interest is SOA, with a specific focus on Web services implementations on various platforms. He is also interested in developing applications using emerging WS-* standards. His current is currently focused on SOA-based enablement of legacy systems.

■ ■ ■ mohit_chawla@infosys.com

Akhil Marwah is a software engineer with the Web Services Center of Excellence at Infosys Technologies, Hyderabad. His current area of focus is Web service implementations using Microsoft Technologies, specifically the MS .NET framework. He is also interested in developing Web service applications using the J2EE stack. He is currently working on areas related to SOA-based enablement of legacy systems.

■ ■ ■ akhil_marwah@infosys.com

Listing 1: Part in response message to indicate version number (tool used - Apache AXIS)

```
<wsdl:definitions targetNamespace="http://localhost:8080/axis/services/SalesProcessor">
  <wsdl:types>
    <schema targetNamespace="urn:SalesService">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <complexType name="Sales_Version">
        <sequence>
          <element name="customerName" nillable="true" type="xsd:string"/>
          <element name="ussales" type="xsd:int"/>
          <element name="versionChange" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
    </schema>
  </wsdl:types>
  <wsdl:message name="getUSSalesResponse">
    <wsdl:part name="getUSSalesReturn" type="tns:Sales_Version"/>
  </wsdl:message>
```

Listing 2: SOAP response generated from the aforementioned service (tool used - Apache Axis)

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:getUSSalesResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="SalesProcessor">
      <ns1:getUSSalesReturn href="#id0"/>
    </ns1:getUSSalesResponse>
    <multiRef id="id0" soapenc:root="0" soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:type="ns2:Sales_Version" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns2="urn:SalesService">
      <customerName xsi:type="xsd:string">krishnendu</customerName>
      <ussales xsi:type="xsd:int">100000</ussales>
      <versionChange xsi:type="xsd:string">changed_bugzilla12</versionChange>
    </multiRef>
  </soapenv:Body>
</soapenv:Envelope>
```

Listing 3: SOAP message generated by the client to indicate the version of the service required (tool used - Apache Axis)

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header><ns1:Version xmlns:ns1="http://xml-Client.books/">Version1</ns1:Version> </soapenv:Header>
  <soapenv:Body><ns2:getDetail xmlns:ns2="EnvelopeService">
    <ISBN>0586061991</ISBN></ns2:getDetail> </soapenv:Body>
</soapenv:Envelope>
```

Listing 4: SOAP message generated by the service indicating the latest version available (tool used - Apache Axis)

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/
```

```
XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <ns1:Recent_Version xmlns:ns1="http://xml-Client.books/">Version5</ns1:Recent_Version> </soapenv:Header>
  <soapenv:Body><ns2:getDetail xmlns:ns2="EnvelopeService">
    <BookName>Robots of the dawn</BookName></ns2:getDetail>
  </soapenv:Body>
</soapenv:Envelope>
```

Listing 5: Sample WSDL for retrieving corporate sales

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions name="CheckCorporateSales"
  targetNamespace="http://www.mydomain.com/webservices/salesreport/CorporateSales.wsdl"
  xmlns:tns="http://www.mydomain.com/webservices/salesreport/CorporateSales.wsdl"
  xmlns:soap="http://www.mydomain.com/webservices/salesreport/CorporateSales.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <message name="LineOfBusinessInput">
    <part name="lob_id" type="xsd:integer"/>
  </message>
  <message name="SalesResponse">
    <part name="yearlySales" type="xsd:float"/>
  </message>

  <portType name="CorporateSalesPortType">
    <operation name="GetLobSales">
      <input message="tns:LineOfBusinessInput"/>
      <output message="tns:SalesResponse"/>
    </operation>
  </portType>

  <binding name="CorporateSalesBinding" type="tns:CorporateSalesPortType">
    <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLobSales">
      <soap:operation soapAction="getLobSales"/>
      <input>
        <soap:body use="encoded"
          namespace="www.mydomain.com/salesreport/CorporateSales"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </input>
      <output>
        <soap:body use="encoded"
          namespace="www.mydomain.com/salesreport/CorporateSales"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
    </operation>
  </binding>

  <service name="CorporateSalesService">
    <documentation>
      Returns the Corporate Sales on an annual basis for a specific Line of Business
    </documentation>
    <port name="CorporateSalesPort" binding="tns:CorporateSalesBinding">
      <soap:address location="http://sample_server:8090/dayofweek/CorporateSales"/>
    </port>
  </service>
</definitions>
```


Listing 6: Complex Type used for version control

```
<schema targetNamespace="http://mydomain.com/services/
getCorporateSales"
  xmlns:qc="http://mydomain.com/services/salesCommon">
  <complexType name="VersionDetails">
    <attribute name="version_id" type="int"
      use="required"/>
    <attribute name="version_details" type="string"
      use="required"/>
    <attribute name="version_date" type="string"
      use="required"/>
    <attribute name="version_status" type="string"
      use="required"/>
  </complexType>
</schema>
```

Listing 7: Use of a custom implementation tag in WS-Addressing header to convey implementation change

```
<soap:Envelope xmlns:wsa="http://schemas.xmlsoap.
org/ws/2004/03/addressing" xmlns:wssse="http://docs.
oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/
wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsa:Action>soap.tcp://hydhtc29925:3119/
MyServiceReceive</wsa:Action>
    <wsa:From>
<wsa:Address>http://localhost/AddressingService20/
MyService.ashx</wsa:Address>
    <wsa:ReferenceProperties>
<Implementation>The Implementation has been changed. The
new interest calculation method is P*T*9/100 instead of
P*T*8.2/100 </Implementation>
    </wsa:ReferenceProperties>
    </wsa:From>
    <wsa:MessageID>uuid:cff4c9ec-f257-4790-a890-
86ae7821bb9b</wsa:MessageID>
    <wsa:ReplyTo>
<wsa:Address>soap.tcp://hydhtc29925:3119/
MyServiceReceive</wsa:Address>
    </wsa:ReplyTo>
    <wsa:To>soap.tcp://hydhtc29925:3119/MyServiceReceive</
wsa:To>
  </soap:Header>
  .....
</soap:Envelope>
```

Listing 8: Use of a custom interface tag in a WS-Addressing header to convey interface change

```
<soap:Header>
<wsa:Action>soap.tcp://hydhtc29925:3119/
MyServiceReceive</wsa:Action>
  <wsa:From>
<wsa:Address>http://localhost/AddressingService20/
MyService.ashx</wsa:Address>
    <wsa:ReferenceProperties>
      <Interface>The interface has been changed.
For future usage , please refer to the WSDL file or
enter integer values</Interface >
    </wsa:ReferenceProperties>
    </wsa:From>
  .....
</soap:Header>
```

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

SAVE 50% OFF!

(REGULAR NEWSSTAND PRICE)

Only \$39⁹⁹ ONE YEAR 12 ISSUES

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



The World's Leading i-Technology Publisher

Web Services Assurance for Insurance

Emerging test automation solutions are key to a successful Web services strategy

■ Timely delivery of a quality Web services solution requires functional testing at each layer, throughout the development process. New test automation solutions empower insurance domain experts to verify critical business processes at each phase and layer of delivery.

Web services promise insurance companies the ability to be rapidly responsive to both regulatory requirements and market opportunities by enabling changes to software systems without disrupting internal integration among applications or external integration with brokers, agents, and partners. Claims have long been filed electronically, but the standards that have emerged from HIPAA (Health Insurance Portability and Accountability Act) have further propelled the shift to electronic exchange of all transactions.

However because of the very power it unleashes – policies underwritten and issued, claims adjudicated and paid, all behind the scenes and screens – it is essential to assure the accuracy of the results, the privacy of the information, and the stability of the infrastructure. Incorrect responses can reduce



WRITTEN BY
LINDA HAYES

revenue or increase losses, security breaches can compromise patient confidentiality, and failure can disrupt operations within and across enterprises.

Unfortunately, the layered architecture that powers Web services challenges traditional testing approaches, which have historically relied on subject matter experts interacting with end user applications. While these applications may invoke Web services under the covers, they provide indirect testing at best, and issues uncovered at the application level may have originated in one of the lower layers. Furthermore, interactions with external providers or consumers are even more difficult to test because they require predictable, repeatable transactions. As a result, most business process assurance verification has been left to the final phase of implementation when defects are the most costly to diagnose and correct.

The good news is that a new breed of test automation solutions is now emerging to specifically target application domain experts, and bring them into the early stages of the test process by providing a level of abstraction that masks the underlying technical complexities of a service-oriented architecture (SOA). These solutions can simulate or isolate each layer of the implementation, both internal and external, and enable functional testing to occur at each level and layer to assure that the final delivery is not only technically sound and delivered in a timely manner, but that it meets all business and regulatory requirements.

These challenges and solutions, described more fully below, are central to keeping pace in the new electronic insurance marketplace while minimizing business risk.

Web Services Implementation

The implementation of Web services though an SOA typically starts from the bottom up, first with the technical infrastructure, including the transport protocol and message encoding, followed by the integration of provider and consumer applications, both internal and external.

Systems architects or developers often have tools available to test these lower layers for technical compliance, but their

deficiency of subject-matter expertise limits their ability to identify the more subtle business rule or logic errors that can potentially be more devastating than the more obvious errors. Additionally, these tools are designed to test the correctness of the data transport and format – in other words the technology infrastructure – but not the transaction content, namely the business application for the data.

Another drawback with infrastructure-specific tools is that they are rarely integrated with higher-level test management, reporting, and analysis capabilities. Like unit test tools for developers, these are typically point solutions that meet a highly specialized technical requirement but do not provide support for the higher-level information management of the testing process.

Because of their technical nature, these tools are not accessible to business experts, and until now, the only way to involve subject matter experts in early-stage testing was to develop even more software that provided primitive user-test interfaces and stubs.

Users were often required to struggle through the editing of highly complex XML message structures to construct or verify the desired data values, and they could then use these testing interfaces and stubs to send or receive the messages from either dummy queues or from actual transports and the applications behind them.

All of this development effort ultimately diverted time and resources from the delivery of the end solution. Subject matter experts spent the majority of their time dealing with technology details rather than applying their knowledge to ensuring that the business problems at hand were being addressed properly. As a result, functional testing of Web services was typically left

“ Only through extensive, automated test execution can companies truly keep pace with market demands, regulatory requirements, and competitive imperatives ”

until the final stage of integration with the end user applications.

However, waiting until the final end-user layer is in place to test business functionality delays the time to market by postponing the discovery of issues and obscuring their source. Furthermore, performing an integrated end-to-end test requires predictable, repeatable transactions, yet all internal and external systems may not be available at the same time, or may not be easily simulated to test interactions throughout the supply chain.

Web Services Testing

Today, there is a new class of functional test solutions emerging to support end-to-end functional testing of SOA at each stage of the development process. These tools provide a universal GUI interface that is analyst-friendly, does not require any technical skills, and empowers subject matter experts to construct, send, receive, and verify messages using files, transports, or the actual applications. These same tools also support testing of the consumer and provider

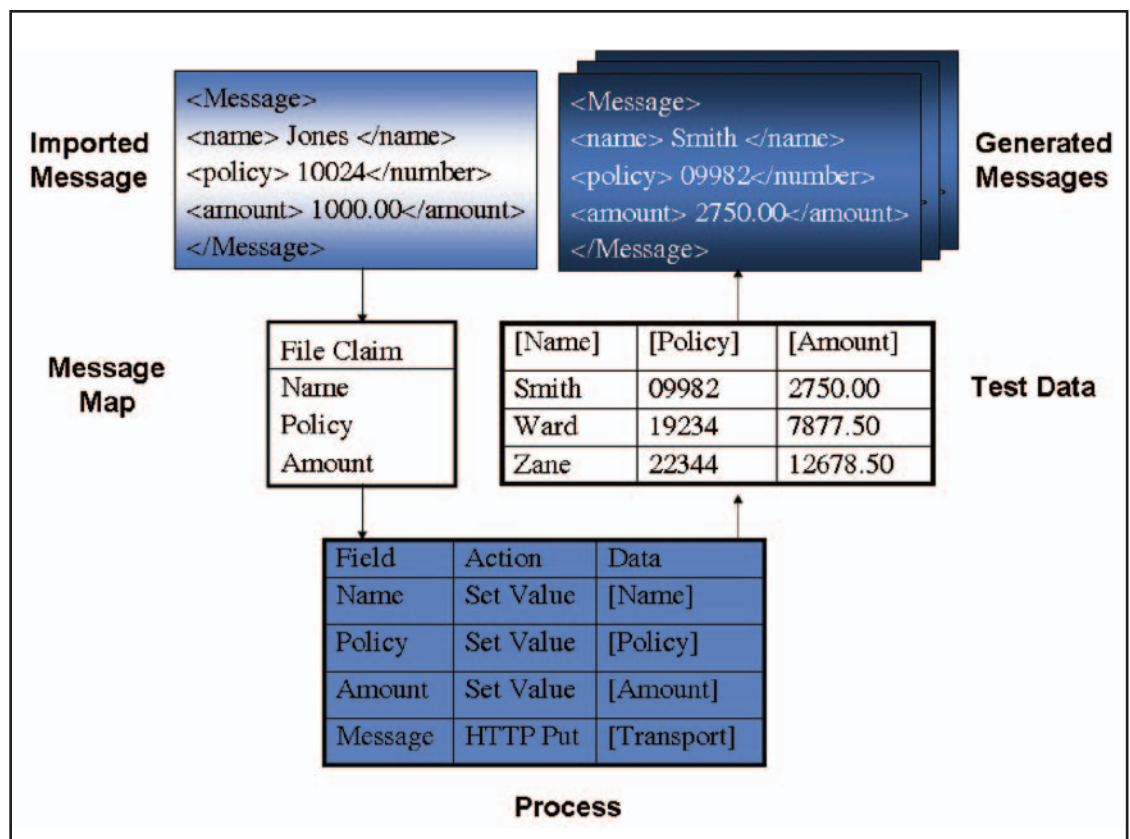


FIGURE 1 Masking the complexity of SOA allows business experts to focus on functional data content

applications through their traditional user interfaces, thus enabling true end-to-end testing, as well as isolating a single layer or point of interface.

Briefly, these tools map the message formats into their data elements and present these to the user in a drop-down, point-and-click interface. Users simply select the type of message they wish to send or receive, then supply the data values for each element to be created or verified. Default values from template messages can be used as a starting point, and only values that are pertinent to the particular type of test need to be supplied.

Once the message contents are defined, users can select from another set of options as to the target or source, such as a file for a simulated interface, or an actual transport for live interaction. This option can be modified as needed, so that when a new layer or application comes online, the target or source can simply be changed without affecting the message.

For example, users can isolate the transport by constructing and sending, or receiving and verifying messages to and from files,

instead of actual provider or consumer applications, thereby assuring that the message queues are functional, the encoding is correct, and the content integrity is maintained. They can also isolate a single application by sending or receiving messages from between the application and a file, instead of from the actual queue. By isolating each layer, issues can be quickly uncovered at their source and before they are proliferated to other layers.

As the architecture nears completion, users can selectively integrate each component into the whole, until they can finally test end-to-end from the provider application, through the transport, to the consumer application. At each stage, not only is the technical infrastructure being exercised but, more important, the actual message content and supported business process as well.

Since these advanced tools can be used throughout all layers of Web services, they can even simulate the correct behavior of faulty components so that the overall integration can continue while the malfunctioning component is fixed. This capability speeds time-to-market and bypasses delays that otherwise slow down test activities.

These tools also employ advanced test-design techniques that allow message schemas to be automatically merged with sets of test case data in order to yield comprehensive functional coverage. By abstracting the data from the schema, testers can focus on exercising business functionality as expressed in data values without being distracted by the technical implementation. Tests can further be organized into complete workflows that emulate complete end-to-end business processes, and provide traceability from high-level business requirements to the low-level implementation.

With the exception of the initial setup and configuration of the transport protocols and providing the schemas, the fact that there is no effort required by the development team to create or maintain custom test harnesses, drivers, and stubs offers a tremendous advantage to the IT organization, and keeps development resources focused on assuring timely delivery of a reliable test, while allowing business experts to assure the quality and accuracy of the final result.

The strategic benefit of these tools from

both an IT and a business standpoint is the ability for the tests to be reused, extended, and repeated automatically over time, and from release to release. This provides comprehensive test coverage for regression and new functionality to assure that there is no unintended impact. Only through extensive, automated test execution can companies truly keep pace with market demands, regulatory requirements, and competitive imperatives.

End-to-End Testing

While Web services are rapidly being adopted for internal integration when and where appropriate, and they are certainly the preferred – if not eventually required – method of integrating with external enterprises, it is a fact that most insurance enterprises have hundreds of applications in their portfolios that are still integrated through proprietary, custom interfaces and data stores. It may take years if not decades before the complete, end-to-end execution of business processes is supported by an SOA – if ever.

This reality means that test solutions cannot be SOA-centric. Instead, they must support a holistic approach that encompasses end-to-end testing throughout the enterprise. Orchestrating an integrated test environment can be a monumental challenge because incoming claims will cascade through perhaps dozens of interrelated applications, each of which has its own business requirements and release timetable.

This need also underscores the importance of a test automation solution that can encompass, but that is not limited to, Web services and SOA. Ideally, testing should be supported at all levels – from the application user interface all the way through the message layer – so that end-to-end verification of a complete business process can be executed seamlessly.

For example, a claim that is submitted from an external source will pass through adjudication, remittance, and other applications internally, and these may share data through a common data store, or through the exchange of transactions or files in any one of many formats and methods. Thus, proper end-to-end testing of claims pro-



Complex JAVA J2EE Hosting made easy.

WebAppCabaretsm

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



JAVA J2EE-Ready Managed Dedicated Hosting Plans:

Xeon I

**SAMEDAY
SETUP**

Dual 2.8 GHz Xeons
2GB RAM
Dual 73GB SCSI
1U Server
Firewall
Linux
Monitoring
NGASI Manager

\$279
monthly

Pentium 4 I

**SAMEDAY
SETUP**

**FREE
SETUP**

2.4 GHz P4
2GB RAM
Dual 80GB ATA
1U Server
Firewall
Linux
Monitoring
NGASI Manager

\$199
monthly
**2nd month
FREE**

4Balance I

1 Database Server
and 2 Application
Servers connected
to 1
dedicated
load
balancing device.
Dual Xeons.
High-Availability.

\$1724
monthly

At **WebAppCabaret** we specialize in **JAVA J2EE Hosting**, featuring **managed dedicated servers** preloaded with most open source JAVA technologies.

PRELOADED WITH:

JDK1.4 . JDK1.5 . Tomcat . JBoss . Struts . ANT . Spring . Hibernate
Apache . MySQL . PostgreSQL . Portals . CRM . CMS . Blogs . Frameworks
All easily manage via a web based control panel.

Details:

- All Servers installed with the latest Enterprise Linux
- Firewall Protection
- Up to 60 GB daily on site backup included at no extra charge per server.
- Database on site backup every 2 hours
- Daily off site database backup
- A spare server is always available in case one of the server goes down
- Intrusion detection.
- 24x7 Server and application monitoring with automatic self healing
- The Latest Bug fixes and Security updates.
- Tier 1 Data Center. 100% Network Uptime Guarantee
- Guaranteed Reliability backed by industry-leading Service Level Agreements

Log on now at <http://www.webappcabaret.com/jdj.jsp> or call today at **1.866.256.7973**

WebAppCabaretsm

JAVA J2EE Hosting

Prices, plans, and terms subject to change without notice. Please log on to our website for the latest price and terms. Copyright © 1999-2005 WebAppShowcase • All rights reserved • Various trademarks held by their respective owners.



cessing must take into account each of the affected applications. While using the message layer may be necessary for verification of the claims as received, it may be easier to employ the user interface of internal applications to check on a claims status, rather than write custom harnesses for multiple data sources or file formats.

Test automation tools that leverage the user interface have been available for 15 or 20 years, typically as record/replay products. These tools have allowed manual test processes to be captured into scripts, and then enhanced using programmatic commands to account for data variability and to handle exceptions, as well as manage other execution issues such as timing.

What sets the new class of test automation tools apart from their predecessors is their ability to not only eliminate the need for testers to learn and use script-programming commands, but to allow tests to be executed against message layers with no user interface and applications with a user interface. This means that analysts can test business processes end-to-end across the span of lower message layers and higher application user interface layers – seamlessly, within a single execution session.

The significance of enabling this capability is the shift of business process verification to the earlier stages of Web services implementation, which assures that defects are identified as quickly as possible when they

“ Ideally, testing should be supported at all levels – from the application user interface all the way through the message layer – so that end-to-end verification of a complete business process can be executed seamlessly ”

are easier to isolate, diagnose, and correct.

Summary

For the insurance industry, where technology is a competitive weapon, capitalizing on the promise of Web services with speed and accuracy is an essential component of an aggressive, successful IT strategy. Assuring quality throughout the implementation cycle and across all layers of the architecture is a key component of a timely delivery that does not compromise business operations.

Emerging test automation solutions that can span both the development process and the business process free up development re-

sources from coding specialized test interfaces and involve business experts as a core component of a successful Web services strategy. ©

About the Author

Linda Hayes, Worksoft's cofounder and chief technology officer, is a pioneer and recognized authority in the field of automated testing. She is an invited speaker at the industry's most prominent conferences, a regular columnist for *Computerworld*, *Datamation*, and *StickyMinds*, the author of numerous publications, including *The Automated Testing Handbook*, and the founder and former CEO of AutoTester, the first-ever automated, PC-based testing tools company. Worksoft is the result of Linda's vision for the next generation of automated testing solutions.

■ ■ ■ lhayes@worksoft.com

Communication Platforms

Can Web services be used in other communication platforms besides the Web?

Yes, they can. Web services are described by WSDL, which is used to describe a service using XML, not necessarily one that is used over the Web. In order to understand this last statement, you need to ask the question: What is necessary for operation over the “Web”? The answer, in one word, is “HTTP.” The Web implicitly means transporting information over HTTP as the communication protocol. In fact, as you probably know, one of the most popular means of using Web services is over JMS, as an alternative to SOAP over HTTP.

So the question arises – why are Web services called Web ser-

vices? Some believe that Web services really means a “web of services” rather than services that assume the Web. We can play around with definitions and words till the cows come home, but the key concept to grasp is that that Web services is a group of well-defined services, and don't automatically assume HTTP. Perhaps WSDL is the misleading term – it should just be an SDL (service definition language). ©

About the Author

The WSJ Editorial Board comprises distinguished professionals in the technology field. Here they share their expertise with the readers by answering frequently asked questions about Web services-related topics.

■ ■ ■ WSJFAQ@sys-con.com

PRESENTED BY
**WSJ EDITORIAL
BOARD**

EDITED BY
AJIT SAGAR

Register, with credit card payment, by October 7 and receive \$200 off the standard registration fee. (Mention Priority Code APN15WSJ to receive this discount.)

Media Partner:
WebServices
JOURNAL

Gartner **Application Integration & Web Services Summit** 2005

Gartner **Open Source Summit** 2005

SOA and Open Source Go Mainstream

December 5–9, 2005 • JW Marriott Grande Lakes • Orlando, FL

Maximize Your Time & Travel Investment



The Industry-Defining Application Integration & Web Services Summit and Brand-new Open Source Summit.

Both in One Terrific Location
\$695 Savings When You Attend Both Events!

Benchmark your enterprise strategies on:

- Service-Oriented Architecture (SOA)
- Open Source
- Event-Driven Architecture (EDA)
- Web Services
- Application Integration and Middleware And More!

Back by popular demand:



Tim O'Reilly, *Founder & CEO of O'Reilly Media*

You've read his books, now hear him live on open source, standards, the "architecture of participation", and the next-generation Web

Conference Chairs:



Roy Schulte
Gartner VP and
Distinguished Analyst



David Smith
Gartner VP and
Research Fellow



Mark Driver
Gartner
Research VP

Register now at gartner.com/us/aiws • gartner.com/us/opensource

Gartner

BPM Suite from Bluespring Software

Strong Web services capabilities and an intuitive user environment

■ In the past, business process management has not been a significant area of concern for many corporations. However, with increased regulatory scrutiny facing companies today, the need for formalized definitions, checks and balances, and management oversight is a reality.

The Challenge

A number of technical solutions have entered the marketplace to assist with the challenges that these requirements present. One such offering is the BPM Suite from Bluespring Software. The BPM Suite is a Microsoft .NET-based system that provides a technical foundation for the design, management, and execution of business processes in an environment of heterogeneous systems.



WRITTEN BY
**BRIAN
BARBASH**

data interface system and submitted to the business process manager via a Web service invocation

2. The report is routed as a task to the appropriate user by role based on the report type; Product errors to a Product Analyst, Sales errors to a Sales Analyst, and Customer errors to a Customer Analyst
3. An external manual process is performed by the user to correct data errors
4. When all errors are corrected by users, a new Data Load is initiated in the data interface system

Along with the general process model, a number of run-time variables have been created to manage the transfer of stateful information between the various activities in the model; the most critical of which is the Error Report. The Error Report is an XML document that contains the details of issues encountered by the interfacing process. XML documents are stored as text strings within the process

model and may be parsed by the XML Reader activity. This activity uses XPath statements to extract specific values from the XML document. In this example, the reader extracts the report type, which is used to determine the user role responsible for addressing items in the Error Report document.

All business processes created in the BPM Designer may be initiated through a Web services call. The service calls include all run-time variables as parameters allowing external applications to set any necessary initial values. For this example, the Error Report value is provided as part of the call from the data interfacing system.

In addition to process models being initiated by Web service calls, process models may also invoke external Web services. The BPM Designer allows the developer to import a WSDL definition from a URL. Once specified, the parameters of the service call must be mapped to run-time instance variables in the process model. The Web service invoker also supports authentication, proxy servers, and SSL certificates.

Building the Business Model Around the Process

Building and structuring business processes in the BPM Designer consumes the vast majority of the effort required to build an application. However, the BPM Suite also provides some simple yet significant business modeling capabilities that greatly impact functionality of process models, the first of which is the design of the security model within the tool. In any business process model, the roles that the users play in the execution path are important. The Bluespring product looks at users and security both from a traditional

Designing Business Processes

The BPM Designer, shown in Figure 1, is the main development environment for the BPM Suite and provides all of the tools for modeling business processes. Out of the box it ships with several activity types, some of which include:

- Database actions
- User interactivity including e-mail, instant messaging, task lists, and document creation
- Web Services integration
- Business Rules processing capabilities
- XML document handling

For the purposes of this review, I have created a simple process model that manages the business functions related to data errors when interfacing between systems (Figure 1). The process is defined as follows:

1. An Error Report is generated by the external

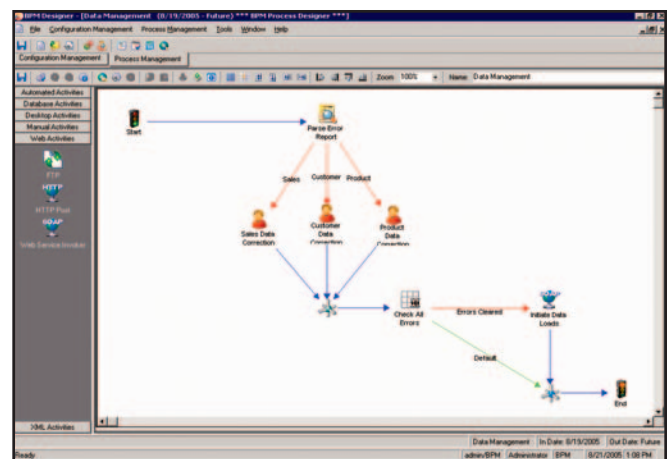


FIGURE 1 BPM Process Designer

“ The Bluespring product looks at users and security both from a traditional access control standpoint (users/roles) as well as from an organizational standpoint ”

access control standpoint (users/roles) as well as from an organizational standpoint. Organization charts, shown in Figure 2, can be created and associated with manual tasks within the BPM Designer. In this example, an org chart has been modeled to capture the reporting hierarchy of the analysts responsible for each type of Error Report: Customer Analysts, Sales Analysts and Product Analysts. The org chart is then assigned to each of the manual tasks in the Data Management business process. Now the business process is independent of the specific IT security structures, which often evolve in an organization, and reflects the true business responsibility hierarchy. As the business hierarchy changes, the org chart may be updated and the business process automatically reflects this change without any additional development effort.

The second component of business modeling in the BPM Designer is the concept of Geographies. As the name implies, a Geography represents a physical location. Locations may be incorporated into process models to determine users available to complete manual tasks.

User Interaction

In the example business process created for this review, there are manual tasks that must be completed in order for the process to finish successfully. User tasks are managed in the BPM Web application, shown in Figure 3. This application provides an out-of-the-box Web portal for viewing, updating, and completing user tasks, as well as basic administration of active processes, depending on user privileges. In the example shown in Figure 3, a task has been created for the Sales Analyst. From this screen, the user may double-click on any assigned task, update the required properties, and complete the task.

As effective as the BPM Web portal is for managing tasks, many organizations will require integration of user tasks and activities into existing portals or other applications. To facilitate this requirement, the BPM Suite provides a full Web services API for task management and administration. It provides access to all tasks and the data supporting those tasks. In the case of the example process created, the full Error Report document is accessible in its raw XML form,

providing for the application of stylesheets for presentation to the user.

Summary

Business process management is becoming increasingly important to today's corporate environment. Challenges abound for IT professionals because BPM solutions must typically cross system boundaries and business departments. This requires strong systems integration capabilities and an intuitive business user environment. Bluespring Software's BPM Suite is an offering that helps address these challenges. It leverages the .NET platform's strong support of Web services with some well thought-out functionality to provide a solid foundation on which to perform business process management.

Notes

At the time of this writing, Bluespring is preparing the release of version 4.3 of the BPM Suite due in October of 2005. Some highlights of the new release include:

- Embedded business rule engine
- Integration with Visio to support diagramming in the tool using a Bluespring-specific template or a standard BPMN template
- Updated Sharepoint integration

About the Author

Brian R. Barbash is the product review editor for *Web Services Journal*. He is a senior consultant and technical architect for Envision Consulting, a unit of IMS Health, providing management consulting and systems integration that focuses on contracting, pricing, and account management in the pharmaceutical industry.

■ ■ ■ bbarbash@sys-con.com

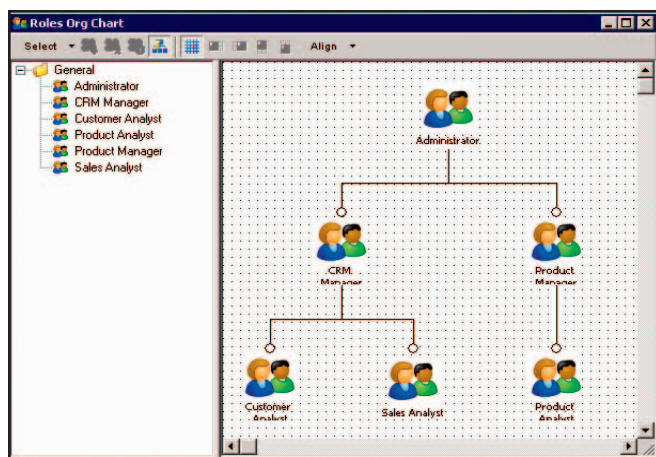


FIGURE 2 Organization chart modeler

Task Name	Account Name	ID	Status	Priority	Due Date	% Complete
Sales Analyst		0	Manual Task	5 (Normal)	8/29/2005 4:34:30 PM	0%

FIGURE 3 BPM Web

How H&R BLOCK Has MAXIMIZED Its Return on SOA

Counting on management infrastructure

■ Working with service-oriented applications is a lot like doing your taxes because: there's really no avoiding it, you need to keep track of a number of different factors, you must maintain a record of data to protect against audits and plan for the future, and you'd better catch any mistakes or you'll have a problem on your hands faster than you can spell "SOA."

Unless you implement management and monitoring capabilities for your loosely coupled system, you expose yourself to any number of potential issues that can cripple a production SOA application. Worse yet, you might unleash problems that you won't even know about until they've impacted your business. Sound daunting? It can be. The good news is that there are solutions to solve the taxing problem of SOA management.

A great example of how to implement SOA management the right way can be found at H&R Block Financial Advisors, the Detroit-based subsidiary of H&R Block, Inc.

A Leading Financial Services Company

H&R Block Financial Advisors offers



WRITTEN BY
FRED CARTER

investment planning, advice, and related financial services and products. With more than \$28 billion in assets under management, they are one of North America's largest broker/dealers.

H&R Block Financial Advisors receives a wealth of leads from the parent company's tax services organization. Consider that H&R

Block handled more than 21,000,000 tax clients in 2003. Many, but not all, of these tax clients are excellent prospects for the financial advisors. To ensure that only qualified leads are routed expeditiously to the right financial advisor at one of more than 250 offices across the country, the company requires sophisticated lead-routing capabilities. When a hefty license renewal fee came up for their previous lead management system, they took the

opportunity to make the move to SOA and implemented the services-oriented Client Acquisition System, or CAS. CAS aggregates roughly 3,000,000 leads in a single tax season. It qualifies them and then routes them based on criteria specified by the branch managers at each financial advisor office.

Here's how it works. CAS executes a series of orchestrated Web services to pull

Customer Acquisition System (CAS) SOA Environment

- Microsoft Windows 2003
- TIBCO BusinessWorks
- Development on Microsoft .NET Framework
- Microsoft Customer Relationship Management
- Packaged applications with Web services interfaces
- Complex interactions across Web services owned by different IT groups
- AmberPoint SOA management software

“ Exception management has always been and will always be a problematic area for IT ”

together the pieces of information required to qualify, route, and assign leads to a financial advisor. TIBCO BusinessWorks is the CAS orchestrator that manages the execution of 15 .NET Web service requests per lead. If a lead is qualified, a final Web service call is made to the CRM system in order to process the lead and assign it to an advisor. The financial advisors have specialized views in their CRM application that enable them to see new leads and to filter prospects based on the criteria of their choosing.

Handling Exceptions

As with all production systems, CAS is not immune to exceptions. The number and variety of data sources, combined with the evolutionary nature of distributed applications, causes errors and unexpected conditions that mandate comprehensive exception-handling capabilities.

“Exception management has always been and will always be a problematic area for IT,” said Scott Thompson, head of application development for H&R Block Financial Advisors. “Unfortunately, even though it’s an absolute necessity for most systems, it tends to pull our attention away from the things the business really cares about, like routing leads. By implementing a dedicated SOA management system we not only had more time to focus on the leads, but also established a framework we can use across the enterprise.”

H&R Block uses AmberPoint SOA management software to manage exceptions. AmberPoint’s agent-based architecture provides centralized visibility into system-wide, services-based interactions. It identifies unique message flows from end to end and scans messages for user-defined exception conditions,

including those that span multiple service interactions. Upon detecting an exception, it alerts individuals and other systems. At the same time, it automatically assembles queryable message trails for inspection by users. Because all messages are available in a single facility, the company’s developers don’t have to search multiple log files to identify the sources of problems. All of the captured operational data is provided via standard SQL reports.

To ensure that H&R Block Financial Advisors’s lead-routing system operates smoothly, the management system monitors the traffic, checks fault codes, and handles exceptions as required. Some exceptions need only to be logged, while others require immediate resolution. For example, if the source system sends invalid data or CAS can’t execute a qualification for determining the quality of a lead, AmberPoint alerts the developer assigned to support the source.

The management system significantly reduces the time it takes H&R Block Financial Advisors to detect, diagnose, and resolve system errors.

“Web services complicate the management of exceptions because of their ‘black box’ nature,” explained Thompson. “Prior to AmberPoint, we relied very heavily on our consuming applications to tell us when errors occurred. In particular, HTTP-related errors were very difficult for us to detect and place metrics around. It was also hard to justify taking the time required to develop our own custom solutions to provide the level of visibility we knew we needed. AmberPoint provides that visibility, usually identifying errors before our consumers know they occurred. We can also look back over time and identify problematic areas that may need to be addressed in future releases of our services.

“Our management system gives me visibility into the traffic so I can see the actual information that’s passed into the service and the information that came back out,” said Thompson. “We also benefit from the system’s ability to serve as a QA mechanism. If there’s a bug in the code, we can fix it, redeploy the code, and use AmberPoint to resubmit the same message to see if the problem persists.”

Since taking CAS into production, H&R Block has caught and handled more than a million exceptions – including errors and other system events. For example, an ignored lead is captured as an exception and quickly remedied.

Management Capabilities

- Application-level visibility into the system
- Early warnings, impact analysis, and timely mitigation actions to prevent service problems
- Detailed usage and trend analysis
- Monitor all services for faults and errors in real time
- Capture and centrally log transaction instances from end to end for diagnostics
- Automated response to minimize business impact of exceptions

Management Benefits

- Assures operational health of the system
- Sends alerts about Web service errors immediately
- Rapid exception diagnosis and shorter mean-time to resolution
- Uniform exception governance across SOA
- Identifies areas that require refinement
- Lower cost of application development and production support

Ensuring the Operational Health

Service level management is another pressing concern for SOA. By managing service level agreements, or SLAs – the formal guidelines they set for acceptable performance levels of applications and services – H&R Block is able to identify and mitigate performance bottlenecks before they can impact the business. They also use this detailed service level data to fine-tune the system for better performance.

Thanks to this greater visibility and the ability to catch performance issues before service level agreements are violated, CAS has achieved 100 percent availability. It has not missed a lead since going into production and has performed flawlessly through the peak loads of this year's tax season.

“ Since taking CAS into production, H&R Block has caught and handled more than a million exceptions – including errors and other system events ”

Abstracting the Management Layer

Whether you build your own SOA management system or purchase one, you'll want to give careful thought to the underlying mechanisms used to provide management capabilities. If you hard-code the management logic into the Web services themselves you'll be challenged to maintain your management capabilities as your system evolves. Having your Web services and clients embed proprietary headers into SOAP requests and responses would limit your management capabilities to those system components over which you have direct control. So, for example, you'd have no visibility into the performance of a partner's Web services. The best approach, therefore, is to abstract the management layer by using Web service intermediaries. With this approach, all of the management policies reside in the intermediaries, thus enabling your system to grow quickly and flexibly – and without added development-time constraints.

What's more, by abstracting the management layer you enable your developers to focus on the business problem at hand.

A Healthy Return

After its first full tax season in production, CAS has already delivered an impressive return on H&R Block's investment in SOA. As compared to the previous lead-routing system, CAS has reduced their support staff from ten dedicated resources to three shared ones. License costs are a fraction of what they used to be. Development costs are significantly reduced as well – as much as 20 percent lower. Leads are routed quickly and effectively to the advisor best suited to work with the prospective customer. In 2005, the volume of leads rose to 3,000,000, 30 percent of which were routed to financial advisors. With the old system it could take as long as 24 hours for a lead to be delivered from a tax office to a financial advisor. Even under the highest load, it now takes only seconds. Most important of all, the company's revenues have risen significantly, due in large part to the better-qualified leads they're sending to their financial advisors. ©

About the Author

Fred Carter is the chief run-time architect for AmberPoint, a provider of Web services management software. Prior to AmberPoint, Fred was the architect and technical lead for EAI products at Forte, a role he continued at Sun Microsystems. Prior to Forte, he held several technical leadership positions at Oracle, where he designed distributed object services for interactive TV, online services, and content management.

■ ■ ■ fcarter@amberpoint.com



WEEKEND WITH EXPERTS

Get a Java injection on the go!

Spend a weekend and have a lunch with experts in an intimate learning environment in the city near you.

No salesmen allowed. Join our technical discussions and hands-on workshops.

The next Roadshow is in Philadelphia on December 10 and 11.

www.weekendwithexperts.com



Visit the *New*
www.SYS-CON.com
 Website Today!

The World's Leading i-Technology
 News and Information Source

24/7

JUMP TO THE LEADING i-TECHNOLOGY WEBSITES:

IT Solutions Guide
 Information Storage+Security Journal
 JDJ
 Web Services Journal
 .NET Developer's Journal
 LinuxWorld Magazine
 Linux Business News
 Eclipse Developer's Journal
 MX Developer's Journal
 ColdFusion Developer's Journal
 XML Journal
 Wireless Business & Technology
 Symbian Developer's Journal
 WebSphere Journal
 WLDJ
 PowerBuilder Developer's Journal

FREE NEWSLETTERS

Stay ahead of the i-Technology curve with
 E-mail updates on what's happening in your industry

SYS-CON.TV

Watch video of breaking news, interviews with industry leaders, and how-to tutorials

BLOG-N-PLAY!

Read web logs from the movers and shakers or create your own blog to be read by millions

WEBCAST

Streaming video on today's i-Technology news, events, and webinars

EDUCATION

The world's leading online i-Technology university

RESEARCH

i-Technology data "and" analysis for business decision-makers

MAGAZINES

View the current issue and past archives of your favorite i-Technology journal

INTERNATIONAL SITES

Get all the news and information happening in other countries worldwide

Does Your SOA Achieve Agility?

SOA actualization: enterprise agility



■ Agility seems to be the buzzword du jour. There are “agile enterprises,” “agile manifestos,” “agile programmers,” and “agile architectures.” Slap “agile” in front of just about anything and marketers believe it will sell better. Yet, one of the primary goals of using service-based architectures was to create “agile systems.”

This begs the question, was it just marketing or is there something to it?

Let's cut to the chase. If you are responsible for a system and a user requests a change to that system, how long will it take to make the change and move the revised system back into production? If it takes a long time, then your system probably isn't agile. If you can do it in a matter of days – well then you have agility! Marketing bologna aside, agility is directly related to the time and effort required to create new functions or to modify existing functions – and then to re-release those functions to the customers.

Small systems are usually agile. As systems grow in size and the dependencies increase, it becomes hard to make changes and re-release. Service-oriented architectures (SOA) are thought to hold the potential to improve this scenario, but do they? More important, does your instantiation of an SOA provide agility?



WRITTEN BY
**JEFF
SCHNEIDER**

Finding the Problem

Imagine a system with 100 parts. Pick a random number from 1 to 100. The number you chose was the part that needed to be changed. How many other parts used that part? How many did it use? In a random world we say that statistically all changes are equal; however, the world isn't random.

Take a software system that has been in production for an extended period of time and review all of the user-initiated change requests. Create categories for the changes, such as:

- Create a new system function/capability
- Capture new data in an existing function
- Add or change a business rule

Then, determine where the time was actually spent, for instance:

- Updating requirements, documents, and specifications

- Locating the responsible code
- Making the code changes
- Building the system
- Testing the changes
- Redeploying the system

Now, add services to the mix and rerun your “change requests” to see if your situation improves. Today, for most organizations, the agility isn't increased – in fact it dramatically worsens. Why? It can be summed up as follows:

- Developers don't have the proper SOA tools
- Developers aren't designing the services properly
- Developers aren't building the composite application “loosely”
- Some systems are not good candidates for SOA-infused agility

Agility and Tooling

Just when most developers were getting efficient at using their platform tooling (Visual Studio, Eclipse, etc.), a whole new set of service-oriented opportunities surfaced. For the last several years the open source community and leading vendors have rushed to bring new tools to the market. However, many organizations have been slow to adopt tools that facilitate SOA. Does your organization have standardized tools for:

- Creating XML Schemas and WSDLs?

- Rapid Service Development (contract first)?
- Intermediary-based routing and mediation?
- A registry of services? A repository of service metadata?
- Automated service testing?

Obviously, this is only a partial list of the tools needed to be productive in a service-oriented environment. The point is, don't expect developers to be productive in an SOA environment using last-generation tools. It won't happen.

Agility Techniques

Having the right tools is an easy way to increase productivity, but it will only get you so far. Ultimately, we must come back to the techniques we use to create the services, combine them, and redeploy them.

Agile Service Design

Designing services is no easy task. We are faced with many of the same questions that we saw in object-oriented design: How do we design services so that they can be easily maintained? How do we refactor for reuse? We are also faced with new questions such as, "How do we keep our platform-to-schema bindings synchronized?"

With regard to refactoring for reuse, it is clear that we must implement the same basic rules of commonality that we applied to object-oriented design – that is, using Venn diagrams to identify commonality and carve it out. It is clear that "fat services," or services that contain too much logic, are likely to have lots of change requests issued against them. This has a ripple effect that causes the entire service to be versioned as well as all of the clients, depending on that service to be versioned. This doesn't sound very agile, does it?

Alternatively, server-side code that is specific to a client can be refactored into a service that reduces the impact of changes required by a client. The goal is to reduce the impact of a change while leveraging common code. If we refactor our scenario into four services (one common and three services containing client-specific features), we are able to preserve the reuse and reduce the impact of making a change.

Now we all know that this type of extension mechanism won't always work. Luckily, the software community has seen this type of problem on more than one occasion and the design patterns to remedy the variations of this problem are abundant. For virtually every "Gang of Four" pattern that discussed an object-oriented solution, there is a minor modification that can be applied to create a service-oriented variation. Remember though, on occasion – the answer isn't service-oriented at all. Don't go throwing out your favorite OO patterns and trying to solve problems in a service-oriented way. Services are a mechanism to complement your current bag of tricks. Implementing best practices in design (service, object, or otherwise) is essential to creating systems that are easily modifiable.

Composite Applications

Service-oriented systems are composed of three primary elements: the service infrastructure (SOAP routers, firewalls, etc.), the services, and the clients. However, there is a special breed of the client that is worth observing – the composite application. In loosely

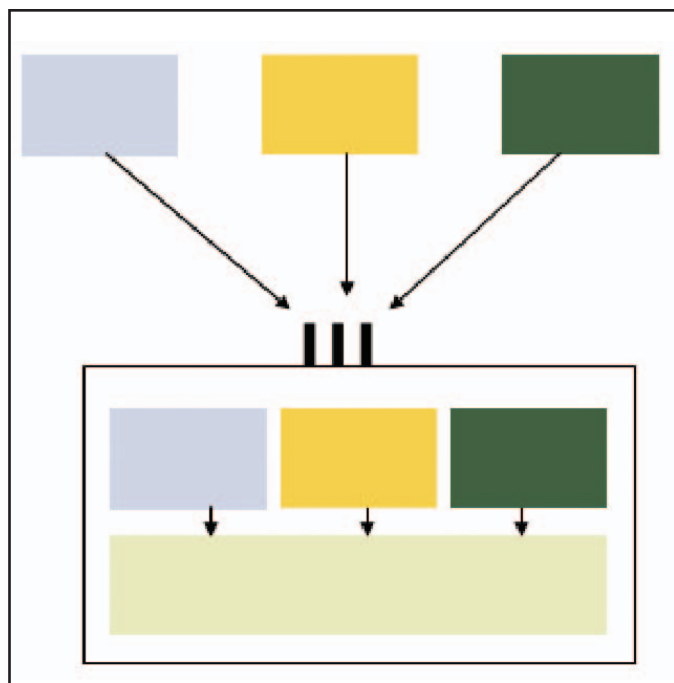


FIGURE 1 Elements

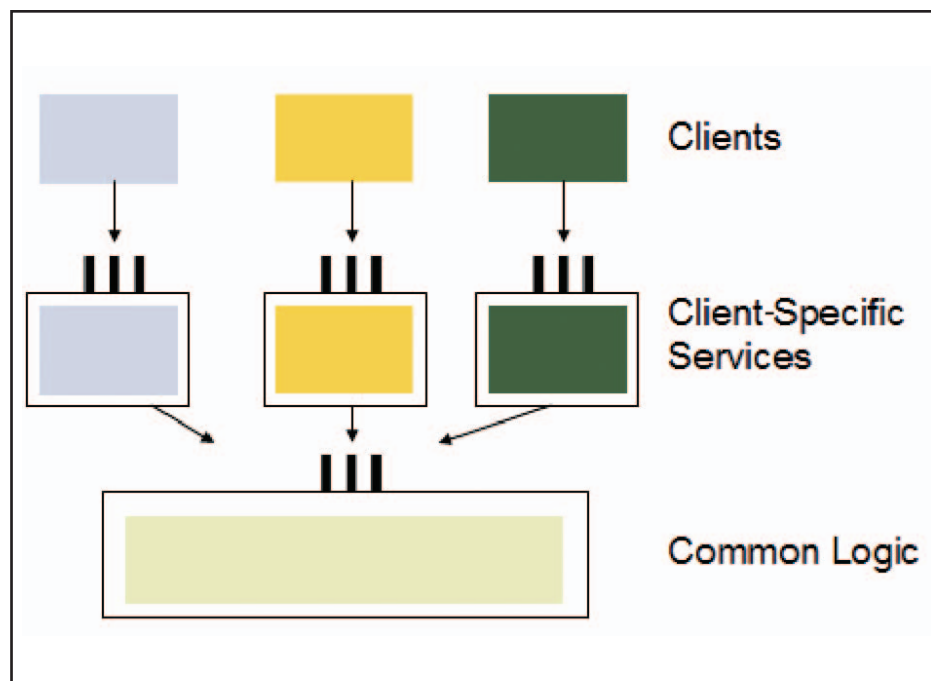


FIGURE 2 Service elements

coupled systems, we identify certain portions of the system that seem to pull it all together. Aside from executing the business services, composite applications interact with the user interface and external gateways. The composite application is the brain of the service-oriented system.

Composite solutions focus on three different areas: agile controllers, reflective metadata, and WYSIWYG/RAD.

Agile Controllers

The agile controller is a piece of business logic that dictates business logic relative to code that is likely to change. Typically, agile controllers focus on the flow of information between humans (workflow) or systems (orchestration). The goal of the agile controller is to give the programmer an easy way to make a change to the portion of the code that is *most* likely to change!

Reflective Metadata

As the enterprise continues to refactor substrate out of applications and put it in common infrastructure, it becomes more important to gain access to that infrastructure. Modern enterprise IDEs place a significant emphasis on accessing enterprise metadata – everything from simple WSDLs to semantic message descriptions and run-time policies. Metadata is the new data.

WYSIWYG/RAD

Since composite applications leverage metadata extensively, it becomes critical to apply smart IDEs that have RAD (Rapid Application Development) capabilities. Dragging and dropping WSDLs and transformations is the norm.

Although the composite application is the brain of the service-oriented system, making changes to it must not require a brain surgeon. When the logic that is most likely to change is placed in an agile controller and appropriate metadata and RAD tools are available, the time

“ Implementing best practices in design (service, object, or otherwise) is essential to creating systems that are easily modifiable ”

and skills required to modify our systems is significantly decreased.

Agile Redeployment

The goal of IT is to put valuable systems in front of our users in a timely manner. Deploying and redeploying in a short time frame is essential to achieving agility. You can implement all of the aforementioned techniques, but if you don't have a facility for rapid redeployment you will have lost virtually all of your gains.

The problems associated with redeployment are usually focused on: right-grained builds, router-based versioning, and hot-deploys.

Right-Grained Builds

In non-service-oriented systems, there is a tendency to take all of the code and compile it all together. In service-oriented systems we must change the granularity of the build. You should be either building (or rebuilding) an individual service or the composite application. Remember – the granularity of what we're re-releasing is often much smaller than what we did in the past. Re-release only those items that have changed!

Router-Based Versioning

Service-oriented systems encourage reuse. However, this creates a new problem where many clients become dependent on a service.

This means that we will need the ability to release new versions of services without hindering those clients that require continuity. Modern SOAP routers can be easily updated with versioning information and can be told to re-route messages to the right endpoint without having to change client code.

Hot Deploys

Most modern application servers have the ability to deploy new functionality without having to take the entire system down. In a service-oriented world, this becomes a “must have” feature. This can be accomplished by having tiny servers (1:1 ratio of servers to services), or in the more traditional manner of having larger servers that can “uninstall” the old service and “reinstall” the new one.

Agility is not something that is natively found in a service-oriented architecture. It is something that must be achieved. Agility will never just fall out by accident; it must be architected, studied, and improved upon. Ultimately, it is safe to say that SOA does provide significant attributes that enable agility – but it is your job to bring them to the forefront. ☺

■ About the Author

Jeff Schneider is founder and CEO of MomentumSI, a consulting firm that specializes in transforming organizations into service-oriented enterprises.

■ ■ ■ jschneider@momentumsi.com

“

Metadata is the new data

”

Reach Over 100,000

Enterprise Development Managers & Decision Makers with...



Offering leading software, services, and hardware vendors an opportunity to speak to over 100,000 purchasing decision makers about their products, the enterprise IT marketplace, and emerging trends critical to developers, programmers, and IT management

Don't Miss Your Opportunity to Be a Part of the Next Issue!

Get Listed as a Top 20^{*} Solutions Provider

For Advertising Details Call 201 802-3021 Today!

*ONLY 20 ADVERTISERS WILL BE DISPLAYED. FIRST COME FIRST SERVE.



The World's Leading i-Technology Publisher

The Momentary Enterprise

Thinking outside the VC box

■ Prior to the year 2000, business was a world in love with office spaces and corporate travel. We traveled to work (the office) every day. We traveled away from the office for customer meetings, for internal meetings, for conferences, for awards ceremonies. We traveled because we could and we believed that it was necessary for the competitive advantage. That all changed rather quickly with the economic downturn of the early 2000s and, of course, 9/11. In short order, we relearned how to do business by staying put.

Another consequence of the most recent recession was corporate payroll compression. As good corporate citizens who are mindful of keeping our jobs, most of us were tasked with far more work than a single individual could perform adequately. To help make us more efficient, we introduced cell phones so that we could make calls while traveling, and WiFi-enabled laptops that enabled us to take our work anywhere and exchange messages with anyone, provided a “hotspot” could be found.

Conference calls, cell phones, and laptops have pushed personal multitasking to unparalleled levels. For example, workers are now able to process the relentless stream of e-mails and instant messages as well as review financial and marketing documents, all while supposedly being active during a conference call.

The preceding has been unsettling for some of us, but not so for others. In fact, for those who are able to do three things at once, it is a stress reliever. Prior to this work style, employees who were required to be physically



WRITTEN BY
JOHN WEBSTER

at meetings (even on-site) would worry about e-mail that was piling up as well as a half-written product presentation due the next day. Stress would slowly build as the meeting drew long. The productiveness of a conference call definitely suffers because multitasking participants are only slightly paying attention. However, no matter – overall efficiency is increased for those who can draw

the connections together and somehow manage to stay engaged in the “here and now” at the same time.

Consider where we’ve come from and

where we’ve now arrived. Before we had limited ways to “connect.” Now we can create multiple connections and converge them. Perhaps even more important, we can do much of this connecting and converging ourselves, without advanced degrees in IT or telecommunications, and using technologies that are available at the local shopping mall. Also, we can create our own virtual office spaces.

Building the Momentary Enterprise

Among the many consequences of connectivity and convergence that lead to the virtual office space is the momentary enterprise. The momentary enterprise takes advantage of an opportunity that may only exist for months. When it has been fully exploited, the momentary enterprise is reconfigured – Lego-like – to pursue another opportunity. A “pop-up” business model is born, thereby changing the competitive balance and leveraging pervasive data.

It is likely that in the coming years we will see sizable businesses (based on yearly revenue) come and go at what will seem to be alarming speeds. Businesses can now be created in a matter of weeks to capitalize on whatever important trend or market demand is surfacing. Huge amounts of venture capital in many cases will not be required for establishing the infrastructure, the billing and accounting systems, the transport or supply systems, the IT function – and the list goes on. Such commodities will be expertly and automatically leveraged by super-deep, business-to-business automation, and new enterprises will start up by focusing their energy on differentiating their value in the mar-

“ It is likely that in the coming years we will see sizable businesses (based on yearly revenue) come and go at what will seem to be alarming speeds ”

**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



**Subscribe
ONLINE**
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!



ketplace rather than creating and supporting all of the associated accoutrements. Also, the fact that their life spans will be measured in months or a short number of years will not be grounds for dismissal from Harvard Business School. A successful business model doesn't need to be measured by its staying power.

Interested? Below is a checklist of things that will help you on your way to the momentary enterprise.

Hybrid PDAs (Personal Communicators)

Hybrid PDAs combine three entirely different business and personal communication mechanisms (cell, e-mail, IM) into a single embraceable device. Their expected ubiquity will enable significant efficiency gains in both our business and personal lives. Hybrid PDAs offer three important features for momentary enterprise entrepreneurs:

- The ubiquity of cell phone use – after all, it is no longer “weird” to be on your phone while in a supermarket, and you can be sure that someone next to you on the street is carrying a cell phone (to the same extent that you can be sure that those around you are wearing shoes)
- The seemingly overnight change from one type of cellular technology that is optimized for voice to a completely different architecture that is optimized for voice *and data*
- The immediacy of digital information sharing when you're away from a computer desktop

The trend toward holding your electronic life in the palm of your hand started when cell phones were crossed with PDAs. The result was a cell phone that could also keep your calendar and your contact database. Next, text messaging was added, and then Internet access and e-mail. Furthermore, there appears to be no end in sight to the future versatility of these multifunctional, handheld devices. For example, NTT DoCoMo has introduced in Japan a cell phone with so many functions that one could get lost just trying to find them all. This unit will do the following:

- Send and receive e-mail
- Play games online
- Access iMode-compatible Web sites and

play downloaded music

- Take digital photographs
- Record sound
- Read bar codes (and someday RFID tags)
- Oh... and make phone calls

It also contains a specialized Sony-developed chip called a FeliCa chip or “smart card” that enables users to pay bills and make purchases over a wireless electronic banking system, operate appliances that can be controlled via a connection to the Internet, unlock doors, and the list goes on. Sooner or later, most of us will carry an intelligent device that is wirelessly connected to the world's vast data networks continuously.

XML

XML formatting allows proprietary databases and records to now have a nearly universal method for describing their contents. One does not need to be a sophisticated programmer who understands how to read a “schema” document or how to encode SQL statements to make sense of XML statements. A computer-literate teen could happen upon an XML fragment and derive some sense from it. He or she could likely import it into a favorite spreadsheet package and sort or average or trend it with a few keystrokes.

Business back ends are now XML-crazy. Information that needs to be expressed to another computer system is now expressed in some XML format. Most significantly, XML enables far higher business-to-business cooperation that is squarely aligned with the Web's chief goal: information exchange (as opposed to data exchange). XML has been enthusiastically embraced by business and allows for significant efficiency gains and better customer experiences. We will see XML reaching into the consumer world and our homes as well via wired and wireless appliances, for example. For the momentary enterprise, XML is the magic glue that allows vast sources of data and internetworking infrastructure – from PDAs to wireless video cameras – to share information.

Virtual Officeware (VOWare) and RIAs

Currently, groupware is an effective collaboration tool used for large project management by equally large companies. For smaller

companies, ad hoc usage of a shared file server often suffices. Small Web servers are also now used by more physically diverse groups.

Groupware, despite early hype, has remained somewhat hidden – deployed appropriately by those large multiregional companies, but avoided by the rest of the world. A trained administrator is usually needed to set it up, and typically it requires an additional capital investment in server and storage hardware and administration. Groupware also focuses heavily on managing shared document repositories, thus making it vulnerable to competing products with a broader vision.

Groupware is currently being replaced by a groupware-like application with far more functionality – such as VOWare. Virtual office software was born as a result of our increasing work mobility and an increasing need for groups to work on projects jointly while spanning locations. Customers and suppliers outside the company can also be included.

Inescapable Data
Harnessing the Power of Convergence

Chris Stakutis and John Webster

This article is based in part on content excerpted from the book *Inescapable Data: Harnessing the Power of Convergence*, by John Webster and Chris Stakutis, published by IBM Press in May, 2005. (Copyright 2005 by International Business Machines Corporation. All rights reserved) ISBN 0131852159. To learn more, please visit: www.ibmpressbooks.com/title/0131852159.

Groove, recently acquired by Microsoft, is an example of virtual office software that combines the attributes of a shared file server, an instant messaging function, and project management tools (e.g., calendar and others). It can be deployed in minutes by users of the application themselves and does not require an investment in additional hardware.

VOware also adds a new level of “awareness” to communication. A bit like IM on steroids, VOware includes the concept of a “workspace” (i.e., a project) with group members, digital assets such as files, and tools such as whiteboard sharing and calendars and so forth. However, it is the combination of these elements that allows for a much deeper insight into what your fellow peers are doing without actually being in their presence. In fact, you can be more aware of what they are doing at any given time than you could if you were all working in the same office complex.

Another more advanced and still-emerging form of groupware is the Rich Internet Application (RIA). Some of the key critical components of RIA are:

- **Rich Web Experience.** Features such as the ability to “drag-and-drop” that we take for granted when using normal workstation applications are notably missing from the current Web-based versions of groupware ones. These features are added back in.
- **Live Data.** In order to be used pervasively, business application endpoints need to have “screens” and panels that show live data (live inventory, collaboration messages, etc.). In the existing world of Internet/HTML applications, such views are typically impossible. The goal of RIAs is to make live data screens possible.
- **Zero-Install Clients.** Users are relieved of the need to install applications and application upgrades on their computing devices. This need becomes more acute as the number of endpoint devices soars.

There are a few companies today such as NexaWeb that offer RIA development environments for businesses to build rich, pervasive Internet applications.

Video Cameras and Webcams

The popularity of consumer digital cameras has driven down the price of their

“ For the momentary enterprise, XML is the magic glue that allows vast sources of data and internetworking infrastructure – from PDAs to wireless video cameras – to share information ”

main components and the quality up. The key component of a digital camera of any type is the CCD element (the chip that actually “sees” a scene and turns it into a digital blip). The price of this component has dropped so low that high-resolution color video-capture devices can be built in to nearly anything. In parallel, wireless networking stormed through the computer industry over the same time period and resides naturally with video cameras. It is now trivial to litter an environment with video-capture devices because the costs and wiring complexity have been nearly eliminated.

These devices can be used to create virtual office environments as well as a new source of data for real-time business optimizations, including the observance of shoppers’ buying habits, the tracking and inspecting products, and controlling automated manufacturing processes. Advances in image-processing algorithms now enable computing networks to actually understand scenes.

The Matrix Worker

We are all subject-matter experts in a particular area. Skills databases now exist that allow established companies to run even more successfully with far fewer full-time employees by matching the people skills readily available in the marketplace to specific project requirements. The amount of detail put into these databases is stunning. In the engineering world, for example, every skill and realm of knowledge that an engineer develops while working on a project is summarized within the database. It could be a new computer language such as C#, or mastery of a new Java

library, or a new computer platform, or even the use of some end-user application such as an accounting package. These databases are a low-cost, high-value source of human capital, available now to the momentary enterprise. Often these people prefer to work as independent consultants rather than full-time employees. Technology and connectivity have truly allowed a great many of us to work anywhere and everywhere, and at any time. As more and more people allow their skills to be better published and exploited, a new form of professional – the “matrix worker” – will emerge.

VCs Need Not Apply

The ingredients for another wave of new companies are all around us – pervasively all around us. They include new wireless extensions of the wired network and the further exportation of technologies such as XML. All you have to do as an entrepreneur is glue them together to create new types of information that fill an identifiable need. In fact, new-company generation may be so easy to do that VCs may not be required. These new opportunities come about mostly from converging wireless with the Web and letting the imagination run free. ©

About the Author

John Webster is senior analyst and founder of Data Mobility Group. He is the author of numerous articles and white papers on a wide range of topics, including data convergence, storage networking devices and management, and storage services and outsourcing. He is also the coauthor of a book entitled *Inescapable Data – Harnessing the Power of Convergence*, published in April 2005 by IBM Press.

■ ■ ■ jwebster@datamobilitygroup.com

Does Your SOA Include a Persistence Strategy?

You need to consider the nonvolatile along with the volatile

■ Truth be told, traditional approaches to integration are really about keeping persistence at the points, within the source or target systems, and replicating data as needed. However with the use of true services, there is a clear advantage in keeping some persistence at a central-tier, for any number of legitimate reasons. Let's explore this in the context of an SOA.

Indeed, as we become better at building services we need to understand the infrastructure that the services will leverage, including orchestration, security, management, and data, and where those functions need to reside in the architecture. While SOAs are like snowflakes – every one a bit different, it's helpful to understand the requirement patterns that will lead to specific architectural decisions. Keep in mind that once these decisions are made, they are not easy to undo later.

Before we dive into this issue, it's helpful to understand the differences between an SOA and a more traditional application-integration infrastructure. The use of services adds a few new dimensions to more traditional information-oriented integration.

The first difference is the federation of services around the SOA. Services don't exist as clusters in a single server, they run on any system that exposes them, inside and outside of the organization. The creation of an SOA simply allows you to manage, leverage, and orchestrate these services, thereby creating composite applications and orchestrations to leverage their value. This is the core value of an SOA, as well as the enabling standards around Web services.

Thus, you end up with composites or processes created out of services that may exist over a dozen or more different systems, and



WRITTEN BY
**DAVID
LINTHICUM**

as such persistence becomes more complex if done at the points. So, in these types of situations (which are becoming more common), it makes good sense to centralize the persistence for the composites and processes, as well as some supporting services, to a *central data tier* or *central data service*. This data tier exposes a custom schema view or views to the composites, and may also abstract some data at the points as well. This is done to provide a data service to the composites directly, or perhaps by using a data abstraction layer such as data abstraction middleware (e.g., IIS or federated database software).

Second are *performance issues*. If the composites are doing most of the processing, and it's really a center-tier process abstracting remote services, then it makes sense to collocate the data as close to the data processing as possible. This is done for both manageability, reliability, and for performance.

Integrity will also become less of an issue when leveraging this type of center-tier persistence. No need to lock a dozen or so tables when you can simply lock one. Moreover, security becomes an easier process as well, since it does not need to be as distributed.

Third is the *storage and management of transactional data*. We all understand the value of leveraging a message warehouse, or the storage of information flowing between

systems. Having persistence at the central tier allows architects to store transactional information for many purposes, including analysis and integrity management issues (logging). Also, with the new focus on compliance and support of audits, this seems to be a likely place to store that type of information as well.

I would also include this notion in support of state-management information for services, processes, or composites. This includes supporting long-term transactions, or multiparty transactions. Again, the controlling data is maintained at a central, shared tier.

Last is leveraging *centralized metadata*. We all know that we need to understand and manage application semantics. Leveraging central-tier persistence allows the SOA architects to get a better handle on this issue, due to the fact that we can place abstracted and composite data elements at the central tier. Also, this is a prime location for a central repository and for the management of application semantics, perhaps using standards such as the semantic Web.

This is not to say that all SOAs will require a central data tier, but it may be a good idea for most. Again, you have to consider your own requirements. Common requirement patterns to watch for include the need to control metadata, state management, heavy database processing by the composites, and security issues. The data may reside in any data storage mechanism such as a relational database, object, or XML database. The choice is determined by the requirements within your SOA, including accommodation of existing legacy systems and schema management.

The use of persistence within an SOA is an inevitable reality. Thus, those building SOAs today should be prepared to cross this bridge. It's better to cross it now rather than wait for the water to rise...believe me. ☺

■ About the Author

David S. Linthicum (www.davidlinthicum.com) is the author of three books on application integration and SOA, a frequent speaker at industry conferences, and the host of the "Service-Oriented Architecture Expert Podcast" (www.soaexpertpodcast.com).

■ ■ ■ linthicum@att.net

A LIMITED TIME SAVINGS OFFER FROM SYS-CON MEDIA

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

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

RECEIVE
YOUR DIGITAL
EDITION
ACCESS CODE
INSTANTLY
WITH YOUR PAID
SUBSCRIPTIONS

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

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

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

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

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

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



CALL TODAY! 888-303-5282

☐ 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: \$120	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

**SYS-CON
MEDIA**

Contract-First Web Services

Six reasons to start with WSDL and Schema

■ I believe that there is a strong tendency for developers to think of Web services as methods. In reality it is not that simple. A Web service defines an XML conversation between client and server. Yet, when a developer sits down to create Web services he or she usually starts by creating a method.

From that starting point it is easy to get a software toolkit to turn your method into a Web service. By default or with a few settings you can allow the toolkit to define the conversation around your method. This method-first development technique works, especially for small or pilot projects, but it is not the best approach. A contract-first approach results in better long-term development, interoperability, and maintenance.

Convincing a development team to work contract-first is not as simple as it might seem. You have to convince the team to spend more effort on unfamiliar languages like XML Schema and WSDL, and rely less on familiar languages such as Java or C#. With that said, there are good reasons to code your Web service the hard way – contract-first. In this article we will look at six good reasons to develop Web services contract-first instead of method-first. By the end of this article I hope you will see that contract-first is a superior way to create Web services and will save you time and money.

RPC Thinking

Before we delve into the six reasons, I want to establish the difference between method-first and contract-first Web services. We can walk through the basic steps of both styles of Web services creation using a simple



WRITTEN BY
STEVE CLOSE

service. This will allow me to show some code and remove the abstractness from the topic. Forgive me for using an example that has been repeated countless times. If you're an old pro at Web services skip ahead to the "Reason 1" heading. The following code is a Calculator with one method. My preferred weapons are Java and Apache Axis, so all examples will be coded in Java.

Your choice of language or platform should have little effect on which way you will make Web services.

Step 1: Create a method that will become a Web service. The code below provides simple add method that will be turned into a Web service.

```
public int add(op1, op2)
{
    return op1 + op2;
}
```

Step 2: Run this method through a software toolkit that turns it into a Web service. There are many of these toolkits on the market. It might be as simple as adding an attribute, or saving the source code to a different extension. For the rest of this article we will call that machine the Web service toolkit.

Step 3: Deploy it to some Web server. The server will provide HTTP handling and leave you with a very simple Web service. Web service toolkits automatically create XML conversion code and a Web service description or WSDL (pronounced *wizzle*) document. This WSDL document is the key to getting a client to call your Web service and will play a prominent role in contract-first thinking.

Step 4: A client can now obtain the WSDL document and create a client through your Web service. The cool thing is that the client can basically treat the Web service like a regular method call. Although XML is being passed via HTTP following the rules of SOAP, the client can be created very easily and the client doesn't need to know about the XML payload.

Contract-First Thinking

Taking a quick look at contract-first development, we can clearly see the differences between method-first and contract-first. In contract-first Web services you first create the WSDL document. The WSDL might have supporting XML Schema documents as well. As you can see from

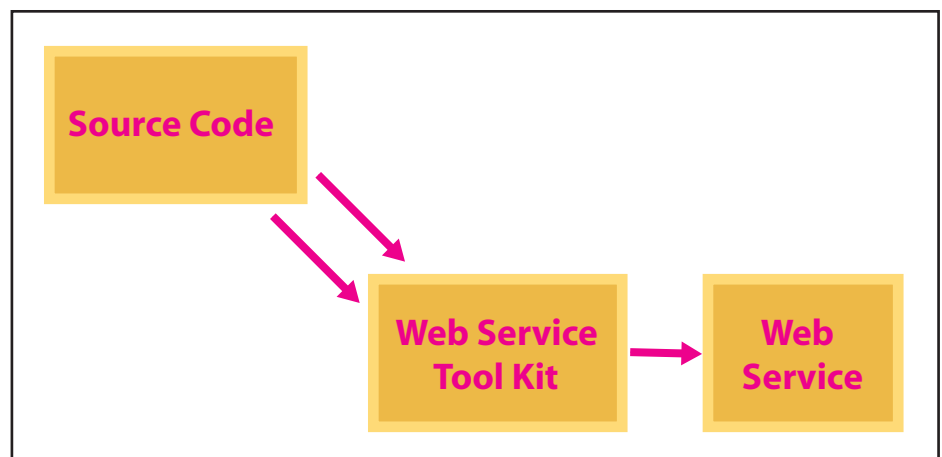


FIGURE 1 Source code becomes a Web service

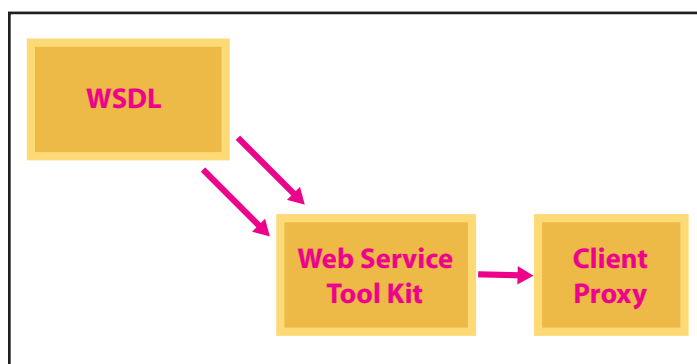


FIGURE 2 WSDL makes clients proxy

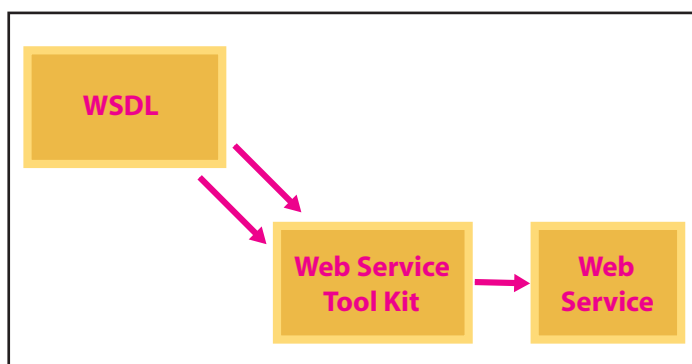


FIGURE 3 WSDL becomes a Web service

Listing 1, the WSDL code is a bit longer and more involved than the simple Java method. In fact, when you look at it in such a simple example, it almost seems ridiculous to suggest that you might write the WSDL before you write the Java code, but that is exactly what I'm suggesting. The steps to create the Web service are similar to what we've already seen, but kind of in reverse.

- **Step 1:** Create the WSDL and supporting Schema
- **Step 2:** Generate the service from the WSDL

It seems my work is cut out for me. The first way seems so much easier at first glance, and for small applications and for learning Web services it is the right way. For larger applications, long-lasting Web services, and service-oriented architecture (SOA), contract-first thinking has advantages that usually outweigh the ease of method-first thinking. The rest of this article we will investigate the six reasons to develop contract-first.

Reason 1: Schema Is More Descriptive

Data types in Java and C# and other languages are described only in code, which is not shared with the client. Clients are generated via the WSDL document. Clients don't have the full definition of the data types unless you include that information in the XML Schema document. As a very simple example, say you are passing in a Person that consists of a firstName, middleInitial, and a lastName. In code this is simple, but some of the data may not be required. Say you want to require the firstName and lastName, but the middle initial is optional. If your WSDL is automatically generated, it is likely to turn out like Listing 2. The middleInitial is created by

default, but with the XML Schema language we can get much more descriptive. With this simple example we can easily make the middleInitial optional by including one little attribute called minOccurs like the code below.

```
<xs:element name="middleInitial"
minOccurs="0" type="xs:string" />
```

I hope this simple example illustrates how schema can be more descriptive, but it is such a tiny example. Imagine a rich set of data, not just a simple person. If you want your clients to understand that the quantity element can only be non-negative integers, or have them understand the pattern that the SKU number must be three letters followed by a hyphen and four numbers, that can be described in XML Schema. By relying on the tool to generate the proper schema, you are passing only a vague description of the data where an e-mail address becomes a string and a month is an integer that can be negative.

Knowing that the WSDL is your client's interface it should be obvious that you want to make it as accurate and descriptive as possible. That might not be enough to write the WSDL first. You could still modify the WSDL after the fact and pass that out to your clients, but we have more reasons to go.

Reason 2: Your Language Is Not Alone

To understand the "we are not alone" idea, we only need to expand the Web service example a bit. You create a Web service and you require that data representing a customer be sent to invoke the service. You may have put some time into the customer type and you realize it should be shared. Other departments also use the same

entity in their services, but wait, here's the problem. You code in C#, they code in Java. You can't just centralize the code. They need to convert it to Java. No big deal, as easily done as said practically, but what if the definition of a customer changes? If the Java coders change the customer your two versions no longer match. Which version is the "central" or "master" customer type? Do I need to change the Java code every time I change the C#? How do you communicate the change to the other team?

The answer is really very easy. Neither the Java nor the WSDL type is the master or center. The center is the XML Schema. The Java and C# can both be generated using the WSDL. The schema is the interface between you and the other Web service developers, so you make that the master. The WSDL is also the interface to the client. The relationship between client and service is defined in these WSDL and schemas so they are by definition the master documents. Think about other distributed technologies. Nearly all distributed technologies define the communication between client and server with an interface. Normally the interface is designed first. Following other distributed technologies, the interface will be first for Web services too.

To get decent reuse of XML Schema, you will need to remove the reused schema types from the WSDL document and store them separately. This allows the definition of type to be shared between development groups and these types will need to be used by many different developers. These types commonly represent corporate entities. How you successfully share these XML Schema so anyone can easily find them I have yet to see 100 percent solved. It's easy when you start – few developers are coding Web services and few types are reused. As it grows it becomes

important to allow users to search for types to make sure they don't re-create entity types.

Reason 3: Change Happens

I know in a perfect world you could create a Web service and it would work wonderfully and we would never have to change it. However, is that how it really works? Change is a constant in the software world, but that can cause trouble for Web services because they are distributed. Take a customer type, for example. Originally the schema might include a name and customerID. Let's say that in version 2 the customer needs to include a phone number as well. So we go ahead and make the change, right? Go to your Java code and change the customer type, why is this such a big deal? Good development teams are experts at versioning their source code and managing releases, but you must realize that this is different... it's distributed.

This simple change is a big deal because your customers are using that Web service. Customers have been using this service for quite a while. Your newly updated service requires additional information and the service won't work for any of the current customers. How to deal with change becomes serious business. Really it always has been, but it is very easy to ignore the fourth dimension because you don't see the cost. Maybe with a small service you could just get on the phone and tell all your customers, "On

November 4 we will be changing the service, please change your client code as well." In larger Web service applications, your clients might be outside of your four walls, or there might be hundreds of them, or you might not know how to contact them at all. You can't expect them to walk lockstep with you from release to release.

Why is Schema better at change? You have more control, which returns us back to the same theme. You can try to avoid changes that aren't compatible with existing clients. Add the phone number in, but don't make it required, make it optional instead. In this backwards-compatible change, existing clients can keep working. If you need to make a nonbackwards-compatible change, make a new schema. XML Schema typically puts a version number in the target-namespace. With major or nonbackwards-compatible changes you create a new schema, but leave the old one in place. The new schema gets a new targetNamespace, which is why you see dates in targetNamespaces. You can support the old Schema for as long as you need to so your clients can change to the new service when their development schedules allow it. In some businesses you can't ever stop supporting an existing Web service.

On a related note, another thing that can change and cause trouble is your WSDL-generation tool. What happens when you move from version 2.2 to 3.1 of your beloved Web service

toolkit? Does the WSDL change too? This kind of accidental change in WSDL may cause client-side tools to generate different messages based on the different WSDL documents. It's not hard to imagine talking to a client who reports that "It worked fine last time, but now my toolkit bombs out. Did your WSDL document change?" At the very least, after your Web service toolkit creates the WSDL you should treat that WSDL document as source code. Don't allow your toolkit to regenerate it. Regeneration is time-consuming and may cause variations to your clients.

Reason 4: The Web Service Is More Than the Method

Most Web service toolkits allow you to add "Intercepting Filters" to your Web service. Maybe your Web service works, but when deployed it requires you to decrypt the message that is sent in. Instead of adding encryption to each service individually, you create a filter that does the decryption and apply it to any number of services. In Axis these are called "Handlers," and in .NET they are called "Web Extensions."

Why is this related to the WSDL? Well, the Web service is no longer just that simple method we created. It is a combination of that method and all filters that are configured to run before the method is invoked or after the method has completed. Really the method no longer contains all of the information needed to gener-

Listing 1: Add Web Service WSDL

```
<wsdl:definitions
  xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://steveClose.com/2004/"
  xmlns="http://steveClose.com/2004/" >
  <wsdl:types>
    <xs:schema
      targetNamespace="http://steveClose.com/2004/" >
      <xs:element name="add">
        <xs:sequence>
          <xs:element name="op1" type="xs:int" />
          <xs:element name="op2" type="xs:int" />
        </xs:sequence>
      </xs:element>
      <xs:element name="addResponse" type="xs:string" />
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="AddMessage">
    <wsdl:part name="addInput" element="add" />
  </wsdl:message>
  <wsdl:message name="AddMessageResponse">
    <wsdl:part name="addOutput" element="addResponse" />
  </wsdl:message>
  <wsdl:portType name="AddInterface">
    <wsdl:operation name="add">
      <wsdl:input message="AddMessage" />
      <wsdl:output message="AddMessageResponse" />
    </wsdl:operation>
```

```
</wsdl:portType>
<wsdl:binding name="AddBinding" type="AddInterface">
  <soapbind:binding style="document" transport="http://
schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="add">
    <soapbind:operation style="document" soapAction="add"/>
    <wsdl:input>
      <soapbind:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soapbind:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service name="CalculatorAdd">
  <wsdl:port name="calculatorAdd" binding="AddBinding">
    <soapbind:address location="http://temporarilyUnknown.
com"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Listing 2: Autogenerated Person Schema

```
<xs:element name="add">
  <xs:sequence>
    <xs:element name="firstName" type="xs:string" />
    <xs:element name="middleInitial" type="xs:string" />
    <xs:element name="lastName" type="xs:string" />
  </xs:sequence>
</xs:element>
```


ate the WSDL. A filter might handle a required header for the SOAP message, but any WSDL defined by just the method won't include this required header element. Clearly at the very least you will need to manipulate the WSDL by hand to add the required header element.

Reason 5: SOAP Faults Are Booleans

SOAP faults are well designed. They are simple yet you can add in flexible details if you wish. SOAP faults are a built in such a way to report errors when a Web service fails, but they are limited. SOAP faults and nonfault data are mutually exclusive. In other words, you can't return a SOAP fault and some real results at the same time. This makes the SOAP fault simple, but not that flexible. Maybe you want to include a partial success, or report a warning with the real response information. To do this you design the message to carry that information. You design your own warning object and design the message to allow the warning and the normal return information. Notice the phrase "you design the message." This is the essence of contract-first thinking. Instead of making a method and retrofitting it with some XML converters, you design the messages that are sent from the user to the service and back again.

Reason 6: Standards Are Written for WSDL

So you want a Web service that works in a heterogeneous environment? If you are thinking SOA you should. Even if today all your developers work in Java, what will they be using in 2012? I'd like to think that my choice of Java will still be there, but what if you get a new CIO? I'm a trainer and we often see business when a client decides to switch from .NET to Java, or vice versa. The direction of a whole IT staff can change with the shaking of a hand – so make your Web services last. Create them to be invoked by any toolkit, any language, and any platform. To make these services work with any platform or Web service toolkit (interoperable), it's best to follow the most accepted standards such as those set up by the Web Service Interoperability group (WS-I).

Contract-first makes following standards easier because all of the standards are written for the WSDL description and SOAP message. Your toolkit might make it easy, or it might make it hard to get the code to generate compliant WSDL

documents. All toolkits are moving toward better standards support, but you can achieve it more easily by using contract-first techniques. Create the WSDL and you can make sure it matches the standards. When you design the SOAP message, design one that matches the standards. I've found it easier to create standard WSDL than to get any of the tools to make it. As an added plus there are great tools for checking to see if your WSDL is standard. Check out SOAP Scope from www.mindreef.com if you are looking for one.

Conclusion

I don't want to leave you thinking that the contract-first way is really the hard way. I made light of that earlier in the article and when you first start it might seem like it. With a little learning, contract-first gets much easier. The trouble seems to be twofold. First, many developers know a programming language much better than XML, XML Schema, or WSDL. This creates a steep learning curve to climb. People avoid that work when under tight deadlines. Second, for tutorials and first exposure to Web services, WSDL and Schema are very tough to grasp. It is usually better to teach a beginner using method-first thinking, until they get the concepts anyway. Keep in mind, you can still start with the code, and just edit the XML Schema and WSDL.

I highly recommend tracking the WSDL after it is generated. If you can do that, reason #1 and reason #6 are partially removed. In addition, after you have one successful WSDL, making another is actually very easy. You end up replacing one or two dozen strings and creating a new XML Schema document. I've found that the WSDL is actually the easy part. Listing 1 might be a bit long for an add method, but it really becomes

easy to read and edit with experience.

Maybe I've convinced you and maybe not. I find that many already use the technique, but most I talk to think it is a pain. In writing this I was hoping to create an easy-to-understand list of why contract-first is popular and why it is a good choice. For those of you who already use it, hopefully you have a bit more ammunition as to why it's a good thing. For those of you who don't, I hope I've gotten you thinking. Keep these six reasons in mind and see if any one reason is important enough to make you switch. ☺

About the Author

Steve Close is the lead Java instructor for Intertech Training (www.intertechtraining.com). He helped found the Twin Cities Java User Group and served as president from 1997-2000. He has authored many popular workshops for Intertech Training on topics ranging from Java 101 to J2EE, and Java Web services. He has presented at JavaOne, SDWest, and No Fluff Just Stuff. He currently focuses his work on Java, XML, and Web services course development and training.

■ ■ ■ sclose@intertechtraining.com

WSJ Advertiser Index

Advertiser	URL	Phone	Page
Forum Systems	www.forumsys.com	801-313-4400	3
Gartner	gartner.com/us/adea	800-778-1997	25
IBM	www.ibm.com/middleware/flexible		5
Intermedia.net	www.intermedia.com	888-379-7729	6
ISSJ	www.ISSJournal.com	888-303-5282	19
IT Solutions Guide	www.sys-con.com	888-303-5252	35
KaPow Technologies	www.kapow.com	800-805-0823	Cover III
Mindreef	www.mindreef.com/tryout	603-465-2204	9
Parasoft	www.parasoft.com/products	888-305-0041	Cover II
Skyway Software	www.skywaysoftware.com	813-288-9355	Cover IV
Smart Data Processing, Inc.	www.weekendwithexperts.com	732-598-4027	30
SYS-CON Website	www.sys-con.com	888-303-5282	31
SYS-CON Subscription	www.sys-con.com	888-303-5282	41
WebAppCabaret	www.webappcabaret.com/dj.jsp	866-256-7973	23
Web Services Journal	www.wsj2.com	888-303-5252	37
XML 2005	http://xmlconference.org	703-837-1070	11

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.

Leveraging Your Host Systems with Web Services

Transforming legacy applications into SOA-based composite applications

■ The promise of service-oriented architecture (SOA), increasing benefits of Web services, and continued business use of legacy systems have all coalesced to usher in a new era of flexible IT alignment for business needs. With only a fraction of legacy applications being Web-enabled and increasing enterprise investment in SOA, organizations need to know how they can use Web services to leverage their legacy systems.

Web services can be used to leverage host systems and enable the transformation of legacy applications into Web-enabled composite applications. Such applications can better serve the enterprise by allowing a tight coupling of business needs with available IT resources.



WRITTEN BY
**FARSHID
KETABCHI**

and mobile access can clearly provide a competitive advantage.

Besides host systems (e.g., CICS and IMS running on mainframe), most organizations may also have ERP and CRM applications such as SAP and Oracle running on UNIX or Windows. There may also be custom applications running on applications servers and middleware. Therefore, a new application may very well require access, aggregation, and composition of data and logic from multiple applications

running on a heterogeneous infrastructure.

One solution is to use SOA principles to transform legacy applications to service-based *composite applications* that also integrate with other applications. Web services are one of the key enabling technologies in achieving this transformation.

Introduction to Web Services

Web services are self-contained, self-describing, modular application components that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. Once a Web service is deployed, other applications (and Web services) can discover and invoke the deployed service.

A Web service message is basically an XML document wrapped in SOAP that is sent across an HTTP/HTTPS connection. There is a Web Service Description Language (WSDL) file associated with each Web service message type that tells the application what to do with the content. The structure of the Web service components is exposed using the WSDL. There may be an entry for this Web service registered in a UDDI, but there doesn't have to be. In summary, XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available, and UDDI is used for listing what services are available.

A Web service exposes the business objects to SOAP calls over HTTP/HTTPS and executes remote function calls. The output is rendered as XML and passed back to the consumer. The consumers of a Web service can invoke method calls on remote objects by using SOAP and HTTP/HTTPS over the Web. Figure 1 depicts a Web service accessing a mainframe.

Web services do not typically have a GUI;

Background

The majority of the Global 2000 run their business on host systems. Various analysts estimate that from 60 to 90 percent of company data is stored on host systems. Many of these systems run key, mission-critical business applications that the organizations heavily rely on; however, making changes to these back-end applications can be difficult and costly.

As companies move forward and extend their technology investments, they can benefit from the reuse of business logic and data from host systems. Additionally, for those applications that require user interaction, browser

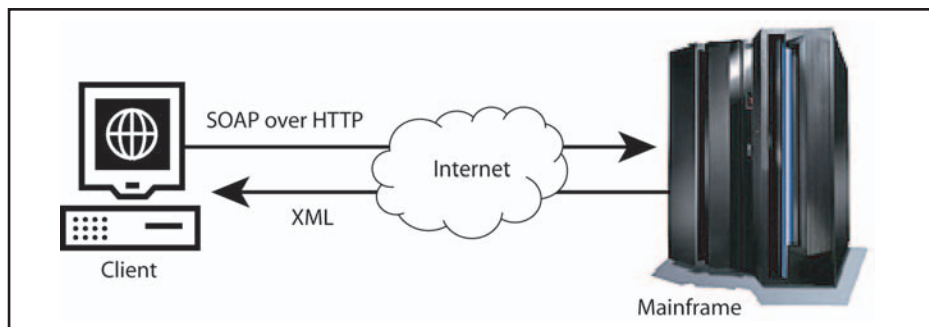


FIGURE 1 Accessing a mainframe via a Web service

instead they share business logic, data, and processes through a programmatic interface across a network. The application developers can then add the Web service to a GUI (for example a Web page) to offer specific functionality to users.

Benefits of Using Web Services

The big advantage of deploying Web services is that they allow different applications from different systems to communicate with each other without time-consuming custom coding. Because all communication is in XML, Web services are not tied to any one operating system or programming language, which means that Java can talk to Perl, or UNIX applications can talk to Windows applications.

Applications typically use multiple Web services in combination to solve specific business problems. Examples include processing an order, updating multiple customer records from a single change request, or scheduling just-in-time arrival of parts for a manufacturing run.

For businesses, adopting an unproven new technology contains risk; there must be enough cost justification and business value in using it. Using Web services provides benefits to the business as well as to the IT department that supports it.

The advantages of using Web services are:

- To use a Web service one only needs to know its public interface and nothing about its implementation
- Use of "open standards" means no proprietary technology lock-in
- Transactions from back-end systems can be exposed as Web services
- Web applications and business services can be developed quickly
- IT cost is reduced through reuse of hardware and software

Application of Web Services to Legacy Systems

Organizations have a need to access data from various host applications and present it over the Web in a usable form to their users. They may also need to allow access to the back-end transactions over the Web for the authorized users. In this section we look at examples of using Web services with back-end legacy systems.

Consider a call center where a customer calls for an inquiry or service request. A customer service representative (CSR) answers the call and looks up the customer information in a legacy system where a CICS transaction accesses data from a DB2 database. The CSR then views the customer information in a 3270 screen, for example, and answers the customer inquiry or provides the customer with the requested service.

The CICS-to-DB2 transaction can be captured as a Web service and used in a secure self-service Web application. In addition, it is possible for customers to use their browsers to access the back-end system information in a simple user-friendly Web page. Now the customers can access and view the information whenever they want and as often as they want without having to talk to a CSR. The use of Web services and the Web itself lowers the cost of customer service and increases customer satisfaction.

Let's now consider a scenario where some of the needed data is stored in DB2 while other data is in flat files, perhaps a PDS (partitioned data set) or a sequential dataset. Furthermore, some of the data is accessed by a CICS system, and some by IMS. In this case, two applications are being used on the same host system to produce the required results. By using Web services, the data is retrieved from the two data sources and

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

FOCUS: .NET

SOA Security: The Only Thing We Have to Fear is Fear Itself

As organizations move to service-oriented architectures (SOA), security becomes one of the key concerns impacting deployment. After all, a company's most sensitive information is frequently stored in the business systems that are now being accessed by the Web services employed within an SOA. As such, security concerns have become part of the enterprise decision-making process relating to the adoption of a SOA.

Ten Things Your EII Vendor May Not Tell You

Enterprise Information Integration (EII) is getting a lot of hype these days, with the vendors giving you very compelling reasons for why you need an enterprise data access layer based on their EII product. However, a lot of their scenarios and case studies present very targeted or simple examples of how their product works without going into the complexities of a real-world environment.

Building Flexible Business Processes Using BPEL and Rules

Efficient business processes are one of the main competitive differentiators for any successful company. These processes orchestrate interactions among systems, services, people, and partners to achieve key strategic and operational objectives. The definition and flawless execution of business processes enable an organization to provide quality products and services, reduce costs, improve customer service, and react quickly to changing market conditions.

Leveraging Your Mainframe with Web Services

The promise of service-oriented architecture, the increasing benefits of Web services, and the resurgence of mainframes have all coalesced to usher in a new era for flexible IT alignment of business needs. With only 5 percent of mainframe applications being accessed by a browser and increasing enterprise investment in Web services, large and small companies need to know how they can use Web services to leverage these legacy systems.

WebServices 
JOURNAL

composed into a single Web page that the customer can view in a browser without knowing where the information is coming from. This is an example of a simple composite application, where data from multiple sources is aggregated and composed and presented in a single view (see Figure 2).

It may very well be that the data resides not only on different applications, but also on different host systems, potentially owned and run by different companies. This is illustrated in Figure 3.

Again, as far as customers are concerned, the application is simple to use. By using a browser the customer can request information, which results in transactions that run on

different computers. The responses from those transactions are consolidated and displayed on a Web page. There may be different Web services that retrieve the data from each application and then the data is composed, perhaps using yet another Web service, into a single record that can be rendered within a single Web page. This is an example of a composite application.

In the aforementioned examples we looked at scenarios in which a customer retrieves and views specific data, but has no ability to update the data. Let's suppose that in our first example a customer wants to confirm the delivery address that a company has for him. He can make a simple query through the browser,

which causes a CICS transaction to access the appropriate data. The data is then displayed in the browser. If the customer then decides that the address is wrong, he can update the information and submit the correct delivery address. This then causes another transaction to run that will update the company's database. This will obviously require security procedures to be in place to ensure that only authorized people can update the database. Although this is similar to our first example, there are now two transactions taking place – one to read the data and one to update the data – both within the same CICS system.

The next level of complexity may be to allow the user to update information that requires two or more transactions to run and act on data from different applications. In Figure 2 information was obtained from CICS and IMS. Now it is a requirement that in addition to a read operation, a write operation will take place on those two different applications on the same host system.

Similarly in our third example, a customer may make a change to information that is updated on two different computers, perhaps owned by two different companies. Refer to Figure 3 again; the customer is able to perform a write operation to those two different applications on two separate host systems.

Here we discussed customers accessing and updating information in back-end systems using a Web browser. However there are many applications such as field service, logistics and transportation, and insurance where access to back-end systems over mobile devices can provide a significant business value and competitive advantage. The Web services that are accessing back-end and host systems can easily be consumed by applications on mobile devices.

Service Orchestration and Composite Applications

Web services are more often consumed by other applications than they are directly mapped to Web pages. In fact an application may use and consume many Web services in the context of a business process. That is, a set of Web services may be *orchestrated* (linked) to form a business process. This is an example of a general composite application that is made up of reusable components such as Web ser-

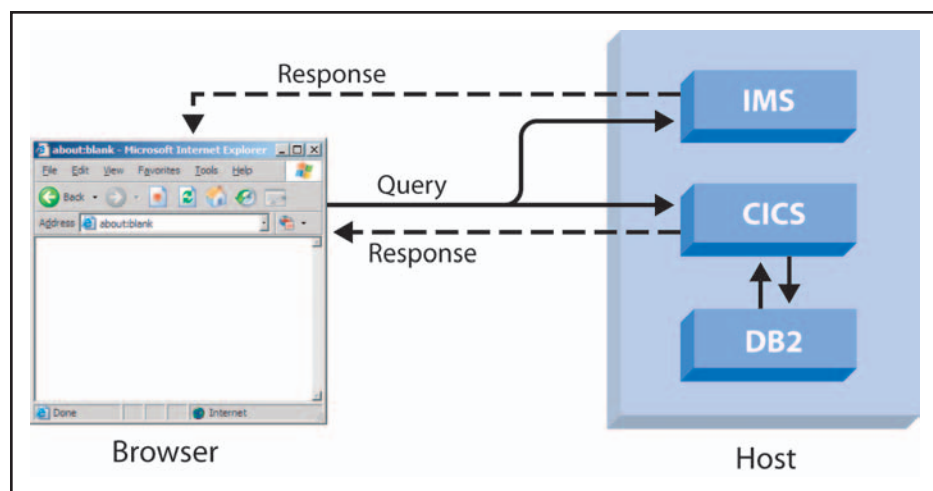


FIGURE 2 Accessing legacy data from a Web application via Web services

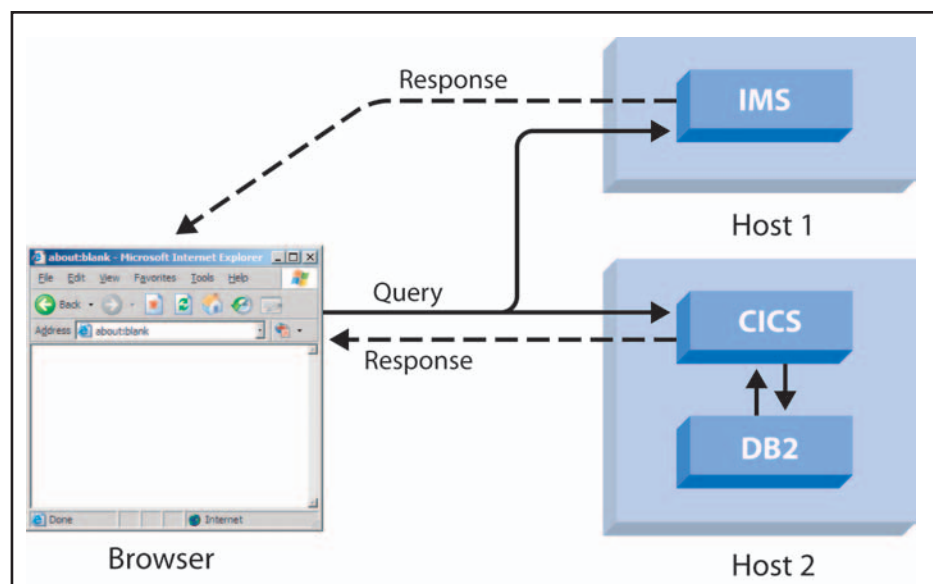


FIGURE 3 Accessing legacy data from different host systems via Web services

“ Composite applications allow organizations to create new business services while leveraging existing systems without requiring changes to those systems ”

vices and that is integrated and orchestrated to explicitly implement a business process.

Many organizations will find that the development of composite applications is where they will get the biggest benefits from using Web services. Composite applications allow organizations to create new business services while leveraging existing systems without requiring changes to those systems. This results in a lower IT cost, as well as more agility and responsiveness, to rapidly changing business environments.

Building composite applications consists of defining reusable components such as Web services from hosts and other back-end systems. There are special tools that enable users to capture and generate Web services from various back-end transactions. Over time there may be a large number of Web services that can be stored and managed in a *service repository*. Once the necessary services are available, they can be orchestrated to form a business process. There are special service orchestration or process modeling and design tools that allow the graphical definition of business processes. The build process is then deployed on and executed by a process engine such as a BPEL (Business Process Execution Language) server or a BPM (Business Process Management) engine.

Composite applications should be standards based and should run in a J2EE environment such as a J2EE application server or in the Microsoft .NET Framework. Security and reliability are also key concerns for Web services and composite applications. There are evolving standards that are covering these aspects such as WS-Security and WS-Reliability that must be taken into consideration.

Let's consider a supply chain management (SCM) application that is used for building new products. The parts inventory information is stored in several applications and databases in various back-end systems. Assume that we want to build 10 units of product P, which requires widgets W1 and W2, which are maintained in different databases. The application checks the back-end systems to ensure that there are enough widgets in stock for 10 units of P. Then the system estimates the delivery date based on the product, parts availability, and other data such as inventory location. If there are not enough widgets in stock, more parts must be ordered from the suppliers, which results in other back-end and external transactions being started. Obviously the product delivery date in this case will be delayed and the inventory databases are then updated accordingly to reflect the used widgets. The order may also have delivery

addresses and customer data that must be verified and updated.

Such a process is made up of many transactions and activities across various systems – all to be performed in concert. Each activity may be performed by a Web service in the context of a supply chain business process. There must also be transaction integrity so that if for any reason things change or an error occurs, transactions can be rolled back and data can be restored to a consistent state.

Figure 4 shows a high-level partial view of what our SCM composite application may look like. The application consists of a set of services (possibly but not necessarily Web services), a business process that orchestrates and links the needed services together, and a presentation layer. The services are generated from the back-end systems. For example, “Create Supplier Order” is a Web service generated from a supplier relationship management (SRM) system, and it is consumed in the process.

The partial process checks for inventory in an ERP, updates it, and then schedules production in an SCM. If there are not enough parts in stock, it places an order for those parts with its suppliers. It then proceeds to create a customer order and schedule it for shipment. Note that an activity such as creating a supplier or customer order itself may be a complex business process made up of multiple activities and services that are encapsulated as a reusable composite Web service.

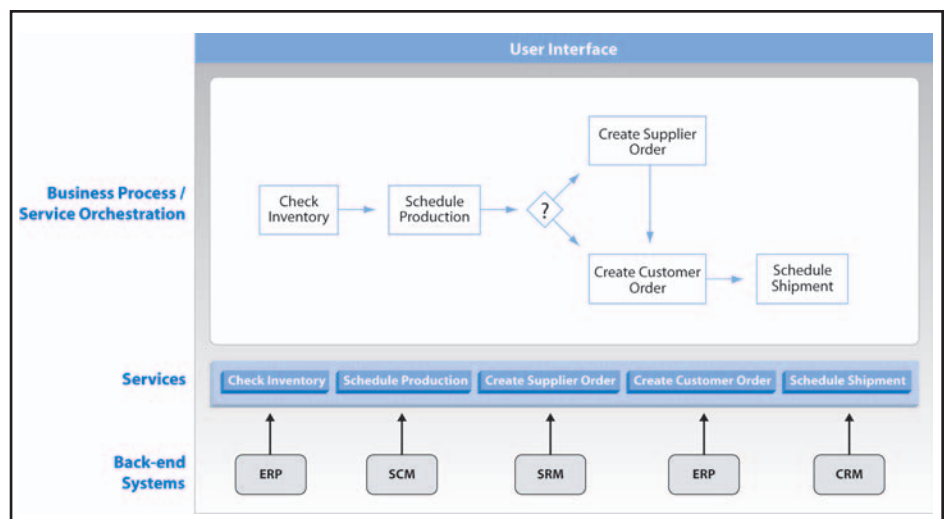


FIGURE 4 A composite application made up of services in a business process

“ Transactions from legacy systems can be captured as Web services and composed into business processes via service orchestration ”

Solving Real Customer Problems

Let's look at some real-life examples of how companies have used Web services to solve business problems.

An Italian company, CartaSi, which provides the Italian banking system with an international payment card system, wanted to “webify” its mainframe applications. The company currently has 7.5 million credit cards outstanding and services 16 banks in Italy. In order to provide superior customer service, CartaSi was seeking ways to streamline their busy customer contact center and provide self-service opportunities to their customers. They decided to transform the information stored on their mainframe systems into easy-to-use interfaces through user-friendly, HTML-based views, instead of the traditional green screens.

Embutidos Estevez, a meat processing and distribution company, is a good example of a company whose legacy applications were not on a mainframe (AS/400s in their case). They are running their Web services on browsers on mobile devices. The AS/400 sales order entry application can be accessed by PDAs via the Internet. The software intercepts the 5250 data stream, transforms it, manipulates the data, and converts it into custom-designed, Web-based presentations. The generated Web pages are available for end users through a Web application server and can be accessed by a Web browser on an Internet-enabled PDA device. The new Web-enabled host application works with PDA devices linked through a GPRS connection. The sales force now submits a sales order as soon as the customer places it, dramatically increasing order processing speed and minimizing data entry errors. In addition,

the PDAs give sales teams spreadsheets, date books, a graphical product catalogue, as well as access to customer information while away from the office.

Tight Coupling of Business Needs and IT Resources

The aim of a business-driven IT organization is to provide optimum IT service to the business users. IT projects based on Web services and SOA are intended to reuse the existing corporate investments in back-end systems and applications in a non-invasive manner without any change and impact in the day-to-day operation of all of the mission-critical systems.

Creating Web services from back-end systems involves a number of stages. The first is identifying functionality from those applications that can be captured as reusable services. The next step is to decide what data is to be shared as well as the associated XML Schema. The next stage is to construct the service description and WSDL. There may also be a central services repository so that the newly created service can be published and made available for various projects to utilize it. The more data that is reused, the higher the value obtained from that service.

When creating Web services from back-end systems, there must be a consistent way to connect to all host and non-host systems to ensure that transactions and information from a wide range of enterprise systems can be accessed and leveraged in creating new applications. These include enterprise applications such as SAP, PeopleSoft, JD Edwards, Siebel and Oracle Applications, as well as middleware and databases such as IMS, WebSphere MQ, Sybase, Informix, and

CA-IDMS. The accepted way of getting to these systems is via standards-based adapter technology such as J2EE JCA (Java Connector Architecture).

In addition to adapters for enterprise applications and databases, there is also a need for access to and reuse of application logic and data that resides in the host systems. This requires specialized programmatic integration technology that provides access to screen-based transactions of such host systems. Such host screen-based transactions can be mapped to modern Web pages or they can be captured as reusable services.

Conclusion

It is imperative for organizations to be able to leverage business data and logic from host systems in new applications and transform host-based applications into new applications. SOA provides the blueprint for such a transformation. One of the key technologies for achieving SOA is the use of Web services in capturing the useful functionality from the back-end systems in the form of reusable services and building composite applications from such services.

Host integration technology and Web services open up a window to the legacy world. Web services can Web-enable host applications. In addition, host-based Web services can be used in building more complex and process-based composite applications using service orchestration.

The principles of SOA, along with host-integration technology, can reduce IT costs and make the IT organization more agile and responsive to business needs. This in turn can improve the bottom line and increase customer satisfaction and retention. All this can be done in a noninvasive way for mission-critical operations, without impacting the back-end systems on which businesses rely. ©

About the Author

Farshid Ketabchi is a senior product manager at NetManage who is responsible for the NetManage Librados adapters and composite application solutions. He has 13 years of experience in application development, product marketing and management, and business development in enterprise software such as BPM. He holds an MS in Computer Science from the University of Minnesota.

■ ■ ■ farshid.ketabchi@netmanage.com

This Month

Transforming Large XML Documents

Indroniel Deb Roy

With the evolution of XML, the XSL standard also became very popular for transforming XML data to XML, text, PDF, etc. However there are some limitations to the XSLT transformation. Today's XSLT processors rely on holding input data in memory as a DOM tree while the transformation is taking place. The tree structure in memory can be as much as ten times the original data size, so in practice, the limit on data size for an XSLT conversion is just a few megabytes. As a result it can only handle XML documents with moderate size – to be processed as the full input, DOM needs to be in the memory for any XSL transformation.

Preventing XML Problems

Dr. Adam Kolawa

Since its inception XML has at times been seen as the cure-all for every problem related to Web applications and integration projects. However, poorly written XML can either slow down an integration project, or worse, cause the integration project to collapse.



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



Transforming Large XML Documents

An alternative to XSLT



WRITTEN BY
Indroniel Deb Roy

With the evolution of XML, the XSL standard also became very popular for transforming XML data to XML, text, PDF, etc. However there are some limitations to the XSLT transformation. Today's XSLT processors rely on holding input data in memory as a DOM tree while the transformation is taking place. The tree structure in memory can be as much as ten times the original data size, so in practice, the limit on data size for an XSLT conversion is just a few megabytes. As a result it can only handle XML documents with moderate size – to be processed as the full input, DOM needs to be in the memory for any XSL transformation.

This major shortcoming of the classical XSLT transformation may be solved with the schema-based transformation API discussed herein. This method uses a stream-based approach to load parts of the document in the memory at one time to proceed with the transformation process. So at any given point of time enough resources are available for the actual transformation process to complete. As classical XSL transformation can transform an XML file to any format viz. XML, text, EDI, EFT, PDF, etc., this approach is restricted to only XML-to-XML transformation. In this approach XML schema plays a pivotal role. As we all know an XML schema can describe the data structure, hierarchy, and validation rules for any XML file. So in this approach, transforming a source XML to another destination XML format is based on describing an XML schema for the destination XML file with control attributes defined in the element definition to aid transformation, and using full XPath APIs to carry out the actual transformation. This approach provides a scalable, stream-based way to transform XML to XML and ideally can handle input of any size, which is impossible to obtain with today's XSLT transformers. This approach can be successfully implemented to transform XML exclusively in the B2B DOMain, where a target schema is always present to validate the generated XML target XML.

The Problems Faced in Classical Transformation

To transform a big XML DOM from one form to another entails a lot of problems using classical XML transformation, as the full input DOM needs to be loaded in the memory. For a huge XML input file, just loading the XML file might fail, given limited system resources. There is no published solution to date to tackle such an issue. There were efforts in this regard to serialize the DOM in permanent storage, to free up memory to get the transformation completed, but that gives rise to several I/O issues. Even the simplest of transformations using this approach to transform big XML documents takes a considerable amount of time, and hence this is not feasible for enterprise usage.

This approach considers all of the complexities of transforming large XML files and comes up with a real-time, scalable solution to this whole problem. Apparently there are no performance bottlenecks in this approach because it's schema-based and works on some basic rules, as defined in the subsequent sections.

“The success of transformation depends upon the available system resources, but passing a very large document might choke the full system resources, and it's not feasible to pump up the system resources every time to get a transformation completed”

The Approach in Detail

This is a schema-based approach to transform a source XML to a destination XML. The schema-based approach is beneficial because the complete structure of the destination XML could well be populated based on the schema definition, and then populating the bare XML structure with the required values into the target XML document might be done with the help of attributes and annotations defined in the schema definition. Here the source XML will be read in a stream-based fashion to load nodes that match the schema definition, and once a match occurs all of the elements that require that XPath to do all of the transformations to populate the skeleton node will be done. Once all of the nodes in the target XML are populated using the loaded input node, it would be removed from the memory and the next chunk of data as node will be loaded to perform the next set of transformations, until the full input DOM is read.

There is one thing to note: the streaming of the input file, i.e., which node needs to be read from the input depends entirely on the user and has to be declared in the schema definition as control attributes. As the schema provides the structure of the final target XML with special processing instructions embedded in the control attributes, the schema will be queried a number of times to get to the correct structure of target XML and will be populated with data using information provided as qualified attributes from namespace *xmlns:saxTran= "http://oracle.schemaTransform/saxTran."* Table 1 shows the first set of attributes needed to provide the basic stream-based transformation functionality.

There may be a lot of other attributes needed in due course of implementation, but for now these are the most crucial ones anticipated.

Let's take a simple example to illustrate the approach in detail. In this example we select a simple *OrgChart.xml* that shows the organization hierarchy of a company. The basic structure is shown below:

```
<OrgChart>
  <Office>
    <Department>
      <Person>
        <First>Vernon</First>
        <Last>Callaby</Last>
        <Title>Office Manager</Title>
        <PhoneExt>582</PhoneExt>
        <EMail>v.callaby@nanonull.com</EMail>
        <Shares>1500</Shares>
      </Person>
      .....
    </Person>
    </Person>
  </Department>
  <Department>
    .....
  </Department>
</Office>
.....
<Office>
  .....
</Office>
</OrgChart>
```

Control Attribute	Meaning
saxTran:streamNode	The node provided in the saxTran:match attribute will be streamed, i.e., if the match is on an xpath "a/b/c" then the streaming will happen on node "c," so the node "c" will be loaded one by one.
saxTran:match	This attribute matches an xpath from the input source and in return produces an element in the output. Where the xpath qualifies in the input source, an element that conforms to the rules defined in the output schema is added to the output XML.
saxTran:map	This attribute maps a qualifying input node value to a destination node in the output XML.
saxTran:function	This attributes defines mainly aggregate functions, which depend on a set of input nodes to compute. As in this approach, the nodes are read and then deleted from the input tree after the transformation is done; aggregate functions in xpath implementation cannot be used.
saxTran:outputNode	This attribute will be used to stream nodes as output streams either to a file or to any other storing device, when the criteria expressed for the attribute becomes valid. Writing to an output stream will actually replenish the memory taken by the output Dom.

Table 1 • The first set of attributes needed to provide the basic stream-based transformation functionality

After the transformation the resulting document should show all of the persons in all departments and some calculations such as count, average, and summation are done on some fields of the Person element. To achieve this, classical XSL is written and could be found in *personinfo.xsl* file. The basic structure of the document after the transformation is shown below:

```
<PersonsInfo>
  <Persons>
    <Person>
      <First>Vernon</First>
      <Last>Callaby</Last>
      <Title>Office Manager</Title>
      <PhoneExt>582</PhoneExt>
      <EMail>v.callaby@nanonull.com</EMail>
      <Shares>1500</Shares>
    </Person>
    <Person>
      .....
    </Person>
  </Persons>
  <TotalPersons>20</TotalPersons>
  <AvgSharePerPerson>200.0</AvgSharePerPerson>
  <TotalSharesWithPersons>4000</TotalSharesWithPersons>
</PersonsInfo>
```


As we know in classical XSLT, transformation of the full input DOM is loaded in the memory to do the transformation, and so there is a fixed limit on the number of “Person” elements the XSL transformation can handle without going out of memory. The success of transformation depends upon the available system resources, but passing a very large document might choke the full system resources, and it’s not feasible to pump up the system resources every time to get a transformation completed. So with classical XSLT transformation, every system has an optimum limit on the document size, which could be transformed. To get rid of this major shortfall, loading the input source in an incremental way seems to be a viable solution, but this approach cannot be applied to a classical XSL transformation because there is no clue about the structure of the output file.

For XML to XML transformation, there is an advantage of knowing the output format of the file if a schema is present to describe the output file. Often enterprise application where XML to XML transformations are carried out, there is a schema present to describe the output file so that after the transformation the transformed file may be validated against the schema. In such a case where a schema is present to define the output xml data this schema-based transformation may be used to transform a document using an incremental approach of loading the input document defined in control attributes in the schema definition.

Let’s consider the example above to see how a schema based-approach can ideally load the input DOM incrementally and discard the processed chunks of data after transformation to successfully and ideally complete an infinitely large document that provides only resources that are similar to the classical XSL transformation.

The schema for the above output in the example could be found in file “personsinfo.xsd.” Some additional attributes are added to some of the elements in the schema declaration to enable the transformation.

In Listing 1, the basic schema definition is pretty simple. To add the transformation, instruction attributes are added from the name space xmlns:saxTran=“http://oracle.schemaTransform/saxTran” so that instructions could be implemented to carry out the actual transformation. All of the control attributes in the Listing 1 schema definition are shown in *italics*. Right now only few elements that are shown in Table 1 are used, but depending on the complexity of transformation these attributes will increase.

Basic SAX-based Transformation Implementation

Now, let’s go under the hood of a basic implementation and see how an incremental loading of the input DOM is possible using the schema-driven transformation approach.

Figure 1 shows the approach. The schema, which defines the target XML, has special attributes to match and map elements from input to target. So, first from the schema the default output document structure needs to be constructed without any values just according to the schema definition up to the element for which the *saxTran:streamNode* is defined. The *saxTran:match* attribute for that element will tell about the XPath of the input node on which streaming needs to be done. On each occurrence of the input node, a partial DOM has been constructed in the memory. All of the XPath references on the schema definition satisfied by the partially loaded DOM have been evaluated and values are replaced in the already created skeleton DOM from the schema definition. There might be *saxTran:function* attributes that also need the same XPath for the function evaluation; for these cases the value for that XPath is calculated and added to the expression for the function call as an argument. Once the XPath definition in the schema definition is fully satisfied, the node on which the streaming is applied is unloaded from the memory and the subsequent one is loaded for processing. Once all of the references of the matching XPath are dealt with from the input source, then the functions that are there to be evaluated are processed and

the final value is populated in the node containing that function.

In the above example, the node on which streaming is applied is the */OrgChart/Office/Department/Person* node. So, at any time during the transformation process the in memory node will look like the following:

```
<OrgChart>
  <Office>
    <Department>
      <Person>
        <First>
          Vernon</First>
        <First>
          <Last>Callaby</Last>
        <Last>
          <Title>Office Manager</Title>
```

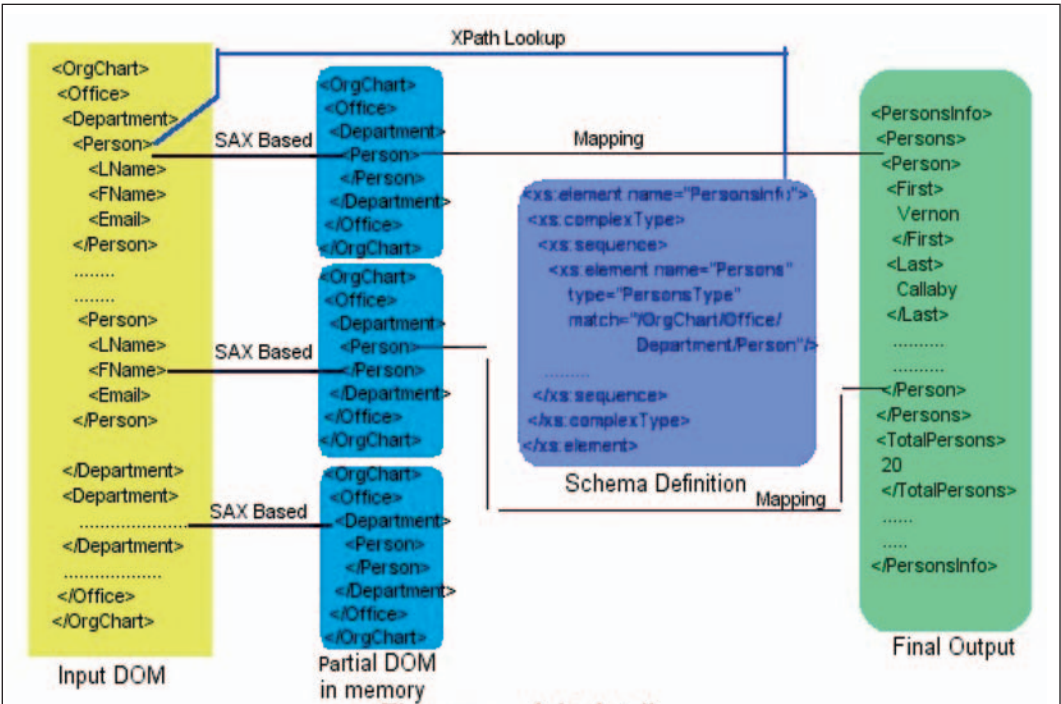


Figure 1 • The approach in detail

```

Title>
  <PhoneExt>582</PhoneExt>
  <EMail>v.callaby@nanonull.com</EMail>
  <Shares>1500</Shares>
</Person>
</Department>
</Office>
</OrgChart>

```

For each *Person* node the values of all the matching XPath like *"/First"/*, *"/Last"/* etc. are populated in the target skeleton structure. For the aggregate functions like the count, sum, avg, etc., the function expression is updated continuously with actual XPath value.

So, for the three aggregate functions, the target node at the middle of the transformation process when three *Person* nodes are done will look like:

```

<TotalPersons><![CDATA[saxTran:count(1,1,1)]]></TotalPersons>
<AvgSharePerPerson><![CDATA[saxTran:avg(1500,0,NaN)]]></AvgSharePerPerson>
<TotalSharesWithPersons><![CDATA[saxTran:sum(1500,0,NaN)]]></TotalSharesWithPersons>

```


So, with each *Person* node repeating, there will be additional arguments added to the functions and populated with actual XPath values. All of the functions will be evaluated when the input source is fully read to populate the final values for these elements.

When using this approach there is no limit to the input file, which can be processed. As the memory is always replenished after processing one stream node, it can handle transformation of infinitely long XML documents without fail. For streaming output DOM too, control attributes might be specified so that it could be serialized after transformation of a required chunk of data. It will provide the flexibility of using any large XML in a transformation process, which was impossible with previous XSLT processors.

Summary

Huge, database dumps or XML coming out of serialized data records can now be transformed effectively with this approach. It can actually augment the classical XSLT engine to provide a fail-proof transformation engine. This is always seen in transforming large XML files using XSLT, and the bottleneck lies in loading the input XML as DOM tree. In most of the cases the XML data repeats for a particular element (a data record) for thousands of times, but loading all of them at once chokes the memory of the transformer. This approach can then augment the classical XSLT engine to stream the input source, and the transformation could be completed without fail. It can also store some of the transient variables within itself to provide the information to the next set of XSLT transformations in the pipeline. This approach will open a new dimension to the world of XML transformation and provide a solution for the impossible task of transforming large XML files. Here I discussed only the approach, so be sure to check back to find the implementation in my next article.

References

- *Specification for the syntax and semantics of XSLT*: www.w3.org/TR/xslt
- *A guide to XSLT transformation*: www.ibiblio.org/xml/books/bible2/chapters/ch17.html
- *Oracle XML technology Center*: www.oracle.com/technology/tech/xml/index.html
- *Oracle StAX (Streaming API for XML) Pull Parser Preview*: www.oracle.com/technology/tech/xml/xdk/staxpreview.html
- *XML Data Streaming for Enterprise Applications Using PL/SQL and SAX*: www.oracle.com/technology/tech/xml/xdk_sample/xdksample_040602i.html 

AUTHOR BIO

Indroniel Deb Roy works as a Software Development Manager for XML Publisher with Oracle. He was previously involved in developing Novell's XML integration server (exteNd Composer) and has worked with various XML and J2EE technologies for the past seven years.

 indroniel.roy@oracle.com

Listing 1

```

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  xmlns:saxTran="http://oracle.schemaTransform/saxTran">
  <xs:element name="PersonsInfo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Persons">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Person" maxOccurs="unbounded"
                saxTran:match="/OrgChart/Office/Department/
                Person"
                saxTran:streamNode="true">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="First" type="xs:string" saxTran:
                    map="/First"/>
                    <xs:element name="Last" type="xs:string" saxTran:
                    map="/Last"/>
                    <xs:element name="Title" type="xs:string" saxTran:
                    map="/Title"/>
                    <xs:element name="PhoneExt" type="xs:short" sax-
                    Tran:map="/PhoneExt"/>

```

```

                    <xs:element name="EMail" type="xs:string" saxTran:
                    map="/EMail"/>
                    <xs:element name="Shares" type="xs:short" saxTran:
                    map="/Shares"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="TotalPersons" type="xs:long"
          saxTran:function="count(/OrgChart/
          Office/Department/Person)"/>
        <xs:element name="AvgSharePerPerson" type="xs:
          decimal"
          saxTran:function="avg(/OrgChart/
          Office/Department/Person/Shares)"/>
        <xs:element name="TotalSharesWithPersons"
          type="xs:short"
          saxTran:function="sum(/OrgChart/Office/Department/
          Person/Shares)"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```



WRITTEN BY DR. ADAM KOLAWA

Preventing XML Problems

Establishing rules and team policies to prevent poorly written XML

Since its inception XML has at times been seen as the cure-all for every problem related to Web applications and integration projects. However, poorly written XML can either slow down an integration project, or worse, cause the integration project to collapse.

When developing integration systems such as Web services or any other business-to-business function, developers may encounter the following problems when writing XML:

- **Non-verifiable code** – XML is supposed to be easily validated by use of Document Type Definitions (DTDs) or schemas. Frequently however, DTDs and schemas may be invalid themselves, too complicated for XML documents to reference, or even insufficient for most businesses. Therefore, there is no way to really guarantee that a certain XML file is valid if it does not reference a valid DTD or schema.
- **Human-readable, yet ambiguous code** – Although human readability can be seen as an advantage of XML, it can also be viewed as a problem. Human-readable code isn't always readable by humans. For example, an element that has a specific meaning to one developer may be of no use, or make no sense, to another developer. Also, human-readable does not necessarily mean machine-readable. If XML code is written strictly for machine consumption, then there is

no reason for having code that makes sense to humans but has no meaning to a machine.

- **Versioning problems** – Maintaining multiple versions of a single document can be very difficult. Developers can either maintain a full version of the code to understand each XML format, or they can reference different DTDs for each format. Both options are possible but each requires a lot of time and effort.
- **Vogue attitude toward XML** – Many developers turn to XML simply because it is the popular language of

If errors are contained in the XML, it is more likely than not that the system will crash

the moment and they do not consider whether or not it is the right solution. More often than not, XML introduces more complexities than needed where a simpler text file would have sufficed.

- **Chaos of standards** – XML standards are still in development and are constantly shifting. Without any stability in XML standards, developers are forced to either keep up with the rapid changes, or fall behind.

Preventing the use of poorly written

XML is more complicated than most developers realize. The key to successfully using XML in an integration project is first understanding the inefficiencies that may cause poorly written XML, and then applying a rule-based system that establishes policies that can be adhered to. This article will outline the many drawbacks of XML, and will address how a rule-based system can prevent the use of poorly written XML in integration projects.

Understanding XML

The Extensible Markup Language (XML) is a family of technologies that describe structured data. By using XML companies can create common information formats and share this information on the World Wide Web. For example, a company can create an XML document to exchange information about its products over the Internet. For a simple example of an XML document, see Listing 1.

XML and Its Inefficiencies

Although the example XML document in Listing 1 appears to be written correctly, how can developers be completely sure that the code is valid and well-formed, is comprehensible to other developers, and adheres to specific standards? The answer to this question lies in a rule-based system that can establish team policies and practices to prevent poorly written XML.

The following sections will outline some of the inefficiencies that can lead to problematic XML, and will address

AUTHOR BIO

Dr. Adam Kolawa is cofounder and CEO of Parasoftware, a vendor of automated error-prevention software and services based in Monrovia, CA. Dr. Kolawa, who is the coauthor of *Bullet-proofing Web Applications* (Wiley, 2001), has written and contributed hundreds of commentary pieces and technical articles for publications such as *The Wall Street Journal*, *CIO*, *Computerworld*, *Dr. Dobbs's Journal*, and *IEEE Computer*. He has also authored numerous scientific papers on physics and parallel processing. He holds a PhD in theoretical physics from the California Institute of Technology.

how a rule-based system can prevent the use of poorly written XML in integration projects. After all, system performance is only as good as the data received and the instructions given. If errors are contained in the XML, it is more likely than not that the system will crash.

Validating XML

One of the main benefits of XML is that it provides mechanisms for verifying document validity. There are two basic mechanisms for verifying document validity: DTD and XML Schema. For example, when creating an XML document developers can reference either of these mechanisms from within the document itself. The DTD or schema that is referenced will specify exactly how the XML document is to be processed, which elements and attributes are contained in the document, and the order in which these elements and attributes should be listed.

Defining DTDs

The following is an example of a simple DTD that can be referenced by an XML document:

```
<!-- ProductList DTD -->
<!ELEMENT ProductList (Product)*>
<!ELEMENT Product (#PCDATA)>
<!-- Product color
    (red|green|yellow|weird) #REQUIRED
    file CDATA #REQUIRED
    id CDATA #REQUIRED
    isFruit (true|false) 'true'>
```

To reference this DTD from an XML document, the following header can be added to the beginning of the XML document:

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE ProductList PUBLIC "-
//OnlineGrocer//ProductList//EN"
"ProductList.dtd">
```

A DTD is a specification based on the rules of the Standard Generalized Markup Language (SGML) and provides basic verification of XML documents. DTDs provide mechanisms for expressing which elements are allowed and what the composition of each element can be. Legal attributes can be defined per element type, and legal

attribute values can be defined per attribute.

Defining Schemas

For an example of a simple schema that can be referenced by an XML document, see Listing 2. To reference this schema from an XML document, the attribute in the element can be specified with the following header:

```
<ProductList xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ProductL
ist.xsd">
```

An XML schema, like a DTD, defines a set of legal elements, attributes, and attribute values. However, XML schemas provide a more robust verification for XML documents. XML schemas are namespace-aware and also cover data types, data bounds, schema class inheritance, and context-sensitive data values – all of which are not covered by DTDs.

Lack of DTD/Schema Enforcement

While referencing DTDs or schemas can guarantee the validity of XML documents, there is no requirement that developers will use headers to reference DTDs or schemas at all. In fact, developers need only to follow simple syntax rules in order for an XML document to be “well-formed.” However, a well-formed document is not necessarily a valid document. Without referencing either a DTD or a schema, there is no way to verify whether the XML document is valid or not. Therefore, measures must be taken to ensure that XML documents do, in fact, reference a DTD or schema.

Using Rules to Enforce Document Validity

To guarantee that an XML document references a DTD or schema, development teams can adopt a rule-based system that can detect and prevent errors within the XML code. Developers can create rules that impose constraints on XML documents to verify validity. For example, a rule can be created that enforces an XML document to contain the sample schema header:

```
<ProductList xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=
"ProductList.xsd">
```

If the document is missing the specified attribute header, an error will occur alerting the developer of the violation.

Human-Readable or Ambiguous Code?

As seen in the sample XML document, XML is human-readable. In other words, XML is created in plain text and utilizes actual words or phrases that have specific meanings to developers. However, even though XML can be read and written by humans, it does not necessarily mean that humans can *understand* XML – developers can still create

While referencing DTDs or schemas can guarantee the validity of XML documents, there is no requirement that developers will use headers to reference DTDs or schemas at all

unreadable XML code. An element that has a specific meaning to one developer may be of no use, or make no sense, to another developer.

For instance, developers can create XML that is completely unintelligible to one another – consider XML tags that are written in Polish or Japanese. Code doesn't need be written in another country to be ambiguous either – ambiguous code written in the same tongue that is to be shared between companies can be quite cryptic as well. For example, the element `<Trans>` can mean anything from transform, transaction, or Trans-Am, depending on the developer and the application.

Establishing Team-Naming Conventions

To prevent ambiguous XML code, development teams must mutually agree upon a standard XML vocabulary. With a standard language in place, developers within a

team will be more likely to understand each other's code.

Naming conventions can be established that verify whether code follows rules that verify anything from W3C guidelines for a specific language, to team-naming standards, to project-specific design requirements, to the proper usage of custom XML tags.

Chaos of Standards

Although the W3C has made an effort to establish a common language, vocabulary, and protocol for XML, these standards are still in development and are constantly changing. Companies that adhere to proposed standards that are not yet fully mature must be prepared to keep up with any changes of the standard in the future. For example, a standard that is in existence today may not exist six months from now. Without any stability in XML standards, developers are forced to either keep up with the rapid changes, or fall behind.

WS-I Basic Profile to the Rescue

In spite of the chaos of standards that

may exist for XML development, the release of Basic Profile provides some guidance to developers seeking a widely used XML standard. The Web Services Interoperability Organization (WS-I) Basic Profile standard consists of specifications that establish a

Developers can now depend on Basic Profile as a common framework for implementing XML and building integration projects


baseline for interoperable Web services. These specifications include guidelines that cover XML 1.0.


Developers can now depend on Basic Profile as a common framework for implementing XML and building integration projects. There are more than 25 WS-I member companies that support Basic

Profile. Therefore, developers can be confident that the XML standards they use will not be privy to constant flux and change.

Summary

Although XML is meant to be a flexible, easy to use, and fully portable solution for Web applications and integration projects, it is not the cure-all that many once thought it to be. The inefficiency of XML is well known among enterprise developers, but it remains ignored in exchange for the perceived advantages of XML such as flexibility, ease of use, and portability. However, the reality of the issue is that XML has a number of drawbacks that enterprise developers should be leery of when creating integration systems.

The key to successfully using XML in an integration project is to first understand the inefficiencies that may cause poorly written XML, and then utilize the proper techniques that verify correctness at each level of the implementation. 

 ak@parasoft.com

Listing 1

```
<?xml version="1.0" encoding="US-ASCII"?>
<ProductList>
  <Product color="green" file="apple_fruit_green.gif"
id="0" isFruit="true">
    Green apples are great for making caramel apples.
    The sour taste
    of the apple and the sweet taste of the caramel
    blend well together.
  </Product>
  <Product color="green" file="artichoke_veg_green.
gif" id="1" isFruit="false">
    Artichoke hearts are the tastiest part of the
    artichoke. When you
    eat the heart, it's love at first bite!
  </Product>
  <Product color="green" file="asparagus_veg_green.
gif" id="2" isFruit="false">
    Asparagus is easy to cook.
  </Product>
  <Product color="green" file="avocado_fruit_green.
gif" id="3" isFruit="true">
    Avocados are the main ingredient in guacamole.
  </Product>
  <Product color="yellow" file="banana_fruit_yellow.
gif" id="4" isFruit="true">
    Smart monkeys eat bananas. Are you smart?
  </Product>
</ProductList>
```

Listing 2

```
<!-- ProductList Schema -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/
```

```
XMLSchema">
  <xsd:element name="ProductList">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Product" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Product">
    <xsd:complexType>
      <xsd:simpleContent>
        <xsd:extension base="xsd:string">
          <xsd:attribute name="color" use="required">
            <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                <xsd:enumeration value="red"/>
                <xsd:enumeration value="green"/>
                <xsd:enumeration value="yellow"/>
                <xsd:enumeration value="weird"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:attribute>
          <xsd:attribute name="file" type="xsd:string"
use="required"/>
          <xsd:attribute name="id" type="xsd:nonNegativeIn-
teger" use="required"/>
          <xsd:attribute name="isFruit" type="xsd:boolean"
default="true"/>
        </xsd:extension>
      </xsd:simpleContent>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

LEVERAGE EXISTING IT TO GENERATE WEB SERVICES

[in minutes]

For
free online
demonstration,
Gartner "Cool Vendor"
release, and software trial:
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

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



(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.