

Web Services

JOURNAL

.NET J2EE XML

September 2005 Volume 5 Issue 9

The "B" in BPM Stands for Business

Differentiating pure-play
BPM from rebranded EAI
and Workflow technologies
pg.26

Enabled SOA Transformation

Leveraging open
source principles
pg.48

Web Services and Enterprise Content Management

Hype vs. reality
pg.52

The Principles Of SERVICE- ORIENTATION

New research reveals
the fundamental dynamics
behind SOA

PG.10



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.



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

The Principles of Service-Orientation

10

New research reveals the fundamental dynamics behind SOA

By Thomas Erl

The "B" in BPM Stands for Business

26

Differentiating pure-play BPM from rebranded EAI and Workflow technologies

By Karl Treier

Enabled SOA Transformation

48

Leveraging open source principles

By Michael Kochanik

Web Services and Enterprise Content Management

52

Hype vs. reality

By Charles Hough

From the Editor Open Wounds

Sean Rhody 7

Industry Commentary Open Servicing

Ajit Sagar 8

WSJ: SOA SOA Testing Framework

A framework to test your services in service-oriented development

Anbarasu Krishnaswamy et al. 20

WSJ: SOA Describing Web Services in 2005

How to make Web services consumable today

Dion Hinchcliffe 30

WSJ: SOA Can Your SOA Make You Money?

Reuse of services + agility = ROI

David Linthicum 34

WSJ: SOA Autonomic SOA

Achieving fully "business-conscious" IT systems

Arthur Mateos & Jothy Rosenberg 36

WSJ: Frameworks Web Service Invocation Framework

Adding wheels to Web services

Anshuk Pal Chaudhari et al. 40

WSJ: Case Study SOA in Action

Experiences with caching, transactions, and security in a highly distributed, networked SOA in the travel and leisure industry

Armughan Rafat et al. 44



The Semantic Organization

MICHAEL WACEY

56



YOU VS APPLICATION INTEGRATION COMPLICATION



WebSphere®

IBM WebSphere middleware is the easy and affordable way to help you overcome virtually any integration challenge. In fact, you can **CONNECT ANY APPLICATION ON ANY PLATFORM WITH OPEN STANDARDS-BASED IBM MIDDLEWARE**. Only IBM has years of proven, trusted experience helping customers build composite applications. The open standards answer to complex application, platform and IT infrastructure combinations, IBM WebSphere lets you re-use your existing IT investments. Imagine increasing efficiencies and making your business more flexible.

SEE HOW AT IBM.COM/MIDDLEWARE/CONNECT

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 wjssecurity@sys-con.com

RESEARCH EDITOR

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

TECHNICAL EDITORS

Andrew Astor andy@enterprisesdb.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

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

Mohamad Afshar, Anshuk Pal Chaudhari, Thomas Erl, Sandeep Gaikwad, Dion Hinchcliffe, Michael Kochanik, Charles Hough, Anbarasu Krishnaswamy, David Linthicum, Rajeev Mahajan, Arthur Mateos, Ravi Nagubadi, V. Niranjan, Armughan Rafat, Sean Rhody, Jothy Rosenberg, Ajit Sagar, Karl Treier, Ambar Verma, Michael Wacey, Markus Zim

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.



Open Wounds

Like many people in the industry, I'm torn over open source software. I'm not opposed to developers creating software and deciding they do it for the love of programming, and have no need for payment – if they want to give their work away, I see no reason why they shouldn't be able to do so, although I think the people who want all software to be free should first get uniform agreement from everyone in the industry to work for nothing before they get on that soapbox. Even though I run a magazine in my spare time, I make my living designing software, and I personally don't want to do it for free.

I'm not opposed to people who want to develop for fun, or for the pure joy of programming. Lots of students in college do this, and many hardcore programmers who don't get enough code during the day seem to grind it out after hours as well.

Eventually though, the economics catch up. Businesses will use whatever they can legally obtain in order to create a competitive advantage, or maintain parity. Even though an application server provides capabilities that would cost millions to develop internally, corporations balk at buying one for tens of thousands of dollars – which is where the open source people come in.

Consortiums such as Apache make it easier for developers who are interested in building a free version of some tool to come together, manage a project, and produce software that is free and useful to the community at large. Linux and Apache Tomcat are probably two of the most useful and successful results of this type of endeavor.

Of course, when you get something for free, it seldom comes with a warranty. This is one of the biggest challenges for open source – the fact that no apparent support structure exists. Corporations that are buying software often look not just at the technical features of a product, but also at the organization's support team and financial



WRITTEN BY
SEAN RHODY

outlook. A troubled software company with a good product can often spiral down because of its financial position, even when they have a superior product. Support is crucial to acceptance of software.

Enter the companies that provide support for open source. Some are new, such as Redhat, and some are existing companies like IBM and Novell, all of whom reap the benefit of the adoption

of open source by providing a security blanket for when things go wrong. In one of the oddities of open source, the developers who contribute their brilliance get nothing, while companies who package free software up and offer support services make tons of money.

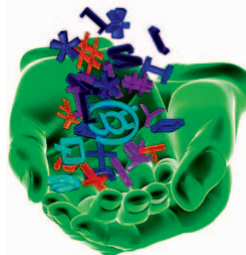
However the biggest challenge to open source remains legal. Intellectual property is a funny thing, and it's hard to separate the TCP/IP stack you wrote at work from the TCP/IP stack you wrote for fun at home. Concepts and ideas coming together and the legal ownership of things can be contested furiously. Also, the risk of being held liable for not purchasing a software license once someone wins a court victory is still a factor that prevents the adoption of Linux and other software in corporations today. It's not a simple world we live in, and free may end up being costly.

Nevertheless, open source software is clearly in use, and it's useful in the corporate world. Many companies have adopted Linux, Apache, MySQL, and other tools that help them reduce their cost of ownership. Things are no different in the world of Web services. Freeware tools abound that make it possible to run a Web services stack without paying any licensing costs. Our focus this issue is on just some of those tools and products that can help you deliver Web services without costing you a fortune – at least until the next lawsuit. ©

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



OR



PRESIDENT AND CEO

Fuat Kircaali fuat@sys-con.com

VP, BUSINESS DEVELOPMENT

Grisha Davida grisha@sys-con.com

GROUP PUBLISHER

Jeremy Geelan jeremy@sys-con.com

ADVERTISING

SENIOR VP, SALES & MARKETING

Carmen Gonzalez carmen@sys-con.com

VP, SALES & MARKETING

Miles Silverman miles@sys-con.com

ADVERTISING DIRECTOR

Robyn Forma robyn@sys-con.com

NATIONAL SALES & MARKETING MANAGER

Dennis Leavey dennis@sys-con.com

ADVERTISING MANAGER

Megan Mussa megan@sys-con.com

ASSOCIATE SALES MANAGERS

Dorothy Gil dorothy@sys-con.com

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@edithroman.com;

Frank Cipolla: 845 731-3832,

frank.cipolla@epostdirect.com

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



Open Servicing



WRITTEN BY
AJIT SAGAR

It seems as though as soon as the open source community rallies around a technology, the IT industry starts taking it more seriously – and finds practical application for it. Ironically, although organizations like the concept, despite the maturation of the open source community in a variety of platforms and technologies, adoption of open source products in large organizations is still an uphill battle. The good news is that mainstream vendor products are now based on a combination of open source technologies, and so mature products from the community are finding homes in many corporations.

One of the greatest values provided by open source is the ability to think outside the “standards” box. We have all seen history repeat itself as standards are proposed, modified, argued about, and change ownership. Some get divided, and some merge. In the end, although technology standards are driven by a consortium, the consortiums are primarily representative of a handful of mainstream vendors with large market shares. This is true in the case of platform standards, such as Java, as well as technology standards, such as XML, and the Web service standards that stem from the base of WSDL, SOAP, and UDDI.

Take the evolution of BPEL (Business Process Execution Language), for example. It has changed names from BPEL to BPEL4WS to BPEL, and now WS-BPEL. Along the way, it has had ownership from IBM, Microsoft, BEA, and other vendors with base application server offerings. Now the standard is owned as WS-BPEL by OASIS. Also during the course of its evolution, it has branched out into other standards like BPELJ. Along the way, open source implementations of BPEL have been made available as alternatives to commercial products that tie you to the vendors’ offering stacks.

The standards in Web services are becoming unmanageable. The combination of standards bodies and vertical standards for WS-Security, WS-Interoperability, WSM, and so on have made it really confusing for organizations to wade through the mire and develop basic services. As a result, the Web services that are being developed in most organizations are not well thought

out, and are on unstable foundations. Besides the complexity of individual specifications, it is not clear how all of these standards will work together to provide a viable technology platform at any point in time. A natural result of this is the incompatibilities between vendor product offerings; often there is a lack of integration between products in a stack offered from the same vendor. Furthermore, on the practical side, it is not clear how Web services that are developed after somehow making standards and products work together will be deployed and managed in a production environment.

This week the Apache Foundation announced a new open source project to address this issue – the Apache Synapse project. Synapse is contributed by WSO2, which is a Web services firm founded by leaders of the Apache Web Service project. And guess where the technology is from? A small island in Asia, south of India, called Sri Lanka. The idea behind Synapse is intended to address the issue of creating something tangible from the quagmire of standards around Web services. Synapse plans to produce a service broker – lightweight and scalable – based on Web services standards. The broker will be developed with contributions from Infravio, Blue Titan, Iona, Sonic Software, and others. Synapse focuses on the implementation of a pure Web services stack, including WS-Policy, WS-Security, WS-ReliableMessaging, and WS-Addressing. Also, Synapse is targeted to enable SOA adoption by combining with other open source components such as Struts, Axis, Spring, and Hibernate. In essence, Synapse is the equivalent of your open source ESB. ☺

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.

Mindreef®
www.Mindreef.com



The Principles of Service-Orientation

■ With the unwavering prominence of service-oriented architecture (SOA) there is an increasing interest in understanding what exactly it means for something to be considered “service-oriented.” Thomas Erl recently completed a lengthy research project for SOA Systems Inc. into the origins of SOA and the current state of service-orientation among all primary SOA technology platforms. This body of work contributed to the mainstream SOA methodology developed by SOA Systems and was also documented in Thomas’s new book, *Service-Oriented Architecture: Concepts, Technology, and Design*. We caught up with Thomas (a previous contributor to *WSJ*) to ask him to share some of the insights he gained from his work with SOA and service-orientation.

There’s no need to mention that SOA has become a major focal point of the IT industry and a primary consideration on numerous corporate agendas. Nor is there a need to get into how SOA has been so heavily promoted that the term has already reached hall-of-fame status as one of the most recognized acronyms in IT history.

What is more important than the term itself is the impact its perceived



WRITTEN BY
THOMAS ERL

meaning continues to have on how automation solutions are constructed. Its popularity to date is largely the result of vendors advertising SOA support or capability as part of their product lines. Because SOA has been so vendor-driven, its meaning has been somewhat divergent, skewed by proprietary technology that is still identified with common characteristics that transcend proprietary boundaries.

These common characteristics are critical to defining and understanding an abstract technology architecture classified as “service-oriented.” Viewing SOA in abstract is what establishes an agnostic reference point from which proprietary implementations can be measured and, ultimately, unified.

Vendor-Oriented Service-Orientation

Vendors and other organizations in the SOA space have published numerous papers, blueprints, and even frameworks. Most such documents serve the dual purpose of educating readers about SOA while marketing related products or services. This is nothing new. Past variations of client-server and distributed architecture models have varied significantly in both technology and design, depending mostly on who and what was used to implement them.

However, because a core expectation of SOA is its ability to harmonize and streamline diverse technical environments, preserving an abstract viewpoint is required to achieving its potential. This is because SOA, when elevated to an enterprise level, can be used to establish an ecosystem in which an agnostic, overarching framework transcends proprietary environments and constraints.

How the components and elements within this framework are shaped and standardized

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

is of critical importance. This underlines the need for a design paradigm that is sufficiently generic so that it can be applied to solutions regardless of implementation, while remaining in alignment with where powerhouse vendors and organizations are currently taking the technology that is fueling the service-oriented computing platform.

Service-Oriented and Object-Oriented

Design paradigms have played an important role in the evolution of technology and application architecture. The most widely recognized paradigm for distributed business automation has been object-orientation. The system-wide implementation technology for object-oriented solutions has traditionally been proprietary, where, despite the use of the agnostic principles of object-orientation, objects or components are designed to function and interact by using technology and protocols specific to a computing and/or vendor platform.

Service-orientation owes much of its existence to object-orientation. Like traditional multitiered architectures, SOA is based on

a model wherein solution logic is distributed. As with object-orientation, concepts such as encapsulation, abstraction, and reusability are fundamental to the design of distributed units of automation logic (services) within SOA. Key differences in these approaches are focused on how these units relate to each other and the scope at which the respective paradigms can be applied.

Service-Oriented and the Separation of Concerns

I have yet to find a better means of explaining service-orientation than to reach back to that fundamental software engineering theory known as the “separation of concerns.” This theory essentially proposes that larger problems be decomposed into a series of individually identifiable problems or “concerns.” The logic required to address or solve the larger problem can then also be broken down into

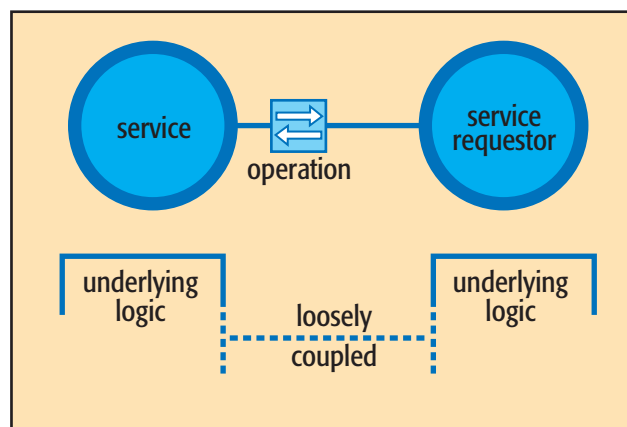


FIGURE 1 The underlying logic of a service requester accessing a service is not directly dependent on the logic that underlies the service itself. Services limit coupling to an exposed service contract, in this case a Web service interface with one operation.

individual units of logic that address specific concerns.

Past design paradigms and development platforms have applied this theory in different ways. Component-based and object-oriented designs, for example, provide specific approaches for the decomposition of concerns and the design of corresponding solution logic. Service-orientation establishes a new and distinct means of realizing a separation of concerns. As a design paradigm, it is an evolution of past approaches, augmented and extended in support of the overall goals and characteristics of SOA.

Common Service-Oriented Principles

Service-orientation began with a modest scope – a basic set of principles centered on an architectural model focused primarily on distinguishing services as reusable and discoverable resources. However, technology architecture in support of service-orientation is making significant strides, and extending its reach into key realms of enterprise computing.

Expectations are being raised surrounding a new era of business automation composed of services as adaptive, shared software assets that promise to infuse an enterprise with organization-level agility, federated interoperability, and vendor independence. These expectations have placed demands on what a distributed automation solution classified as “service-oriented” should be capable of, expanding the breadth of the service-oriented

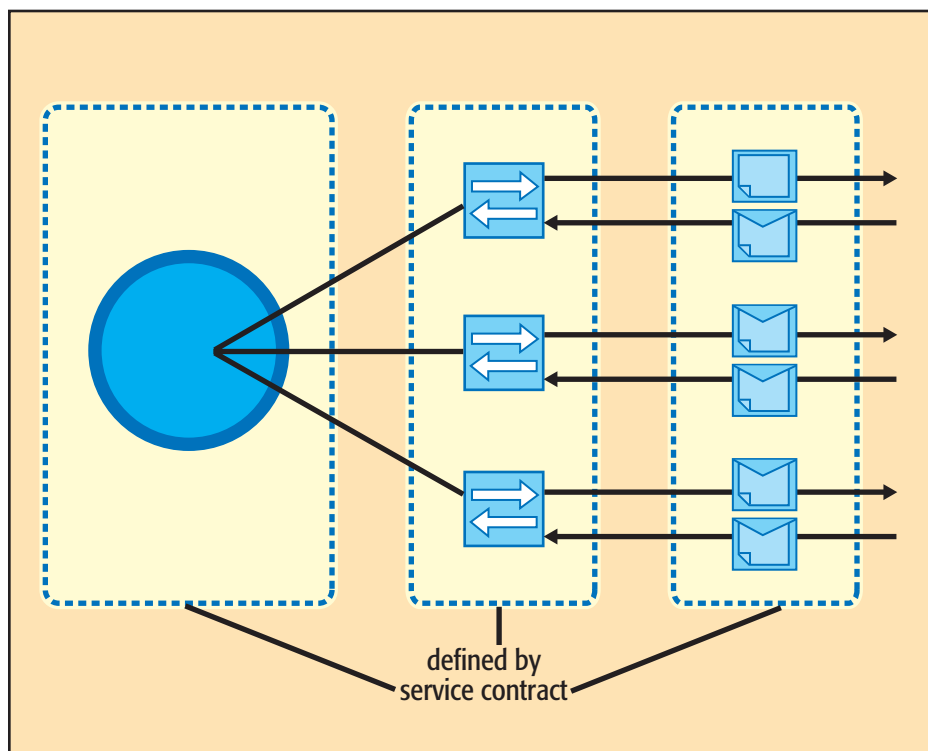


FIGURE 2 Service contracts formally define fundamental parts of a service. In the case of a Web service interface, this includes operation and message components, as well as the overall service context.

paradigm and adding to and further shaping its principles.

So far, eight common and fundamental principles have been identified. Note that these are classified as “common” in that they represent a cross-section of the most widely accepted design approaches and best practices promoted and practiced by the organizations most responsible for realizing the contemporary SOA movement.

Here then are the common principles of service-orientation:

- services are loosely coupled
- services share a formal contract
- services abstract underlying logic
- services are composable
- services are reusable
- services are autonomous
- services are stateless
- services are discoverable

Services are Loosely Coupled

Coupling refers to a connection between two things. Within automation solutions, the

extent of coupling is a measure of dependency in relation to the connection between units of automation logic. A tightly coupled relationship indicates a high degree of dependency, whereas a decoupled relationship indicates no dependency (and no connection).

A fundamental principle of service-orientation is that units of logic that are classified as services retain a minimal level of coupling. The underlying logic of a service and its requestor are therefore not tightly coupled nor are they decoupled. They ideally retain a level of coupling that is classified as “loose” by limiting dependencies between a service and its clients on an agreed-upon service contract.

Services Share a Formal Contract

Units of automation logic that are classified as services must provide a contract in which the terms of engagement are defined. As a whole, service contracts can be composed of legal and technical information. This principle is primarily concerned with service description documents that comprise the technical

service contract and provide published details about the service, such as its programmatic interface, communication requirements, constraints, properties, usage policies, and even preferences.

Software programs that request consumption of the service must adhere to the conditions of this contract. By restricting awareness of the service (as well as the scope of allowable communication) to what is documented in the service contract, a loosely coupled relationship is formed. The result is the consistent abstraction of a service's underlying logic.

Services Abstract Underlying Logic

Aside from what is published and described in the service contract, information regarding a service is hidden. This limits dependency (and coupling) to the service contract and essentially treats the service as a black box.

One of the primary benefits of service abstraction is that it allows for the mechanics of the service (development technology, deployment environments, etc.) to change with minimal impact to client programs that have become (loosely) dependent on it. The golden rule, of course, is that the service contract should remain immutable, so as to preserve established service-client relationships.

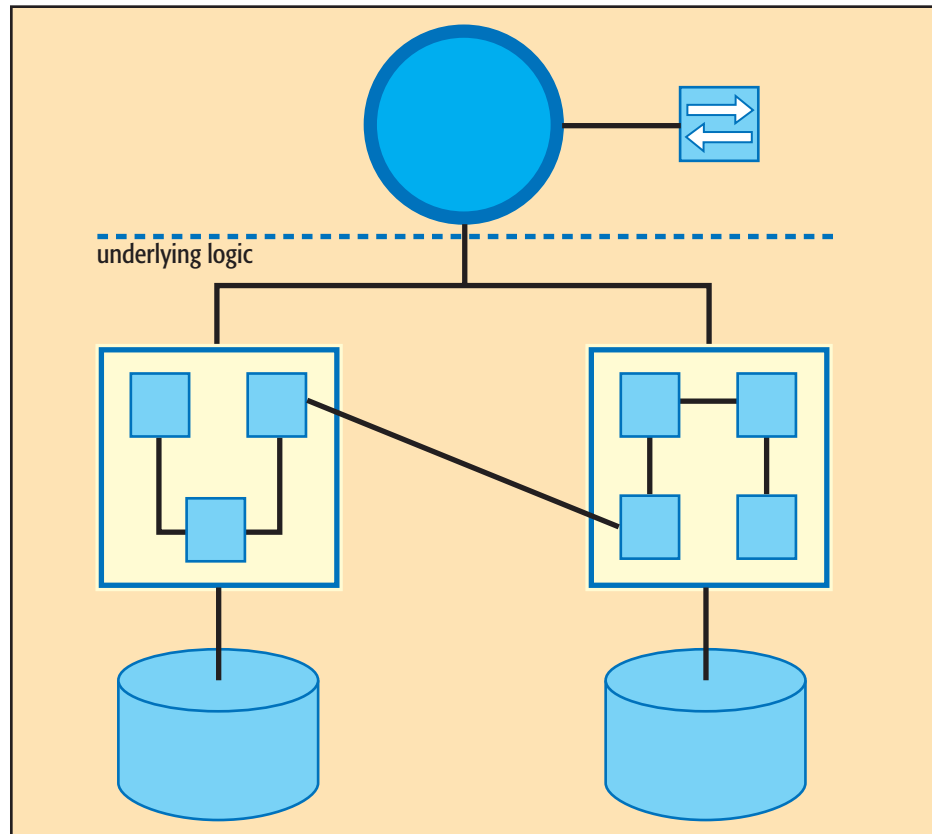


FIGURE 3 Web service operations are part of a service contract that abstracts the underlying details of the functionality they expose.

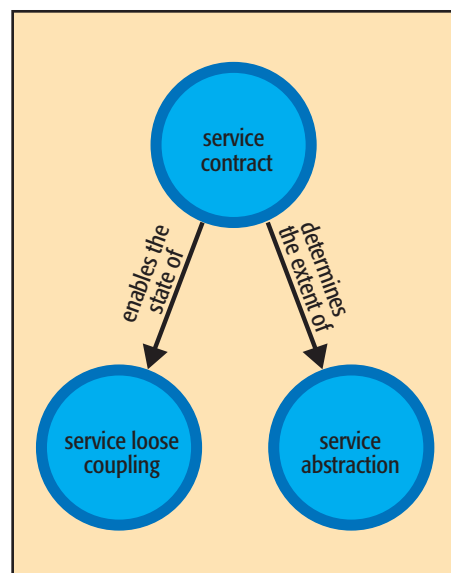


FIGURE 4 The three core principles form distinct relationships that establish fundamental parameters of service interaction.

“ The benefit potential for service-orientation is high, but with the broad application scope of its principles comes the responsibility of properly planning and standardizing its incorporation ”

The Core Trio

In a nutshell, the targeted use of service contracts abstracts underlying service logic, thus limiting the extent of coupling. Applying the principles that implement these characteristics results in a predictable and consistent relationship between a service and its requestor, thereby establishing a fundamental dynamic of primitive service-oriented architecture.

There are, however, additional characteristics that distinguish both service-orientation and SOA. These build upon the foundation laid by the core principles and are what mould services into standardized units of automation logic capable of realizing specific benefits associated with SOA.

Service Are Composable

Service abstraction imposes no limit on what a service can encapsulate. This includes encapsulation of other services. A service composition represents a set of aggregated services that are united to collectively perform a particular task. The principle of composability applies to individual services, and strongly encourages that services be designed in support of aggregated assembly as composition controllers, members, or both.

By ensuring that services are capable of participating in multiple compositions, an inventory of adaptive services can be accumulated. The introduction of new or augmented business requirements can then, to a larger extent, be addressed by the reorganization (or remodeling) of services into new composition configurations. Essentially, the modeling of these compositions can be viewed as a form of service reuse.

Services Are Reusable

Regardless of the design approach taken, a well-established benefit to separating concerns is that individual concerns may not be specific to a particular activity or business task. The solution logic decomposed to address these crosscutting concerns can therefore become reusable. The reuse potential in SOA broadens because the scope of SOA itself can be extended beyond solution or even enterprise environments.

Attaining an effective level of reuse often requires that a service be stripped of solution and/or business process logic in order for it to become adequately generic. As a result, service reuse is a principle that is almost always realized through standardized design. Key service characteristics that support and help to foster reuse are autonomy, statelessness, and discoverability – each of which is addressed by a corresponding principle.

Services Are Autonomous

It is preferred that services exist as independently as possible – both from each other as well as from other parts of the technical environment that may want to share the resources encapsulated by the service. Autonomy represents the governance a service has at the time of execution over the underlying application logic required to carry out the functions exposed by the service contract. The extent to which a service can control its underlying logic dictates the level of its autonomy.

Purely autonomous services have absolute ownership of their resources, which allows them to be better tuned for efficiency, reliability, and availability. However, attaining

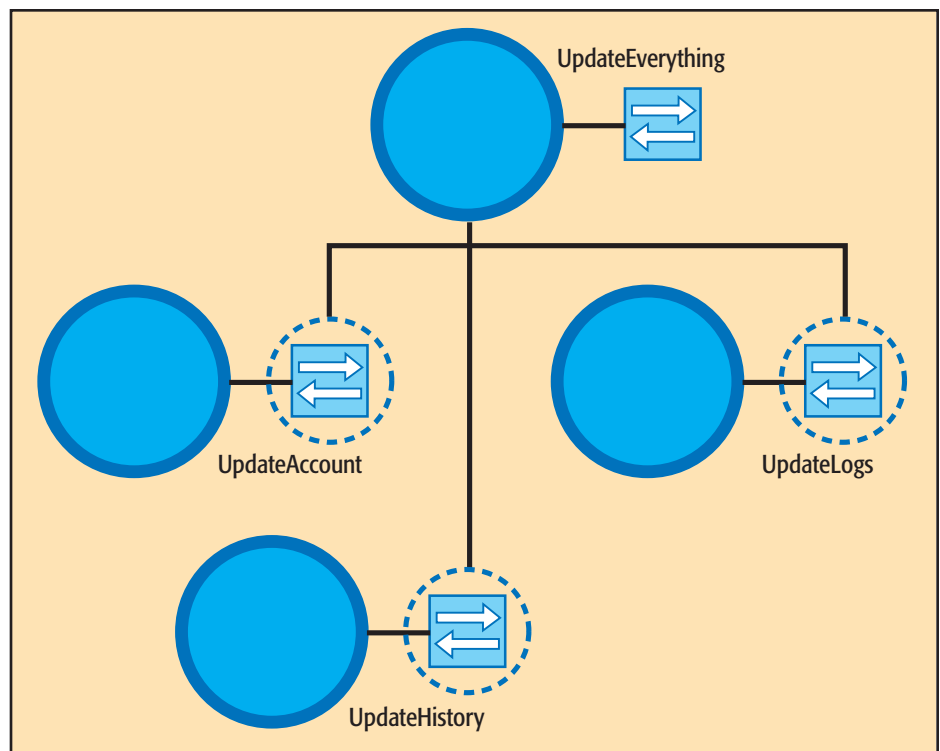


FIGURE 5 Service composition with Web services is actually achieved through the composition of individual service operations.

this level of autonomy can be challenging. For example, integration architectures where services encapsulate legacy system logic frequently require that resources be shared with existing legacy clients. Also, the autonomy of a service that is spearheading a composition will be dependent on the collective autonomy of all services involved.

Either way, services with a high level of autonomy are considered the best candidates for reuse, especially in environments in which usage patterns from multiple clients cannot easily be predicted. Note that although frequently discussed in tandem with reuse, autonomy in no way guarantees statelessness.

Services Are Stateless

State information is data specific to a current activity or task. Its lifespan is generally associated with this task and therefore the manipulation and retention of state-related data typically must occur at runtime.

State management can consume considerable resources. Maintaining a condition of statelessness therefore benefits a service by increasing its scalability and availability. Furthermore, the processing of state information typically requires automation logic that is specific to the business task being executed. For services to maximize reuse potential, their context and underlying logic must be as generic (task-neutral) as possible. Emphasizing stateless design supports the reallocation of task-specific logic outside of the service or to dedicated, stateful services.

Note that unlike autonomy, services cannot attain a level of pure statelessness. While processing, a service is stateful for a period of time. This principle simply emphasizes that the duration of this period be minimized.

Services Are Discoverable

Service discovery has been a well-promoted aspect of SOA. This principle does not require that service-oriented applications be outfitted with service registries or other forms of discovery mechanisms. Instead, it supports the overall notion of discoverability by focusing on the design of the service itself.

Service contracts describe technical aspects of a service to facilitate consumption by potential clients. Specifically, discoverability introduces conventions that foster clarity

and descriptiveness on a granular level. These conventions govern discoverability characteristics related to both human interpretation and automated evaluation of the service contract.

Additional Considerations

There are further qualities commonly associated with services and service-orientation that are relevant to generic service design, but that do not necessarily qualify as governing principles. A well-known example is the common belief that services should have coarse-grained interface designs. This type of consideration is important when designing service-oriented solutions, but should be classified

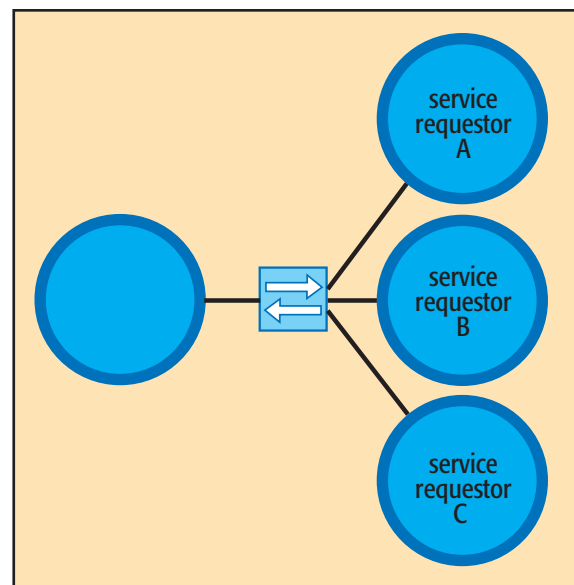


FIGURE 6 The functionality offered by an agnostic Web service operation can be reused by multiple service requestors, each possibly involved in different business tasks.

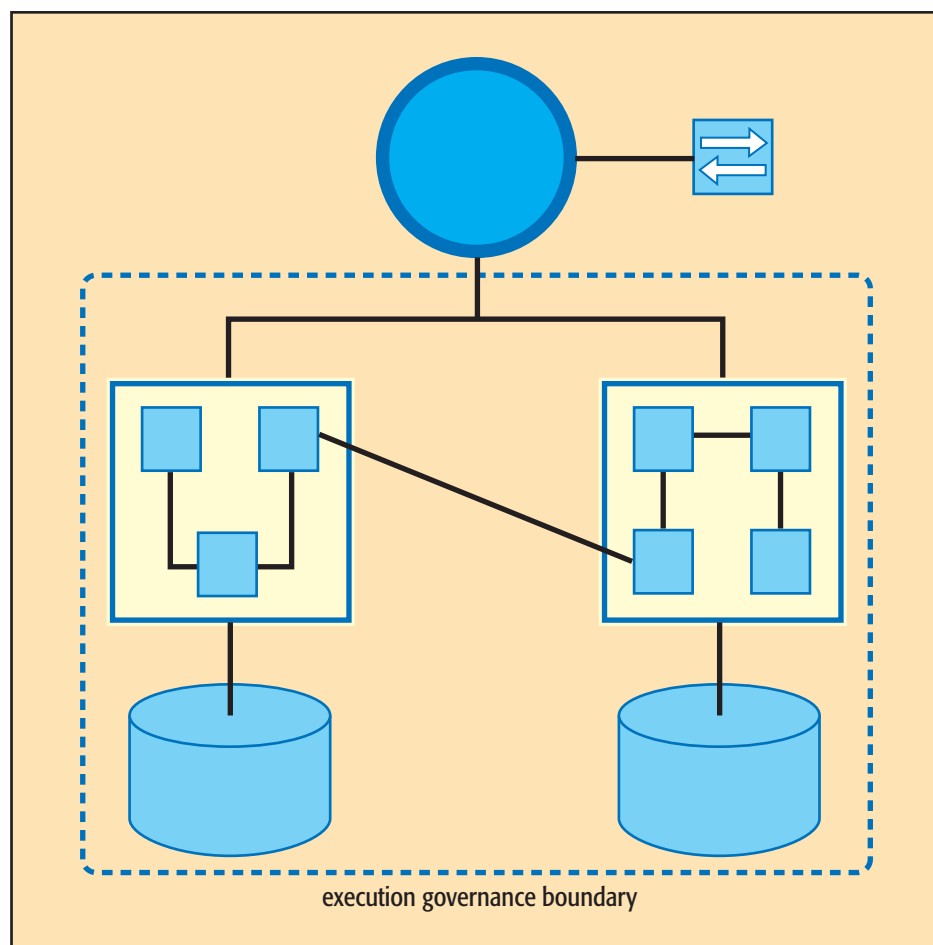


FIGURE 7 Autonomy can be assessed at the Web service operation level by defining the execution boundary required for the operation to perform its function.

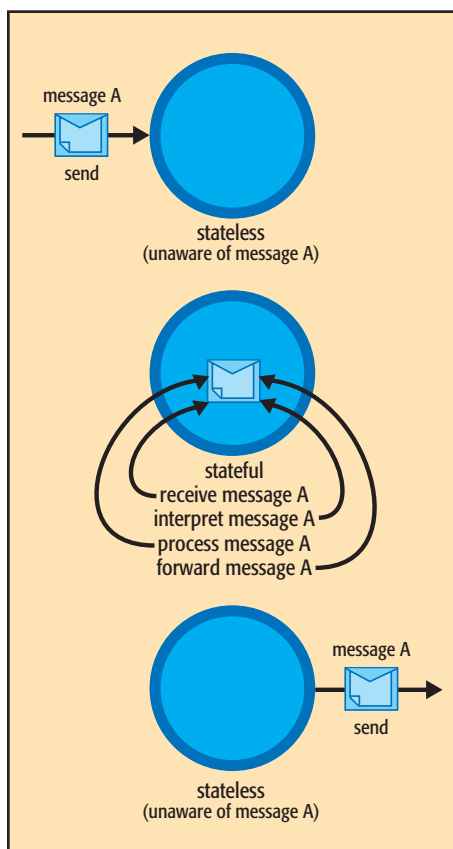


FIGURE 8 In this example, an intermediary Web service passes through stateless and stateful stages while processing a message.

more as a design guideline to be applied when it suitably fulfills specific requirements.

For example, it is almost always desirable for a service to be reusable, autonomous, and loosely coupled from its consumers; however, the appropriate measure of interface granularity needs to be assessed on a service-by-service basis (especially when designing compositions). This is why a distinction between principles and guidelines is beneficial.

How Service-Orientation Principles Interrelate

Each principle influences or is influenced by others. Interprinciple relationships establish the overall service-orientation dynamic and are therefore an important aspect that is worth understanding prior to incorporating service-orientation into concrete designs.

The manner in which principles interrelate is a broad subject matter. Relationships can be straightforward or complex, uni- or bi-directional, direct or inadvertent. Further, the extent to which applied principles affect each other can also be influenced by external factors, such as environmental conditions.

A glimpse of how the core trio from Figure 4 acts as a primary enabler of the remaining principles is illustrated in Figure 9.

Service-Orientation and Web Services

You may have noticed that we have yet to really discuss service-orientation in relation to Web services. As with object-orientation, the design paradigm established by service-orientation is intended to be implementation-agnostic. Its goal is to establish a set of interrelated principles for the purpose of attaining the benefits associated with contemporary SOA, regardless of how SOA itself is manifested in the real world.

Having stated that, though, it is important to acknowledge the Web services technology platform as providing the foremost means of realizing these principles. Loosely coupled relationships, the use of service contracts, the abstraction of underlying logic, and the ability for services to be composed are natural characteristics that can be realized through the mere use of Web services.

This highlights the suitability of Web services as a means of achieving service-orientation. In fact, it's fair to say that the emergence of these common principles has been directly influenced by the availability and popularity of the Web services technology platform. However, despite the ability for Web services to natively fulfill core principles, realizing an effective level of service-orientation requires attention to design and standardization.

For example, in the Web services world, reusability is made possible through deliberate analysis and design, during which generic processing logic is created and expressed through standardized WSDL, XSD schema, and policy definitions. Web service autonomy is also achieved through explicit design with the focus on the level of exclusivity a service has to the processing logic it encapsulates.

Statelessness within Web services can be maximized through the deferral of state-related processing responsibilities to other parts of the solution environment (including intentionally stateful activity management services), and discoverability can be enhanced through the use of naming conventions and annotations that are applied to individual service descriptions.

Also worth noting is the use of SOAP headers as a key feature of the Web services platform that supports several principles, including reusability, autonomy, and statelessness. SOAP header blocks allow messages to be outfitted

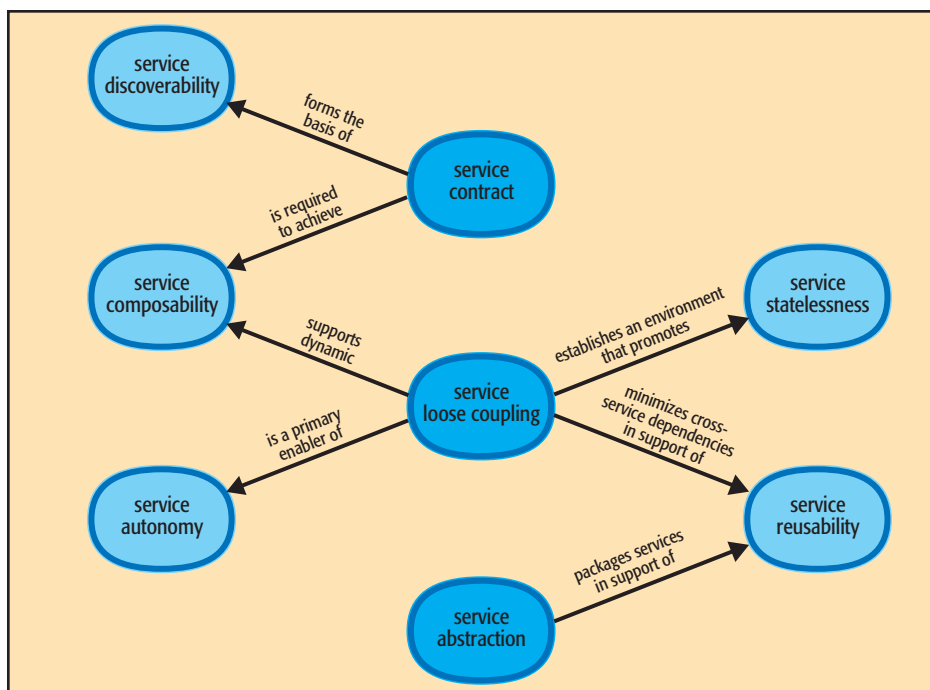


FIGURE 9 A perspective of how the core three principles support and affect the remaining five. Many additional perspectives exist, as each principle has distinct relationships with others.



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

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

24/7

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

JUMP TO THE LEADING i-TECHNOLOGY WEBSITES:

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

“ Services are the fundamental building blocks of SOA and the quality and consistency with which they are constructed ultimately determines not only the success of the solutions that utilize them, but also the potential of an organization to propagate service-orientation throughout its domains ”

with a wide range of metadata associated with the message contents as well as the business process or activity in which the message is participating. This further alleviates the need for services to contain task-specific logic, and emphasizes the creation of generic services that are capable of facilitating multiple requestors in multiple business process contexts.

Service-Orientation and the Enterprise

The scope of this discussion has so far been limited to principles that apply to the design of individual services. However, service-orientation is a broad paradigm that is capable of realizing large-scale SOA benefits through the application of key principles to the enterprise as a whole.

For example, services can be categorized into service models with predefined functional characteristics and implied, high-level contexts. These standardized services can then be organized into coordinated layers that are positioned to abstract entire domains within an organization. The use of service layers effectively implements a loosely coupled relationship between these domains. This significantly amplifies the effect of service-orientation, resulting in the potential to contribute to and even realize major benefits, such as

increased organizational agility.

Of course, it all begins with the service. Services are the fundamental building blocks of SOA and the quality and consistency with which they are constructed ultimately determines not only the success of the solutions that utilize them, but also the potential of an organization to propagate service-orientation throughout its domains.

To this end, I generally emphasize the development of highly agnostic and standardized service logic so as to allow an organization to consistently build an inventory of reusable and compatible services, each abstracting a distinct part of the enterprise. As the quantity of these services grows, the fulfillment of new business automation requirements becomes less of a development effort and more of a modeling exercise.

This aspect of service-orientation broaches a large subject matter I haven't yet touched upon in this article, namely the incorporation of service-orientation within business modeling and business logic encapsulation. Deriving services from business models, isolating process logic into an orchestration service layer, and using services to express enterprise ontologies are important steps for abstracting enterprise domains through service-orientation.

Summary

Variations of service-orientation exist, with numerous SOA vendors and organizations furthering its cause while also contributing divergent definitions and implementations. To accurately establish it as a mainstream design paradigm, extracting commonality from these contributions provides a useful, generic set of accepted principles.

These principles can be applied for different reasons and at different levels. They propose a shift in mindset and philosophy in solving real world problems through targeted abstraction of solution logic. The benefit potential for service-orientation is high, but with the broad application scope of its principles comes the responsibility of properly planning and standardizing its incorporation.

As with past design paradigms, simply separating concerns is not enough. A common vision of how this separation is accomplished is required. The unit of logic used to address a concern, the system built with units of logic, the enterprise composed of systems, the business community consisting of enterprises – all benefit from threads of commonality. Service-orientation is an ambitious paradigm that aims to define these threads with the ultimate vision of weaving them into all levels of business automation. ©

• • •

This article contains diagrams from SOA Systems Inc. methodology documentation and *Service-Oriented Architecture: Concepts, Technology, and Design* by Thomas Erl (792 pages, Hardcover, ISBN: 0131858580, Prentice Hall/Pearson PTR, Copyright 2005). For more information, see www.soasystems.com and www.serviceoriented.ws.

About the Author

Thomas Erl is the founder of SOA Systems Inc. (www.soasystems.com), an enterprise solutions provider specializing in SOA consulting, planning, and training services. Thomas is the author of the newly released *Service-Oriented Architecture: Concepts, Technology, and Design* as well as *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*, the best-selling book of 2004 in both SOA and Web Services categories. (Visit www.serviceoriented.ws for more information regarding these books.) He is a member of several OASIS committees and also participates as a speaker and instructor for private and public events and conferences. Thomas has published numerous papers, several of which have been printed in SYS-CON Media publications.

■ ■ ■ terl@soasystems.com



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:



SOA Testing Framework

A framework to test your services in service-oriented development

■ Service-oriented architecture (SOA) has become today's technology buzz and it's rapidly becoming a mainstream approach to enterprise systems design.

Beyond the buzz of SOA, organizations face several challenges as they attempt to truly effectuate the paradigm shift towards SOA. One critical challenge is: How can we assure the quality of the business services that we build? Can the services we build withstand the test of rapid organizational change? One way to address this challenge is through the use of effective testing methodologies and tools for the services deployed under an organization's SOA fabric.

A new development methodology – service-oriented development of applications (SODA) – is emerging to accompany the paradigm shift toward SOA. Today's agile software teams need effective tools to smoothen the transition. This article analyzes, designs, and demonstrates a tool that can prove invaluable to organizations implementing SODA: an automated services testing framework.

This article will demonstrate an automated SOA testing framework. Founded upon J2EE, XML, and SOAP as well as open source frameworks including Ant, JUnit, and XML-Unit, this powerful tool can prove invaluable to the agile organizations that are implementing SOA. Using the tool, organizations can both unit test for fine grained, "atomic" services as well as integration test loosely coupled, coarse grained business services that orchestrate the granular services. Since change is constant, development teams can automatically execute the services' original

WRITTEN BY
**ANBARASU
KRISHNASWAMY,
RAVI NAGUBADI, &
RAJEEV MAHAJAN**

set of test cases to ensure quality immediately as peripheral changes are continually being made. The framework has the following features:

- Completely XML configurable
- Advanced test input configuration (XML input, database state management)
- Sophisticated output verification capabilities (for both success cases and errors)
- Complete integration with Ant builds
- Automated JUnit code generation and execution
- Easy-to-read HTML report generated as output

The Need for Automated Services Testing

In any software development project, testing requires significant time, effort, and discipline. The following common steps should be followed to assure proper and efficient testing:

- Define a test plan that outlines the testing process and exit criteria

- Derive test cases from use cases or business requirements
- Generate test data and/or scripts for each test case
- Outline the expected results for each test case
- Execute the test cases
- Verify whether the results of the test cases match the expected results
- Generate reports to measure the software's quality against the test cases
- Fix/resolve remaining defects in the software
- Continue executing/verifying/fixing until all test cases succeed and defects are resolved

With the advent of extreme programming and agile development methodologies, organizations are advancing their testing efforts dramatically. Rather than employing error-prone, time-consuming manual tests at the end of development, many teams are now automating testing during – and often before – development. Code is integrated nightly and run against suites of automated tests to continuously verify quality.

Testing services in a SODA environment is not very different from testing ordinary software components; the main difference is that services are generally reusable implementations of *business* processes that orchestrate the functionality within lower-level components and other finer-grained services.

Services are highly amenable to testing. Services have well-defined interfaces and hence well-defined inputs and output. Their behavior is therefore predictable enough to apply commonly used black-box testing methods that are very amenable to automated testing.

Automated testing of services will become crucial to organizations that are undertaking SOA initiatives. Enterprise environments are constantly changing. For example, at one time, an organization may use JD Edwards as its ERP for managing orders. Then a merger happens, and suddenly, the underlying ERP is standardized to the parent company's Oracle ERP. In an SOA world, the underlying implementations of the order management services will have to be swapped. However, the service interfaces should remain the same. (More commonly, O/S patches are frequently installed, service

packs are applied to application servers, small fixes and enhancements are developed, and so on...)

Because change is constant, development teams should be able to automatically execute the services' original set of test cases to ensure the quality immediately as changes are being made. This will save untold time and effort, and will remove the error-prone elements of the process.

Typically, from the broad business requirements, individual service use cases are created. This will lead to service design and implementation. In a Test Driver Development (TDD), test cases are created from the use cases or service design well before the construction of the services. Test cases should be generated to test out a number of things such as decision points, validation rules, varying system states, business rules, and other expectable conditions. Figure 2 shows the transformation from service design to test cases to account for all of these parameters. Typically hundreds of test cases could be created to test the system thoroughly. Any changes to any of these parameters will mean it's necessary to run all of the tests all over again. This clearly warrants the need for the automated testing.

Requirements for a Testing Framework

A framework that can effectively meet the needs of software teams that develop and test services in a SODA environment should have the following capabilities to allow you to:

- Test services at a unit and integration test level (a unit test of a coarse-grained "composite" service naturally serves as an integration test for fine-grained services that are orchestrated by the composite service)
- Invoke target services using a standards-based approach (SOAP)
- Be very flexible and configurable to perform only the necessary steps in testing
- Allow test cases to be configured – rather than coded – through a user-friendly XML configuration file
- Leverage proven open source build/test technologies and frameworks (e.g., Ant, JUnit, XMLUnit)
- Auto-generate test case classes/code based upon XML configuration
- Provide options to adjust the underlying

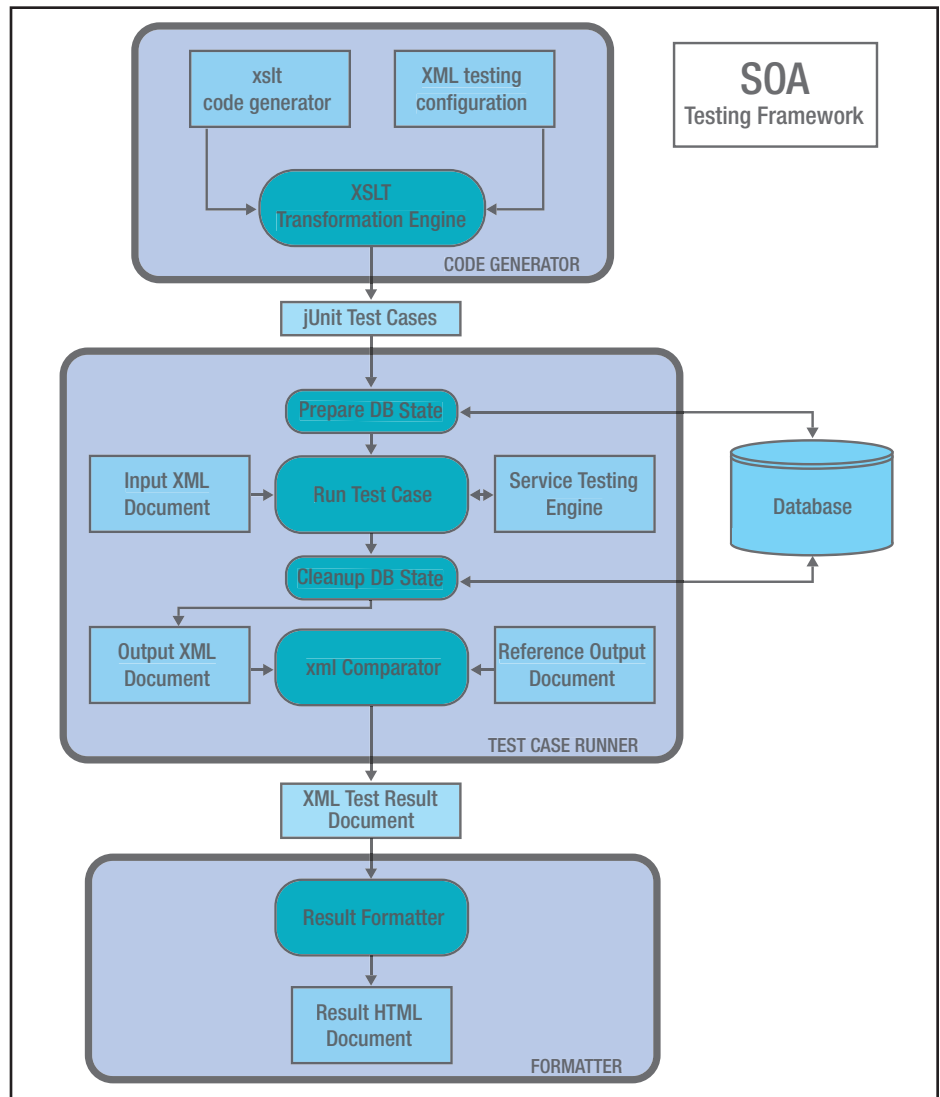


FIGURE 1 Design of testing framework

- ing state of the service ecosystem as an alternate and corollary "input" to a test case (e.g., to test for different service behavior based on variances of data in the database)
- Verify XML output of a service against the expected XML output defined for a test case
- Ignore specific XML nodes/elements during output verification (for variable like sequence numbers or date/time)
- Handle error conditions gracefully and return the error message to the caller in a standards-based way
- Provide the ability to set the state automatically for every test case run
- Verify error conditions against expected error code(s)

- Execute all necessary testing actions from a single command line
- Support running a batch of tests or one specific test at a time
- Generate well-formatted, human-readable reports about the test results
- Have the ability to integrate with automated build and testing frameworks like Cruise Control

Design of the Framework

Figure 1 shows the design of the framework. The framework is designed to fulfill the requirements that were presented in the previous section. JUnit is used to run the tests. An XSLT code generator reads the configuration

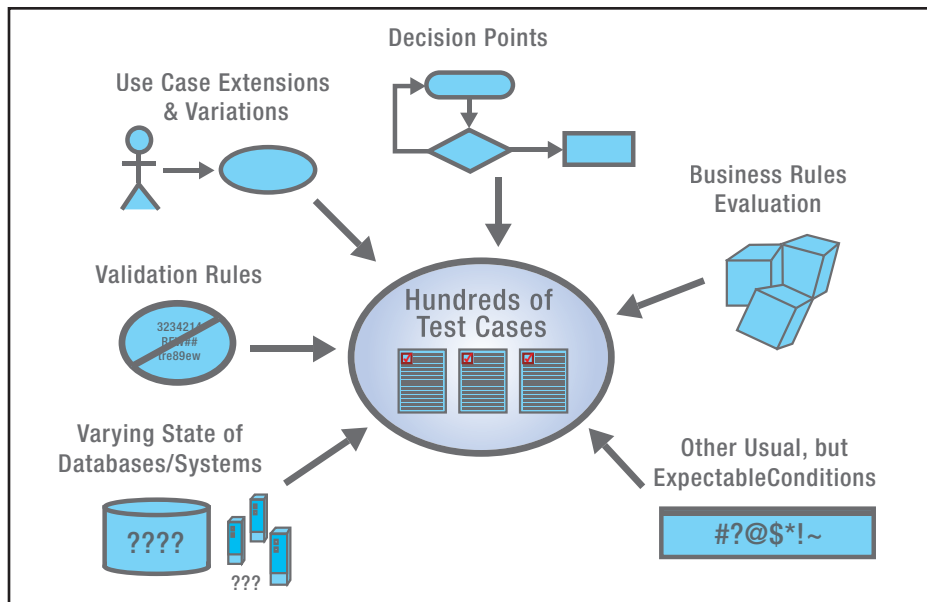


FIGURE 2 Service design to test cases

file and generates the test cases. Then the test case runner prepares the database state for each test case as configured and runs the test case using XMLUnit, taking the provided input document. The database state is restored back to the original value after the test is run. The result of the test is verified with the expected output by using a configurable XML comparator. Results of the tests are stored in XML files and formatted to user-friendly HTML as per the provided XSL style sheet by the results formatter.

Test Configuration File

The behavior of the tests is primarily controlled with an XML configuration file. This configuration file provides the following information to the framework:

- Name of the test case suite
- Base directory for all files for a test case suite
- Service URI to invoke the Web service
- Name of the test case
- Location and name of input document for test case
- Does the test case involve a change to the database to test for the effects of data variances?
- If so, database update/cleanup script locations and documents
- Is the expected result an error condition or a successful output?
- If error – what is the error code?

- If success scenario, location and name of output document against which to verify the result
- Document elements to ignore when comparing with the expected output

Given this information, junit test cases can be automatically generated. Table 1 illustrates the properties of a sample test case suite and Table 2 details the properties for each individual test case element. Listing 1 shows a sample configuration file for a fee-calculation service. Listing 3 shows the XML schema of the configuration file.

Code Generator

In order to execute each of the test cases individually and assert the results, junit requires each one of the test cases to be a separate method. In order to accommodate dynamic additions of services through configuration, the framework generates the junit test cases by using an XSLT code generator. XSLT is a natural choice for a code generator because it

can be easily integrated with Ant and the input is in the form of XML. It is also a very elegant approach to generate the test cases. Listing 2 shows a snippet from the XSLT code generator.

The code generator simply reads the configuration file and generates the junit test case method for each test case listed. One Java class is created for each test case suite, and the class is given a name based on the test case suite name.

Running the Code Generator

Since we created the code generator in XSLT, it is very simple to run it. We use the XSLT Ant task to do this. The following shows the Ant target:

```
<target name="codegen">
  <xslt basedir="test_data/codegen"
    destdir="${test.case.dir}"
    includes="Test*.xml"
    extension=".java" style="test_data/
      codegen/style/codegen.xsl">
  </xslt>
</target>
```

To run the code generator, run “ant codegen” from command line.

Test Runner

The test runner module is responsible for running the test cases generated by the code generator. Test runner runs the tests and compares the results from the service with the reference output. The test runner also makes sure that the database state is changed and restored at appropriate times and the result output is created. Test runner provides a choice of running a specified test case or all test cases.

To run all test cases, run “ant testOnly” from the command line. To run a single test case, run “ant runTestCase -Dtc=<test_case_number>.”

Setting Database State for the Test

When testing services, you will find the

Name	Description
Name	This is the name of the test case suite – the generated test case class will use the same name
baseDir	Specifies the base directory where the input and output files are present
inputDir	Input files directory – relative from the base directory
outputDir	Output files directory – relative from the base directory
serviceURI	Service URI is the end point of the service to be invoked

TABLE 1 Test cases (This table describes each of the attributes and elements for test cases and test case. Any attribute that is specified in the outer level test cases may be overridden by the elements in the inner level.)

need to make changes to the database in a number of situations. In SOA there is an easy way to perform this. If you create a generic service that performs this update, it can be invoked before the actual service is called.

What we did was to create a generic service that can execute any SQL statement provided. The SQL statements are listed in an XML file. When we change the state of the database, we must also reset it back to the original state after invoking the service. This is achieved by providing a list of compensating SQL statements in another XML file.

These files are placed in two different folders in the file system, so that the framework can access them as required. The database state change is achieved by means of two SQL files for each test case. An `sql_prepare` file has the SQL to change the database state before the test case is run, and an `sql_cleanup` file has the SQL to restore the database back to the original state. This is a very simple XML file that has multiple “statement” elements enclosed in an “sql” root element.

This is an example of the `sql_prepare` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<sql xmlns="http://www.yourcompany.org/
  payment/sql" xmlns:xsi="http://www.
  w3.org/2001/XMLSchema-instance">
<statement>delete from per_fee</statement>
<statement>insert into per_fee values
  ('Online Convenience Fee', 'FF34', 10.00,
  'WE98', 1, '', 'Y')</statement>
</sql>
```

This is an example of an `sql_cleanup` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<sql xmlns="http://www.yourcompany.org/
  payment/sql" xmlns:xsi="http://www.
  w3.org/2001/XMLSchema-instance">
<statement>update per_config set value =
  '10.77.2.9' where param='CS_HOST_IP1'</
  statement>
</sql>
```

While this example shows changing the state of the database, it could really be anything. It could be reading and populating a cache, updating some information in the session, or running another service to set the context for the service to be run.

Name	Description
Name	This is the name of the test case – the generated test case method will use the name.
baseDir	Specifies the base directory where the input and output files are present
inputDir	Input files directory – relative from the base directory
outputDir	Output files directory – relative from the base directory
serviceURI	Service URI is the end point of the service to be invoked
errorCase	Indicates if this test case is an error case or success case (the valid values are “Y” and “N”; “Y” means error case and “N” means success case)
inputFileName	Name of the file that has the input XML document
outputFileName	Name of the file that has the expected output XML document (applicable only if errorCase is “N”)
Ignore	Elements in the response document that should be ignored when comparing the actual output and expected output (multiple elements may be listed as a comma-separated list)
errorNumber	Expected error number (applicable only if errorCase is “Y”)
dbUpdate	Indicates if any changes to the database state are required for this test case. If this is “Y,” the SQL files in <code>sql_prepare</code> and <code>sql_cleanup</code> folders under the <code>baseDir</code> will be executed before and after the test case execution, respectively. The file names should be <code><test case name>_sql_prepare.sql</code> and <code><test case name>_sql_cleanup.sql</code> .

TABLE 2 Properties for each test case element

Service Testing Engine

The testing engine encapsulates the common logic required to read the input data and reference output data. It uses SAAJ API to invoke the services in a generic fashion. It also has the logic to verify the results of the service while ignoring specified elements. Outputs of many services have variable elements that change for every invocation. Some examples of this are transaction ID, transaction date and time, and system-generated parameters.

The test engine also handles the error scenario when the service may return a SOAP fault with an application-specific error code. It compares the error code returned by the service with the expected error code specified in the configuration file.

Results Formatter

Results formatter consolidates all of the test results and creates human-readable output in HTML format. The formatter can be run using the “ant junitformat” command. The format of the output is specified in an XSL.

Summary

In this article we discussed how useful and powerful an automated testing framework could be in an SOA development. We also showed some techniques to implement the testing framework. There is a trend in the industry to move toward a shared service model. IT should be aligned with business and should be responsive to the

changes in business. This makes changes in services inevitable. As multiple departments consume the services, it is important to upgrade the services but to still reduce downtime. There is no doubt that the SOA testing framework will be a very useful tool for achieving this.

About the Authors

Anbarasu Krishnaswamy is a consulting technical manager with BEA professional services. He is a six-year BEA veteran, and has been working on enterprise applications for over 12 years. He holds a Masters degree in Computer Science and is certified by BEA and Sun. In his current role he helps customers in various vertical industries to architect enterprise applications. He specializes in architecting enterprise systems using J2EE, EAI, and SOA.

■ ■ ■ anbarasu@bea.com

Ravi Nagubadi is a principal consultant at Sofbang LLC. He brings eight years of rich technology experience with a strong focus on solutions to critical business issues. He has consulted in a diverse set of industries such as healthcare, government, education, insurance, and financial services. His expertise includes portals, commerce, content management, business integration, and enterprise architecture.

■ ■ ■ magub@sofbang.com

Rajeev Mahajan is a practice manager with BEA and brings 15 years of deep IS technology experience to his skill of linking business drivers to technology. He has broad industry experience in healthcare, high tech, financial services, manufacturing, and retail. He also has implementation experience with portals, content management, CRM, e-Business, and enterprise application integration.

■ ■ ■ rmahajan@bea.com

Listing 1: Sample Configuration File

Listing 1 displays the configuration of a service test case suite that contains two test cases – one that expects an error condition and another that expects a successful output result. The listing demonstrates the hierarchy of data elements. All of the common properties are specified at the test case suite level and specific properties are specified at the test case level. A property that is specified at the test case level will override the same property at the global level. For example, you can change the base directory for a specific test case by setting it in the test case properties.

```
<testCases name="TestAddFeeService" baseDir="test_data\\
addFeeService\\" inputDir="in\\" outputDir="out\\"
serviceURI="http://localhost:7001/paymentEngine/org/
yourcompany/soa/payment/processes/fee/FeeService.jpdl">
  <testCase name="1_2_1a" errorCase="Y" dbUpdate="N">
    <baseDir></baseDir>
    <inputDir></inputDir>
    <outputDir></outputDir>
    <inputFileName>1_2_1a_in.xml</inputFileName>
    <errorNumber>10007</errorNumber>
    <serviceURI></serviceURI>
    <ignore>transactionTime</ignore>
  </testCase>
  <testCase name="1_2_27" errorCase="N" dbUpdate="Y">
    <baseDir></baseDir>
    <inputDir></inputDir>
    <outputDir></outputDir>
    <inputFileName>1_2_27_in.xml</inputFileName>
    <outputFileName>1_2_27_out.xml</outputFileName>
    <serviceURI></serviceURI>
    <ignore>transactionDate,transactionTime </ignore>
  </testCase>
</testCases>
```

Listing 2: Code Generator XSLT

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.
w3.org/1999/XSL/Transform">
  <xsl:output method="text"/>
  <xsl:template match="testCases">
    package org.yourcompany.tests;
    import javax.xml.soap.SOAPMessage;
    import junit.framework.Assert;
    import org.custommonkey.xmlunit.Diff;
    public class <xsl:value-of select="@name"/> extends
    BaseTestService
    {
      <xsl:for-each select="testCase">
        public void test<xsl:value-of select="@name"/>()
        {
          try
          {
            String testCaseName = "<xsl:value-of
select="@name"/>";
            log("\n\n***** Begin test case: " + testCaseName +
            " *****\n\n\n");
            System.err.println("\n\n***** Begin test case: " +
            testCaseName + " *****\n\n\n");
          }
        }
      }
    }
  </xsl:template>
</xsl:stylesheet>
```

Listing 3: Configuration File Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" ele
mentFormDefault="qualified">
  <xs:element name="baseDir">
    <xs:complexType/>
  </xs:element>
```

```
<xs:element name="errorNumber">
  <xs:simpleType/>
</xs:element>
<xs:element name="ignore">
  <xs:simpleType/>
</xs:element>
<xs:element name="inputDir">
  <xs:complexType/>
</xs:element>
<xs:element name="inputFileName">
  <xs:simpleType/>
</xs:element>
<xs:element name="outputDir">
  <xs:complexType/>
</xs:element>
<xs:element name="outputFileName">
  <xs:simpleType/>
</xs:element>
<xs:element name="serviceURI">
  <xs:complexType/>
</xs:element>
<xs:complexType name="testCaseType">
  <xs:sequence>
    <xs:element ref="baseDir"/>
    <xs:element ref="inputDir"/>
    <xs:element ref="outputDir"/>
    <xs:element ref="inputFileName"/>
    <xs:element ref="errorNumber" minOccurs="0"/>
    <xs:element ref="outputFileName" minOccurs="0"/>
    <xs:element ref="serviceURI"/>
    <xs:element ref="ignore"/>
  </xs:sequence>
  <xs:attribute name="name" use="required">
    <xs:simpleType/>
  </xs:attribute>
  <xs:attribute name="errorCase" use="required">
    <xs:simpleType/>
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="N"/>
      <xs:enumeration value="Y"/>
    </xs:restriction>
  </xs:attribute>
  <xs:attribute name="dbUpdate" type="xs:string"
  use="required"/>
</xs:complexType>
<xs:element name="testCases">
  <xs:complexType/>
  <xs:sequence>
    <xs:element name="testCase" type="testCaseType"
    minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string"
  use="required"/>
  <xs:attribute name="baseDir" type="xs:string"
  use="required"/>
  <xs:attribute name="inputDir" type="xs:string"
  use="required"/>
  <xs:attribute name="outputDir" type="xs:string"
  use="required"/>
  <xs:attribute name="serviceURI" type="xs:anyURI"
  use="required"/>
</xs:element>
</xs:schema>
```


Subscribe Today!

— INCLUDES —
FREE
DIGITAL EDITION!
(WITH PAID SUBSCRIPTION)
GET YOUR ACCESS CODE
INSTANTLY!



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

Editorial Mission

Greater Collaboration and Efficiency Through Education

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

SAVE 50% OFF!

(REGULAR NEWSSTAND PRICE)

Only \$39⁹⁹

ONE YEAR
12 ISSUES

www.ISSJournal.com

or

1-888-303-5282

**SYS-CON
MEDIA**

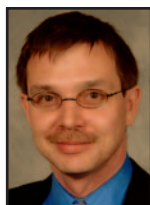
The World's Leading IT-Technology Publisher

The “B” in BPM Stands for Business

Differentiating pure-play BPM from rebranded EAI and Workflow technologies

■ The use of the term BPM (business process management) is becoming increasingly confused as former EAI vendors claim BPM capability and the lines between BPM + SOA and ESB applications are blurred. This article will differentiate what is unique and distinct about BPM compared to other technologies such as EAI (enterprise application integration) and ESB (enterprise service bus), and it will also attempt to characterize what exactly constitutes a Business Process as opposed to an Application Integration, Web Service Orchestration, or Data Translation problem. All the while we will be remembering and focusing on the fact that the “B” in BPM stands for Business.

Enterprise Application vendors who are keen to preserve the revenue-generating capabilities of their offerings and who are also keen to grow market share have historically recategorized themselves with emerging market segments and their associated three letter acronyms. For example, consider sales force automation (SFA) and contact management applications that overnight morphed into customer relationship management (CRM) applications, and applica-



WRITTEN BY
KARL TREIER

tion server vendors that became ESB platforms. It's not surprising then, given the dramatic growth being experienced and predicted for BPM applications (some at the expense of other market segments such as EAI) that we yet again see vendors of similar, but different technologies such as EAI and Workflow claim that these are BPM offerings. The unfortunate effect is buyer confusion as to just what a BPM tool is and what a BPM tool is not. To help buyers understand the distinction, let's start by clarifying the fact that true BPM

technologies may have EAI and Workflow roots, though they were built from the “ground up” to close the gap in EAI and Workflow deficiencies. Since BPM technology is a relatively new software space, even the most mature BPM companies have only been around for eight years or fewer. These new players and independent thought leaders in the space can only hope to continue to try to educate the user community about the true characteristics of an application that will deliver the promise upon which the technology was predicated.

Let's examine what caused the BPM application category to emerge and how it is radically different from preceding technologies, even though vendors of those technologies claim to now be BPM applications. To understand where BPM came from you need to recognize that existing technologies were not filling a missing need in the growing digital workplace. The two major technologies to consider in this category are EAI and Workflow. EAI was borne from the fact that businesses were investing considerable sums in CRM, ERP, B2B, eCommerce, and other applications that addressed the needs of a narrow user community inside their enterprise, but each application was tied to its own data repository. To synchronize the data between these applications and external partner companies, EAI, with its catalog of application-specific adapters, emerged and filled the need. Given

the emergence of Web services as a standard mechanism for interacting with applications, EAI applications have shed their dependence on adapters and have morphed into a new breed of application with a new acronym and an equally technical sounding name, so now we have ESB applications. I'm opening myself to endless debate I'm sure, but I would categorize applications from Microsoft (BizTalk), Tibco, and Vitria squarely in this category. Workflow, on the other hand, was focused not on system-to-system interaction, but rather on human-to-human interactions and the sequencing of work from one person to another. However Workflow didn't really attempt to automate any step of the process; its objective was quite simply to ensure that the right person was tasked with an activity at the right time in a well-defined sequence. When you consider what the focus was of each application category it is clear to see that EAI was focused on systems and data, and Workflow was focused solely on the human worker.

So what is the philosophy behind business process management? Well, first of all let's agree

that BPM applications exist to support business performance. BPM focuses on process and rules as being the central and controlling entities. The human workers, data, and systems are supplemental entities that the process interacts with and orchestrates to accomplish the goals of the process. Second, business processes are inherently human-worker dominated. A recent Gartner survey revealed that 85 percent of BPM implementations were around human worker-centric processes. The systems and applications that the human workers interact with are merely vehicles by which the human workers get access to the data they need to perform their functions in an efficient manner. Let me just repeat this last sentence again in a different way so that this crucial point gets fully communicated. The systems and applications that your workers use are tools that merely provide data and enforce rules in a format that is relevant to their job functions. A BPM tool should not seek to supplant those systems and applications, it should instead empower the workers to continue to perform their tasks in the applications that are most relevant to

their job functions. However it should eliminate nonproductive activities such as manual re-keying of data into those applications. So in this respect the third key characteristic of a BPM application sometimes causes it to be confused with EAI applications (or more likely cause EAI applications to be characterized as BPM). BPM applications should be able to move data between diverse systems, and not just enterprise applications like CRM and ERP, but also classic human worker productivity tools such as spreadsheet, word processing and collaboration applications.

I think the best way to differentiate BPM from EAI and Workflow is to examine an example of a process that could not be handled by EAI or Workflow alone, that exhibits all of the characteristics described above, and that has been successfully implemented using a pure-play BPM application.

Figure 1 is a conceptual diagram of the enterprise customer acquisition process that might be found in any company that sells configurable products or services at negotiable prices and

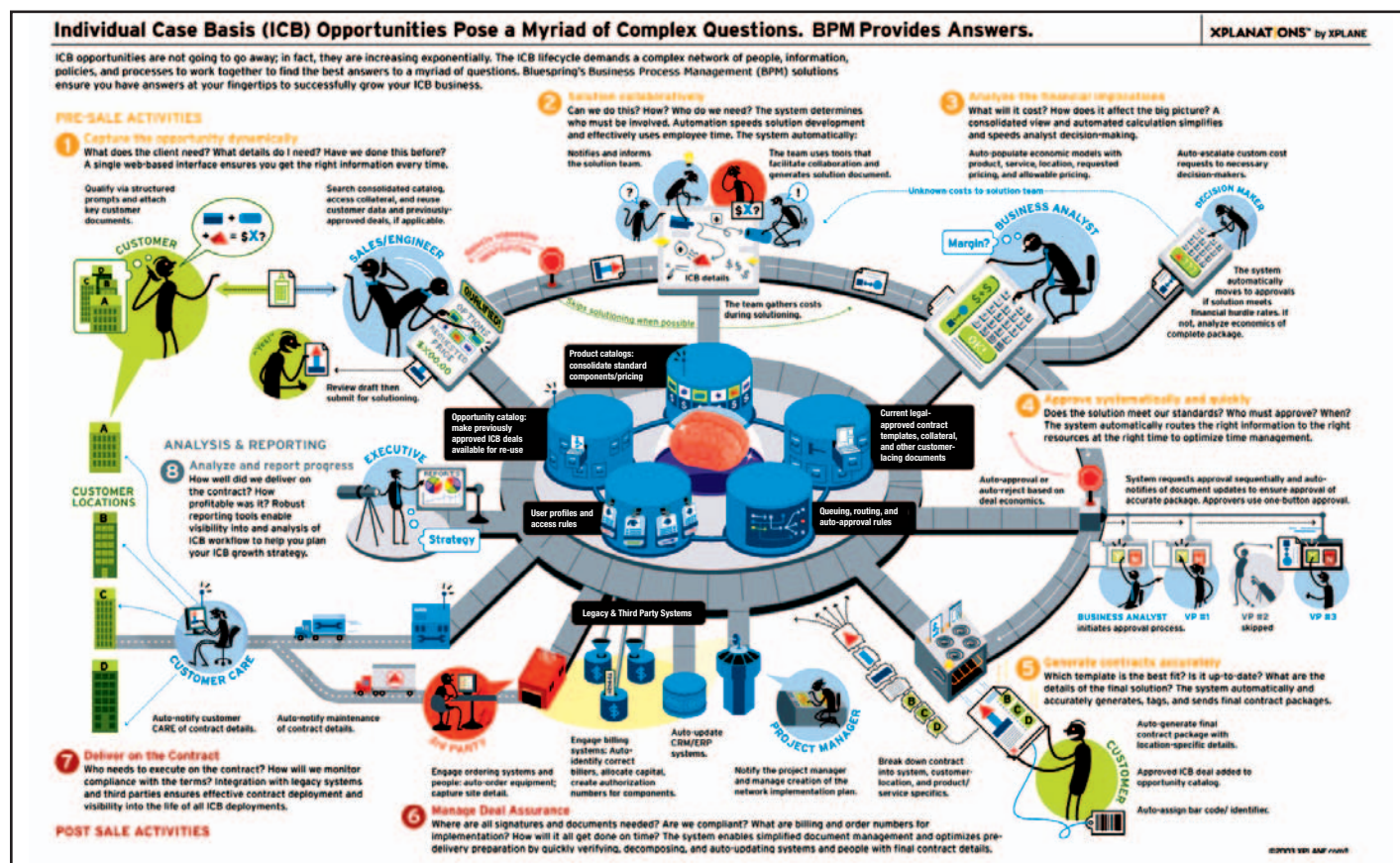


FIGURE 1 Enterprise customer acquisition

under custom contract terms. This particular process starts at step 1 with a sales executive or sales engineer entering the characteristics of the opportunity such as customer details, product configuration, quoted pricing, and contract terms into a CRM application (the relevant and preexisting application for that user community). This starts the process inside of the BPM application that takes over the management of the process, tasking of workers, application of rules, and migration of data from system to system. In this example the engineering department (step 2) is tasked with the next set of activities that are relevant to the process and the data is populated into the relevant pre-existing application. Step 3 sees the process and associated data moving on

and review of custom-contract language via a word processing program and collaboration portal. The end product of this process is the automated creation of contract documents that are populated with all of the data captured in the CRM application and each application used along the way, and because a BPM application manages the entire process, users and management gain real-time visibility into the state of any opportunity moving through it.

Could an EAI application achieve the same result? It is extremely unlikely: at best EAI applications could move the data between applications, but EAI would not task or notify users when they needed to perform some action on that data, nor would it provide the end-to-

to the employee, rather than forcing them to change behavior and learn a new application. Automation is introduced where appropriate, such as the elimination of nonproductive activities like the rekeying of data, all the while allowing the human workers to interact with the process using common desktop tools that they already know how to use. Only BPM applications can solve or handle complex, highly manual, enterprise-wide business processes—like customer acquisition, demand planning, forecasting, and IT asset management. Only process management technologies that facilitate human worker interaction within the process using the preexisting and relevant applications those human workers already have is a true BPM application. It is this author's opinion that the best BPM application is one that is transparent to human workers in all respects other than that it should seek to eliminate nonproductive activities.

In summary I will digress slightly by highlighting a legitimate polarization that is occurring between true BPM applications. Of course polarization is occurring based on platform (J2EE vs. .NET), and to a lesser extent, on underlying architectural philosophy (SOA or not). The emerging difference that I want to highlight centers on the notion of who is doing the implementation work. Who designs the process, who defines the rules? Should this be put in the hands of IT professionals or the process owners (business analysts and subject matter experts)? BPM tools were created to optimize business performance and provide an agile infrastructure to keep up with the business as it evolves over time. BPM tools automate where appropriate and effectively interact with the human worker elsewhere. These characteristics represent the makeup of a legitimate BPM offering. ©

■ About the Author

Karl Treier is the chief technology officer at Bluespring Software (www.bluespringsoftware.com). Karl is a 20-year veteran of the IT industry and has performed a broad range of classic leadership roles in development, architecture and product management, though it was his early customer-facing roles in training and support that have inspired a customer-centric development methodology that has translated into a world-class BPM product, built on the Microsoft .NET platform. Karl is a graduate of Wakefield College in England and also holds numerous industry certifications, including MCSD, MCSE, SCJP, and CCNA.

■ ■ ■ karl.treier@bluespringsoftware.com

“ To help buyers understand the distinction, let's start by clarifying the fact that true BPM technologies may have EAI and Workflow roots, though they were built from the 'ground up' to close the gap in EAI and Workflow deficiencies ”

to the financial analysts who have responsibility for ensuring the profitability of the opportunity. BPM eliminates the re-keying of data at this step by automatically creating and populating Excel spreadsheets that are derived from templates and applies rules based on the results of calculations within those spreadsheets to make a determination as to whether or not the financial analysts even need to review the opportunity for profitability. If the financial analysts need to review or approve the opportunity then the spreadsheets are e-mailed to them for review and approval. By way of the application of rules defined within the BPM application, step 4 ensures that the right levels of management are involved in approving the opportunity, and their interaction is facilitated through e-mail. Finally step 5 involves legal in a collaborative process around the creation

end visibility into the process and its human worker activities. What about Workflow? Certainly Workflow would be able to sequence the human workers within the process, but its inability to integrate systems and applications into the process would almost certainly mean that users would end up having to use UI elements and forms that are rendered by the Workflow application. Since Workflow applications also merely just sequence the activities but not the movement of data, no real audit trail or guaranteed data integrity will exist.

BPM on the other hand was successful in implementing this enterprise-wide process because it started with and focused on what the process and rules were, not on what the systems were. Most notably, BPM enables people to do their jobs in a way that comes naturally

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

Describing Web Services in 2005

How to make Web services consumable today

■ For years, WSDL has been synonymous with describing interoperable Web services. However as the now venerable Web Services Description Language (WSDL) 1.1 heads towards its fifth birthday, it finds itself beset with challenges and opportunities. Currently in its second incarnation and heading for its third, WSDL has been the definitive gold standard for providing descriptions of Web services. It has especially been a smash success in terms of being the only broadly accepted format for Web service description. Moreover, it has generally done a commendable job at it, and until recently, WSDL has been without much competition.

The Web services world has changed considerably though in the last half a decade and the problems that the Web services model originally addressed have also changed appreciably. At the same time, the Web services community itself has grown in size and diversity, often in leaps and bounds. The same community has also evolved significantly in terms of understanding its own requirements for Web service description. In the face of the widespread success of Web services, recent developments, and its own near-ubiquitous adoption, will WSDL and its successor always be right for the job?

In the last several years, increasing diversity of Internet clients and services has become an important development for the Web services world. For example, entities such as online retailers Amazon and Yahoo now allow external partners to plug into their own IT infrastructure through Web services with very little formality. The number and style of computing platforms that provide Web services have also proliferated. Notable new



WRITTEN BY
**DION
HINCHCLIFFE**

languages and development platforms such as Ruby and Python have emerged as Web services players, to name just two of the more popular ones.

The standards side of the Web services story has grown more complex and sophisticated too. New arrivals are put into the mix of WS-* standards all the time. Absorbing all of these new Web services standards is only part of the problem that organizations face. It's describing the use of these

standards to partners and clients that is becoming a growing challenge. Fitting WS-* descriptions into WSDL's aging view of describing Web services has also not been straightforward. This is because many new Web services standards, most of which are layered on top of SOAP via its extensibility model, often don't have a well-defined mechanism to describe them in the WSDL despite advances in this area with WS-Policy. This means how WSDL fits into large Web services stacks and non-SOAP Web service protocols will continue to be a problematic and important issue. It's not yet clear how well the next version of WSDL will address all of these challenges.

WSDL: The Web Services Description Workhorse

What advantages does WSDL provide today for the current generation of Web services creators and consumers? Ostensibly, WSDL still delivers a very usable, standardized, XML-based (and hence machine readable) description that most Web services development tools can use to make it easy to understand how to interact with a Web service. This includes information such as location, protocol, operations, and input and output message structures.

A classic example of what automated tools can do with WSDL is the generation of a proxy class. In this scenario, the WSDL for a given Web service is read and used to create a representation of the Web service's interface in the local programming model. This is usually a proxy object that looks like a local resource. Using it is indistinguishable from calling code written locally. When the automatically generated proxy object, guided by the WSDL, is invoked, it actually interacts with the Web service to do the work and return any results. There are significant advantages and disadvantages to using WSDL in this way as recent industry experience has revealed, and as we will see below.

Fortunately for the Web services community, WSDL has generally aged well. The vast majority of the experience with WSDL over the last few years has been quite positive. In fairly static environments with sophisticated Web services toolkits, wiring together clients to Web services is greatly facilitated by WSDL. WSDL makes it straightforward for automated tools to read the message structure, operations, and endpoint information from a WSDL document and create the easy-to-use local representation of the Web service. Programmers usually know exactly what to do with an object in the local programming language that represents the Web service. This is terrific in a homogenous world, especially behind an organization's firewall, and it has worked well for Web services users for years.

However, as Web services really begin to be used in an ever more widespread manner on the Internet and by more widely diverse clients, the inevitable flaws in WSDL have become apparent. For one thing, WSDL is designed to be used by automated tools. If one has to process it manually or develop a tool to do the same, WSDL's structure can make it rather difficult to do. Second, the type system that comes stan-

dard with WSDL, XML Schemas (also known as XSD), is capable of doing things that either most programming languages can't do, or that can only be done in one particular programming language and not another.

To recapture this interoperability, there has been a growing trend moving away from developing a Web services programming language first. Contract-first Web service development is increasingly popular and helps ensure the message structures in the Web service will work with most programming languages. This trend is being driven by the fact that Web services are succeeding in bringing together increasingly different groups of users possessing different Web services environments, both inside and outside an organization. In particular this means a given Web service is ever more likely to be used from a multitude of different programming languages.

However Web services built from a programming language-first perspective and subsequently described in WSDL create a problem: a Web service that tends to be polluted with idioms from the programming language. A Web service developed this way is difficult to use from environments that aren't the same as the one in which it was created. All of these issues chip away at the promise of ubiquitous interoperability that Web services have offered from the beginning.

Looking Forward to WSDL 2.0

To address the experiences with WSDL over the last few years, the Web services community has been hard at work researching and repositioning WSDL and Web service description in general. In the W3C space, work on the next generation of WSDL, version 2.0, has been under way for some time now. WSDL 2.0 has the promise to describe not only traditional SOAP Web services, but also a wide variety of services that can be provided over the network, particularly including REST.

Certainly WSDL is heading in the right direction goal-wise. Reconciling and aligning WSDL so that it encompasses a wider variety of Web services models like REST and others will be extremely helpful to frequent consumers of Web services, particularly if they are from external organizations. Most organizations would probably be very happy to have a Web service description language that covers 90 percent of their needs.

WSDL 2.0's inclusion of several popular Web services models is an excellent step as well as a necessary one if a single, standardized Web description language is to be widely adopted. Also while WSDL 2.0 shows promise, adoption will likely be hindered by its almost too-robust structure and feature set. A detailed description of WSDL 2.0 is not the point of this article, but note that WSDL 2.0 is designed to

specifically be used by tool creators to automatically create "glue" to connect Web services to client code and infrastructure. Unfortunately, this can leave out a wide audience of Web services users that: are using emerging languages, platforms, and toolkits; do not use the exact same Web services stack; or are trying to develop their Web services contract-first.

Leveraging an understanding of all of this has particular

poignancy for Web services development and the community. This is because from a Web services creation standpoint, the common understanding of which service description language(s) are accepted and expected will likely change in the near future. On the consumption side of Web services, the need for an easy-to-use Web services description that lays out how to interact with the service will likely grow in importance.

If there is just WSDL 1.1 and WSDL 2.0 to deal with, can't a Web services supplier just deliver descriptions in both? Not necessarily, especially if the Web service is non-SOAP. In fact, the larger Web services community has been increasingly concerned about the issues of WSDL 2.0 complexity and the various shortcomings of WSDL 1.1. To that end, a variety of alternative Web services description formats has been proposed. Some of the formats are existing standards while some are ad hoc inventions designed to solve particular problems. Many of these are getting increasing attention and it's likely that Web services creators and users will both need to be familiar with them depending on the situations they face.

The State of Web Service Description Languages

At the time of this writing, there are at least 14 different generally known service description formats that can be used to describe Web services. Included in this list is the Web Service Description Language (WSDL) 1.1 (www.w3.org/TR/wsdl), WSDL 2.0 (www.w3.org/2002/ws/desc/), SOAP Service Description Language (SSDL) (www.ssd.org/), Really Simple Web Services (RSWS) (<http://webse.vices.xml.com/pub/a/ws/2003/10/14/salz.html>), Web Resource Description Language (WRDL) (www.prescod.net/rest/wrdl/wrdl.html), Norm's Service Description Language (NSDL) (<http://norman.walsh.name/2005/03/12/nsdl>), Simple Message Exchange – Descriptor (SMEX-D) (www.tbray.org/ongoing/When/200x/2005/05/03/SMEX-D), Web Application Description Language (WADL) (<http://weblogs.java.net/blog/mhadley/wadl.pdf>), Resedel (<http://recycledknowledge.blogspot.com/2005/05/resedel.html>), Resource Description Format (RDF) (www.w3.org/RDF/), RDF-Forms (www.markbaker.ca/2003/05/RDF-Forms/), Web Services Modeling Lan-

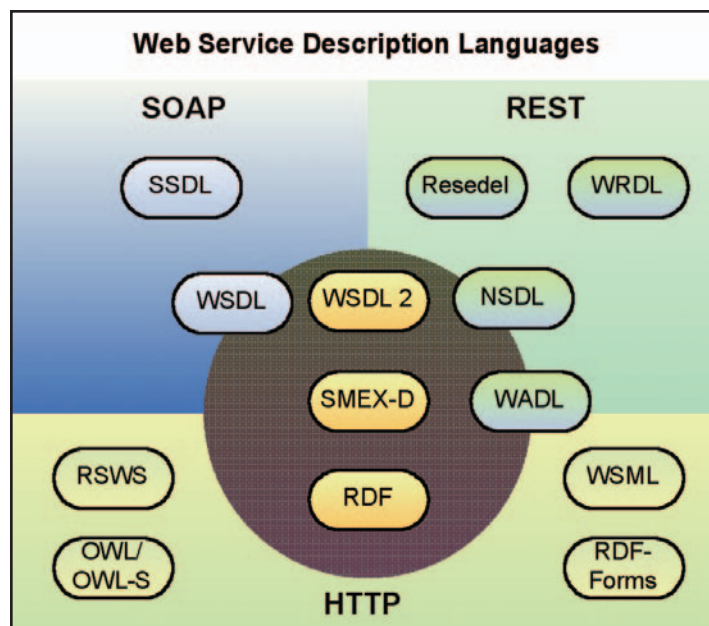


FIGURE 1 | A taxonomy for Web Service Description Languages based on the Web service model

guage (WSML) (www.wsmo.org/wsml/index.html), Web Ontology Language (OWL) (www.w3.org/TR/2004/REC-owl-features-20040210/), and the Web Description Language (WDL) (www.pacificspirit.com/Authoring/WDL/).

WSDL 1.1 and WDSL 2.0 are the W3C's answer to Web service description and WSDL 2.0 is actually quite capable and robust, but at over 100 pages for the Core Specification alone, WSDL 2.0 is considered too weighty for many development tasks.

SSDL is a fascinating and feature-rich Web service description language for sophisticated SOAP Web services and their interactions. It includes full support for an XML-based service description in Sir Tony Hoare's concurrency description language, Communication Sequence Processes (CSP) (www.usingcsp.com/),

tion types of REST Web services using XPath and even has a preliminary version in Python. WRDL is not generally accepted in the REST community yet.

NSDL is a creation of Norman Walsh and is a creation in direct response to particular difficulties the author had with WSDL. Designed to make the use of Web services straightforward for ordinary programmers while at the same time making it possible for Web service providers to describe interfaces in a standard way and easily achieve interoperability. NSDL created some waves in the Web services community and is a partial catalyst for some of the other Web server description languages on this list.

SMEX-D is described as the simplest possible description of an exchange of XML messages. The brainchild of XML cocreator

and NSDL and was created by John Cowan. It's still in the early stages of development.

RDF is the eminent grise of service description languages and was conceived by Tim-Berners Lee himself. While not specifically designed for either SOAP or REST, RDF is extremely powerful and will likely be the description engine that powers the forthcoming semantic Web, if and when it gets off the ground. At that point, or even before (which is the point here), it would make a lot of sense to describe Web services in terms of RDF.

RDF-Forms is a state machine-like description language can be used to describe how a service changes the state of information. Created by Mark Baker, RDF-Forms is highly regarded and while it's not an actual service description language, it is an effective way of describing how REST and some SOAP services actually operate. There is increasing talk of its being an effective model for incorporation into a service description language, or evolving into an SDL in its own right.

WSML is a sophisticated and complex Web services modeling language. It can form very detailed descriptions of set of Web services. The WSML working group is attempting to develop a proper formalization language for semantic Web services and a rule-based language for the semantic Web. While not in widespread use yet, WSML has a surprisingly extensive list of working tools, and even a full SDK. Large organizations may be able to use WSML effectively to create complete models of their enterprise service architectures. While they're overkill for any small development effort, languages such as WSML are getting traction in the formal methods and semantic Web community.

OWL/OWL-S is another branch of the semantic Web story. It's a full Web ontology language that is designed to describe the full meaning of Web content, including Web services. OWL uses XML Schemas, RDF, and RDF Schema, and adds a layer of vocabulary on top of it. While OWL is another language that is probably too heavyweight for many Web services developers and users today, its key position in the semantic Web indicates that reconciliation with it and the other Web service description languages will eventually have to happen.

WDL is a REST Web service description lan-

“ The right Web services description language makes it easy for a given set of clients and services to interoperate – and that's what Web services are really all about ”

as well as a Pi Calculus for describing the protocol framework. SSDL tools are in alpha stages now but there is already a processor for it in C# and in the .NET Framework 2.0.

RSWS is a minimalist answer to Web service description, and its name, Really Simple Web Services means what it says. Developed by Rich Salz, RSWS attempts to fit the Web service schema definition, interface definition, and location definition into the practical model that “avoids the excesses of WSDL.” RSWS's overall theme is to be good enough instead of comprehensive, and reusability and support for typeless schemas of any format (including non-XML) are listed as two major benefits.

WRDL is a REST service description language and hence has beaten WSDL 2.0 to the punch in the realm of REST-based description languages. WRDL describes the representa-

Tim Bray, SMEX-D is an exercise in elegance and simplicity and its entire description runs a mere three printed pages. Though it's safe to say it has some clout due to its creator, SMEX-D has only recently been getting attention, especially since it supports SOAP and REST and pretty much any other Web service model. No tools yet support SMEX-D but then again, they probably aren't needed.

WADL is an interesting and novel new entry into the world of Web services description. Designed specifically to provide the simplest possible alternative to WSDL, WADL is designed to describe simple XML/HTTP integration scenarios and Ajax clients. WADL uses a combination of textual description and XML schemas, and is the brainchild of Marc Hadley.

Resedel is another REST web service description language. It's a hybrid of SMEX-D

Description Language	Web Service Models Supported	Tool Support	Schema Support	Notable Strengths	Possible Weaknesses
WSDL 1.1	SOAP/HTTP	Extensive	XSD/Other	Universally recognized and accepted	Does not support REST or some other Web service models. Many WS-* protocols haven't reconciled their descriptions with it.
WSDL 2.0	SOAP/REST/HTTP	Initial	XSD/Other	Embraces a wide variety of Web service models. Likely to be the successor to WSDL.	Large and sophisticated; this may affect learning curves and adoption.
SSDL	SOAP	Alpha	XSD	Excellent for the most detailed and robust SOAP service description.	Only supports SOAP, and has limited tool support.
RSWS	HTTP	None	XSD/Relax NG	Simple and easy to learn.	Non-standard, not well known, and no tool support.
WRDL	REST	None	XSD	One of the few REST description languages.	Limited tool support, only describes REST.
NSDL	HTTP/REST	None	XSD	Supports a variety of non-SOAP Web service models and is simple.	No tool support, not well known.
SMEX-D	SOAP/REST/HTTP	None	XSD/Relax NG	Supports most common Web service models. Very simple.	No tool support, still experimental.
WADL	HTTP	None	XSD	Designed especially for HTTP.	Doesn't explicitly support SOAP or REST.
Reseddel	REST	None	XSD	Solid REST support.	No SOAP support.
RDF	SOAP/REST/HTTP	Limited	XSD	Well known and has plenty of tool support. Provides ability to plug Web services into the Semantic Web.	Not really designed for describing Web services.
RDF-Forms	REST	Limited	XSD	Not really a description language, but a useful change description language for REST.	Needs reconciliation with other Web service description languages.
WSML	SOAP/REST/HTTP	Initial	XSD	Formal and sophisticated, with tool support.	Overkill for many applications.
OWL/OWL-S	SOAP/REST/HTTP	Limited	XSD	Adds additional semantic meaning to Web services.	Standardized but not well known or understood.
WDL	REST	None	XSD	Very capable REST description language	No support for SOAP model, little tool support yet.

TABLE 1 | Comparison of Web Service Description Languages

guage and targets a different way of describing things than does WSDL 2.0. Specifically, WDL assigns operations to resources and actually leverages all of the strong capabilities of WSDL 2.0 such as schema and operation definition, URI construction, and configuration of HTTP parameters.

Conclusion: Where Web Service Description Is Heading

There is little doubt that the landscape of Web service descriptions will grow increasingly crowded and complicated in the next few years. It's also quite likely that some of these methods for Web service description will become de facto and accepted alternatives to WSDL and the other standards-based description languages. This isn't necessarily bad since it means the right tools are available for the job of building

or using Web services. The right Web services description language makes it easy for a given set of clients and services to interoperate – and that's what Web services are really all about. This means Web services creators and users will need to continue to track the prevailing winds and offer descriptions of their Web services in the most commonly requested formats.

On the standards front, WSDL 2.0 will bring important and needed change to the industry as it provides a wall that fewer can scale, and forces some to take a serious look at credible alternatives, particularly for heterogeneous applications and lightweight nonstandard Web services. Unfortunately, there are few that are clear alternatives today, leaving Web service description very much in a rebuilding year. Hopefully 2006 will bring us reconciliation and resolution as more practical experience

and improved solutions resolve the issues surrounding Web services description. ©

About the Author

Dion Hinchcliffe is cofounder and Tier-I Architect for the premier enterprise architecture firm Sphere of Influence Inc. (www.sphereofinfluence.com/), founded in 2001 in McLean Virginia. A veteran of software development, Dion works with leading-edge technologies to accelerate project schedules and raise the bar for software quality. He is highly experienced with enterprise technologies and he designs, consults, thinks, and writes prolifically. His latest work is with enterprise-scale service-oriented architecture, where he blends advanced standards-based engineering with Agile Development Processes. Dion enjoys all of the major platforms, ranging from UNIX, Linux, and Java to Microsoft .NET and SQL Server. He maintains an active blog (www.hinchcliffe.org/default.aspx) where industry-shaping technologies are discussed in depth.

■ ■ ■ dhinchcliffe@sphereofinfluence.com

Can Your SOA Make You Money?

Reuse of services + agility = ROI

■ Why do we do what we do? I mean, why do we design and implement SOAs?

The truth is we do so to improve our business, thereby making it more adaptable and ready to accept change without major disruptions. However, what does this mean to the bottom line?

These days, IT architects and business people must work hand-in-hand to determine if changes proposed are the proper course for a business. Indeed, the application of an SOA has different degrees of ROI, depending on the problem domain. The cost of implementing an SOA should be directly related to the benefits to the business, in both hard and soft dollars. Let's explore this.

Why We Service-Enable

We implement SOA for two major reasons. First is the ability to save development dollars through *reuse of services*. These services may have been built inside or outside of the company, and the more services that are reusable from system to system, the more ROI we get from our SOA. Second is the *ability to change* the IT infrastructure faster to adapt to changing needs of the business. This, of course, provides a huge strategic advantage and thus allows for the business to have better chances of long-term survival. While determining the ROI on agility is difficult to figure out in hard dollars, we know the value is there.

Reuse of Services

Under the heading of service reuse, we have a few things we need to determine



WRITTEN BY
**DAVID
LINTICUM**

to better define the value. These include:

- The number of services that are reusable
- Complexity of the services
- The degree of reuse from system to system

The *number of reusable services* is the actual number of new services created, or, existing services abstracted, that are potentially reusable from system to system. The *complexity of the services* is the number of functions or object points that make up the service. We use traditional functions or object points as a common means of expressing complexity in terms of the types of behaviors that the service offers. Finally, the *degree of reuse from system to system*

is the number of times you actually reuse the services. We look at this number as a percentage.

Just because we abstract existing systems as services, or create services from scratch, that does not mean that they have value until they are reused by another system. In order to determine their value we must first determine the Number of Services that are available for Reuse (NSR), the Degree of Reuse (DR) from system to system, as well as the Complexity (C) of each service, as described above. The formula to determine value looks much like this:

$$\text{Value} = (\text{NSR} \times \text{DR}) * C$$

Thus, if you have 100 services available for reuse (NSR=100), and the degree of reuse is at 50 percent (DR=.50), and complexity of each service is, on average, at 300 function points, the value would look like this:

$$\text{Value} = (100 \times .5) * 300$$

or

$$\text{Value} = 15,000, \text{ in terms of reuse}$$

If you apply the same formula across domains, with consistent measurements of NSR, DR, and C, the relative value should be the resulting metrics. In other words, the amount of reuse directly translates into dollars saved. Also, keep in mind that we have to subtract the cost of implementing the services, or of creating the SOA, since that's the investment we made to obtain

“ The cost of implementing an SOA should be directly related to the benefits to the business, in both hard and soft dollars ”

the value.

Moreover, the amount of money saved depends on your development costs, which vary greatly from company to company. Typically, you should know what you're paying for, functions or object points, and thus it's just a matter of multiplication to determine the amount of money we are saving by implementing a particular SOA.

Agility

Agility is a strategic advantage that is difficult to measure in hard dollars, but not impossible. We first need to determine a few things about the business, including:

- The degree of change over time
- The ability to adapt to change
- Relative value of change

The *degree of change over time* is really the number of times over a particular period that the business reinvents itself to adapt to a market. Thus, while a paper production company may only have a degree of change of 5 percent over a five-year period, a high technology company may have an 80 percent change over the same period.

The *ability to adapt to change* is a number that states the company's ability to react to the need for change over time. The notion is that the use of an SOA provides a better ability to change IT to adjust to needed changes in the business.

Finally, the *relative value of change* is the amount of money made as a direct result of changing the business; for instance, a retail organization's ability to establish a frequent buyer program to react to changing market expectations, and the resulting increases in revenue from making that change.

Determining an SOA's ROI is not an exact science, but with some analysis and some realistic data points, you can figure out how much value your SOA implementation has brought you, or will bring you. Again, we need to cost justify the use of this approach and technologies, and the information presented here should help you along the road to creating your own business case. ☺

■ 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). You can reach David at Linthicum@att.net.

■ ■ ■ Linthicum@att.net

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

FOCUS: .NET

The Momentary Enterprise

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.

Web Services Assurance for Insurance

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.

Best Practices and Solutions for Managing Versioning of Web Services

Service-oriented architecture and Web services are being critically considered by most organizations today in some form or another. The adoption of SOA and Web services has taken up 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.

Does Your SOA Achieve 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?

WebServices 
.NET J2EE XML JOURNAL

Autonomic SOA

Achieving fully “business-conscious” IT systems

■ Service-oriented architectures (SOA) and autonomic computing are among the hottest topics in IT today. SOA simplifies integration and facilitates the componentization of enterprise-wide systems, thereby enabling optimal business agility. Autonomic computing allows these systems to operate without human intervention – through self-configuring, self-healing, and self-managing capabilities. By combining autonomic computing with SOA, enterprises can now achieve a new IT utopia, named “Autonomic SOA.” However, this new level of autonomic computing goes beyond just reacting to traditional IT issues, such as increasing the speed to diagnose, repair, and prevent future issues. Autonomic SOA is also able to respond dynamically as changes occur within business conditions and processes, thereby empowering enterprises everywhere to achieve flawless execution on a daily basis.

A Healing Touch: Autonomic Computing

A useful analogy for understanding autonomic computing is the human nervous system. Autonomic controls in the human nervous system send indirect messages that regulate temperature, breathing, heart rate, and digestion without conscious thought. If your core body temperature rises too high your autonomic systems cause your sweat glands to secrete and cool you down automatically and involuntarily. By analogy, an autonomic computing system keeps core IT systems functioning at a basic level. If the CPU of a particular system appears to be overburdened, or another runs out of disk space, the autonomic system adapts to the situation by adjusting and reallocating available resources to maintain the system's equilibrium.

An autonomic IT infrastructure is made

WRITTEN BY



ARTHUR
MATEOS



JOTHY
ROSENBERG

up of a network of organized, “smart” computing components that give us what we need, when we need it, without a conscious mental or physical effort. It gives us systems that look alive and think they are alive. However, in order to fully leverage these interactions, you must have the ability to gain the intel-

ligence behind the autonomic components – you must go beyond the window dressing or the outer layer. Only then will you have a truly fully functional IT system.

So while we can now see how powerful autonomic IT systems are and how much easier they can make the life of the enterprise IT organization, we can't stop there; if we did, business would never progress, never grow, and never achieve optimal execution. Why not strive for full business consciousness? Where did the business failure occur and why? How can I improve my business

execution? Is the answer in the IT infrastructure or is it in the business processes and the humans who interact with them?

Danger Zone: The World of SOA

SOA is the de facto standard for enterprise integration today. Adopting SOA gives businesses many benefits. From an IT perspective these benefits include lowered cost and complexity of integration as well as platform and technology independence. Standardization and reuse enable the business to be more agile in flexibly connecting its IT resource silos into composite applications, which can improve business process efficiency.

However SOA itself presents some significant challenges that need to be addressed by an autonomic approach. SOA is defined by loose coupling where messages and the exact format of those messages between service nodes can be specified at run time. SOA implies extensive distribution and scale because it links systems all across the enterprise and runs processes that reach every department in the organization. Over the years, corporations have developed fiefdoms, islands, stovepipes, or whatever you want to call them; they all represent communication and control challenges. SOA operates in the virtual world and that is a world that is even riskier and more threatening than the physical world. It's a world based on high-performance computing devices so things happen fast (including bad things), and that can leave little time for analysis and thought. All of this can, in the end, have a bad impact on the business users who are supposedly served by these leading-edge systems.

Real-World Business SOA

In SOA, business information flows between service nodes. Because interfaces are published and the flows are XML-based the business information itself is now accessible and actionable. SOA gives us the pathways and connections that move critical business messages between systems and applications that need to do the real work of business processes. Recalling our previous nervous system analogy, this is like the message from our brain telling our arm to move the steering wheel to the right. Hand-eye coordina-

tion and feedback pathways allow us to drive a car, and the combination of monitoring (our eyes), full visibility of everything that is happening (all of our other senses), and analysis (understanding and interpretation) makes sure that we stay on the road.

What's Inside: Autonomic SOA Characteristics

SOA is the bridge between IT and business. Transparency applied to SOA means being able to tell the difference between “is alive” in the context of the business and just “looks alive” in the IT context. Furthermore, combining autonomic IT and SOA allows us to determine if it's a business problem or an IT problem that needs to be fixed. Businesses need to bring together SOA for business with an underlying autonomic IT infrastructure. In order to know that from the business effectiveness viewpoint, all is well and executing optimally and to be able to make conscious, effective business decisions – that is, keeping the business in homeostasis – businesses need both. We call this combination of autonomic IT with SOA, *autonomic SOA*.

An autonomic SOA must have the following core characteristics: flexibility, accessibility, and transparency.

Flexibility – Autonomic systems must be able to sift through data in a platform- and device-agnostic manner. In SOA composite applications, data is in the form of messages that are passing from one service node to another. In those messages resides critical data that is driving the execution of the processes that control the business. If you can understand what is in those messages, then you've unlocked a more in-depth understanding of your business. When you act on that business information, you're empowered to handle risk quickly and proactively. You'll even have the power to aggressively seize business opportunities and in the process achieve optimal business execution.

Accessibility – Quite simply this means that the systems are always on. That has to mean more than that they're just powered up, and more than that electricity is running through the internal transistors and running the operating system. It should mean always on from a full business-execution perspective. Are the correct messages being received

and sent? Are they following the correct syntax and semantics? Are they supporting the business process they are supposed to support? Are those business processes themselves always on from end to end, so a business interaction with customers always works as expected to their full satisfaction?

Transparency – Usually this means that the system interacts with users and performs its functions completely while shielding (and not burdening) users from the intricacies of

The major keys to fully realizing autonomic SOA are deep visibility, a holistic business context, a nonintrusive light touch, accessibility and analysis, and self-managing feedback loops.

Deep visibility – Any and all business information that is contained in messages that flow between service nodes in composite applications must be easily accessible and prone to analysis. When margin is the most critical business question, deep visibility

“ The autonomic movement enables us to focus on what really matters in IT systems: the information ”

how the system does its job. Of course this should be true regardless of whether you examine this at the IT level or whether you look more at the business level where the composite applications operate. In the realm of SOA, transparency takes on even more significance: transparency is our eyes and ears to what is happening in the business right as it happens, with enough intelligence applied to our observations that we know what to do with what we see.

Designing Autonomic SOA Systems

Autonomic SOA systems must be designed to harness real-time information, so as to empower immediate access for making critical business decisions. Gone are the days of stale computational data. Immediacy and appropriateness of data for its audience are now the critical measures of business success. SOA makes composite business applications out of services that are linked together through in-flight business messages. This information-in-motion was previously invisible. It is precisely this information through which valuable business insights are to be gained. This information-in-motion is in play; it's “right-now” business information as it is happening, and it is critical for success with SOA.

could mean full access to understanding the number of items ordered and their corresponding prices. In another business case, where fraud detection is the objective, it may be more important to have user names, items that were attempted to be ordered, prices, and credit card numbers.

Holistic business context – This is required so that you can see end-to-end how all of the messages that are crisscrossing an SOA composite application form a business process. Analyzing how that business process does or does not align with the business's goals will be critical. Are business managers who are involved in approving large expenses doing their job effectively? Or, are expenses being rubber-stamped and approved without adequate due diligence?

A nonintrusive light touch – You cannot afford to perturb the very messages that need to be observed. Fortunately the underlying infrastructure makes it possible to get directly into the stream of messages flowing into and out of service nodes. This has to be done efficiently and the services that reside on each node need to be discovered automatically as do their message formats and meanings.

Accessibility and analysis – Autonomic SOA requires access to and analysis of the business information at the speed business occurs to support business decisions that can

impact the course of business before it is too late. Discovering that there was a major fraud incident or a breach of compliance in a day or two will not protect the business from major punitive regulatory punishments.

Self-managing feedback loops – Finally, autonomic SOA requires self-managing feedback loops, which are self-limiting and configurable. This means that knowing what is going on in the context of the business is a good first step, but more importantly, something proactive must be done with that knowledge. It won't always be acceptable to directly feed the results back into the business because sometimes human intervention is critical, so feedback loops must be configurable.

An Omniscient IT Presence: Autonomic SOA Nirvana

What would it mean for businesses to have completely flexible, self-healing, self-aware, self-protecting, self-adapting, and self-improving composite applications? What follows are

some real-world scenarios of what this nirvana would bring to any enterprise IT infrastructure.

The autonomic SOA sees everything. When something goes wrong with the business it alerts its "brain," which may be an application designed to make decisions and take action or it may be humans who need to assess the situation before action is taken. The system is smart and only alerts once per type of incident when patterns are detected or ranges are exceeded. Security violations, attempted fraud, compliance violations, product margins going negative, production problems, stock outages, or any other threat or risk to the business will be handled smoothly in this fashion.

When attempted fraud is detected, this self-protecting autonomic SOA system will notify security systems that disable the account of the fraud perpetrator. When margins are detected to be shifting from positive to negative because volumes of high margin products in the mix are lower than forecasted, this self-improving system will change the product

mix and start shifting towards lower margin but higher volume products in the mix.

The focus of autonomic computing is primarily on the IT Infrastructure. To help organizations solve their critical pain of managing far-flung networks of systems, devices, and software, autonomic computing promises a network of components that self-manage. The actions taken when issues are detected include provisioning, de-provisioning, fail-over, reroute, and so forth. However, for business to fully realize the benefit of the autonomic movement, the emphasis needs to shift to the *information*. In SOA that information is information-in-motion flowing through SOAs.

As enterprises realize the enormous flexibility and cost benefits of SOA, they are building critical new business applications out of reinvigorated legacy components and reusing existing networked resources. A few examples will help crystallize this discussion.

Reseller channel margin management is a critical business process for a company in the highly competitive market of selling computers and peripherals to consumers through a large reseller channel. Deals are carefully constructed based on assumptions of volumes that will be achieved. This in turn leads to pricing certain elements of deals at below cost (negative margin), assuming the high margin elements of those deals will overcome the loss leaders and make the entire package a profitable transaction. If volumes do not reach assumed levels, the entire deal remains at negative margin and damages the business quickly as it loses money. The business needs to see volume misses happening and make a change immediately. Just as in the case of a race car driver who sees the curve coming as his competitor comes up on the inside, a business needs to compensate with a change in steering and acceleration to beat competitors around the curve. Our computer manufacturer depended on their autonomic systems for continuous self-healing infrastructure to support their far-flung reseller network, but it was their SOA approach with the ability to leverage information-in-motion that gave them the ability to optimize their business performance and subsequent profitability.

Credit card fraud detection and response is a critical business process for online merchants who, because card is "not present" are responsible for any fraudulent transactions below the \$50 credit card company limit. Increasingly, low-dollar fraud based on stolen credit cards is becoming prevalent because sites such as eBay make it easy to move large quantities of goods. It's necessary to detect patterns of fraud as fraud happens. This can be done by looking for instantaneous spikes in revenue that are all coming from a single dollar amount, then examining credit card numbers at that specific dollar amount to see if it is one or just a few different numbers, in which case it is probably fraud. Our credit card merchants have underlying autonomic IT systems on top of which their SOA-based



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 New York City on October 15 and 16.

www.weekendwithexperts.com

composite application allows them to deal with fraud as it happens.

It's the Information, Stupid!

The autonomic movement enables us to focus on what really matters in IT systems: the information. SOA, with its messages passing between nodes in composite business applications, puts a fine point on the matter. However autonomic IT systems, while they're a huge boon for IT systems and their managers, require more. To fully realize that potential enterprises must combine the enormous business benefits of SOA with underlying autonomic IT computing systems. The resulting *autonomic* SOA systems will give you fully business-conscious IT systems, thus enabling you to focus on your business. This is how you will finally achieve a revolution in business insight and predictability that will ultimately achieve what every business wants most: excellence in execution. Best of all, this is realistic and achievable today.

“ Immediacy and appropriateness of data for its audience are now the critical measures of business success ”

About the Authors

Dr. Arthur Mateos is the vice president of products at Boston-based Service Integrity (www.serviceintegrity.com), a provider of real-time Business Intelligence (BI) software for service-oriented architectures (SOA). Before cofounding Service Integrity, Dr. Mateos served as senior product manager at Inktomi. In this position he was responsible for Content Distribution (CDS) and Wireless Messaging product lines. Under his direction, the CDS product line was instrumental in driving more than \$60M in revenue. He received a BA in Physics from Princeton University and a PhD in Nuclear Physics from MIT.

■ ■ ■ arthur.mateos@serviceintegrity.com

Dr. Jothy Rosenberg is a strategic advisor and cofounder of Service Integrity. He is currently vice president of software at Ambric, a fabless semiconductor company. Dr. Rosenberg also cofounded GeoTrust, the world's second largest certificate authority and a major innovator in enterprise managed security solutions. He is also the author of *How Debuggers Work* and *Understanding Web Services Security* (2003; Addison-Wesley). Jothy holds patents on watchpoint debugging mechanisms, content certification and site identity assurance, as well as a pending security compliance monitoring patent.

■ ■ ■ jothy@ambric.com

Looking to Stay Ahead of the i-Technology Curve?

Subscribe to these **FREE** Newsletters >

Get the latest information on the most innovative products, new releases, interviews, industry developments, and i-technology news

Targeted to meet your professional needs, each newsletter is informative, insightful, and to the point.
And best of all – they're FREE!

Your subscription is just a mouse-click away at **www.sys-con.com**

IT solutions
JOURNAL

NET JOURNAL

JAVA DEVELOPER JOURNAL
JDJ

WebServices
JOURNAL

information
STORAGE +
SECURITY
journal

LINUXWORLD
MAGAZINE

wireless
BUSINESS TECHNOLOGY

LINUX BUSINESS
WEEK

XML JOURNAL

MX
developer's journal

WebSphere
JOURNAL

wldj
JOURNAL

ColdFusion
Developer's Journal

SYS-CON
MEDIA

The World's Leading i-Technology Publisher

Web Service Invocation Framework

Adding wheels to Web services

■ The open source initiatives have transformed the technical landscape by leaps and bounds in the past two decades. It has gone a long way toward breaking the stranglehold of monopolistic software companies over technologies. When programmers have the ability to read, redistribute, and modify the source code for a piece of software, the software evolves faster and better. It happens at a speed that is way faster compared to the slow pace of conventional software development.

Web Services and Open Source

Today there is a glut of choices for developers of software. The choices extend from choosing operating systems to application development framework components. The business enterprises are now harnessing the power of open source by using ready-made open source components and frameworks. This helps the enterprises save on development costs and also cut down on project implementation time.

The introduction of Web services has now provided a way of integrating applications based on open standards seamlessly across technologies. The emergence of Web services can primar-

WRITTEN BY
ANSHUK PAL CHAUDHARI,
SANDEEP GAIKWAD,
AMBAR VERMA,
& V. NIRANJAN

ily be attributed to the open source community. It also has brought the leading technology vendors across the spectrum, endorsing and supporting open source in a big way.

The Need for a Web Services Invocation Framework (WSIF)

Currently the programming models in the J2EE framework support synchronous and asynchronous invocations. However this is accomplished by using two different programming models, shown in the subsections below.

JAXM (Java API for XML Messaging)

JAXM provides a rich set of interfaces for

implementing message-oriented asynchronous invocation of Web services. One of the bigger advantages of JAXM is that it does not limit message standards restricted to SOAP. Hence we can have a rich set of message standards and have different message approaches that can include vertical specific schemas like Accord for insurance, FiXml for the financial industry, etc. However JAXM lacks fully formed asynchronous message profile support and requires a domain-specific profile.

JAX-RPC for RPC-style synchronous communication

JAX-RPC is now inherently embedded as a part of a J2EE specification. It is very easy to develop potentially complex RPC-based Web services. However, RPC-based Web services lead to tight coupling between the service providers and the consuming clients. RPC basically supports a synchronous model where the client has to wait until the execution of the remote procedure is completed.

This necessitates a developer to have knowledge of two different sets of APIs, so this fuelled the need for a framework that would be an abstraction layer above the various programming models. Thus this abstraction layer will be responsible for the invocations across varied bindings. This also has the ability to future-proof the enterprise services against change in the SOAP API, for instance, Apache SOAP to Apache Axis.

Say, for example, we have a very intricate software system composed of various chunks of software – message-driven beans, mainframes, Java classes, JMS, and Web service. The objective now is to write an application that will in turn utilize all of these chunks of software and accomplish some result. The moment we take that application to a different server, the complete code breaks; furthermore, there is number of issues associated with this situation.

WSIF fixes the problems by allowing WSDL to be the normalized description of the complete software system and will allow us to access that system independently of the protocol and the location. So it may be a SOAP, an EJB, or JMS – we have an API centered on WSDL that we utilize to access the functionality. The separation of the API from the actual protocol also means we have flexibility – we can switch protocols, location, etc. without having to even

“ The emergence of Web services can primarily be attributed to the open source community ”

recompile the client code. So if an available EJB becomes available as a SOAP service, we only need to change the service description (the WSDL), without having to make any changes in applications that use the service.

The motivation for WSIF is that we wanted to see the “services-oriented architecture” become more extensive than just SOAP. There are a number of different protocols, transports, and distributed computing technologies that offer more than SOAP does today – especially in terms of management, transactions, security, and other quality of service (QoS) features.

Therefore the primary goals of the WSIF are:

- To have a binding-independent mechanism for Web service invocation
- To prevent clients from experiencing the hassles of binding to a particular protocol for a Web service
- To allow dynamic selection between multiple bindings to a Web service
- To facilitate the development of Web service intermediaries

Introduction to WSIF

WSIF is a toolkit that provides a simple API for invoking Web services irrespective of their technical bindings. WSIF provides a way to call any service regardless of the transport protocol or where it is located. It allows the description of the application code, which might be Simple

Java Classes or Enterprise Java Beans or Java Messaging Service or Web service, etc. in the Web Service Description Language (WSDL). The WSIF API allows clients to invoke services that focus on the abstract service description. In a WSDL the service is described in terms of the following parts:

- **Port Type:** Defines the abstract interface of the service
- **Binding:** Defines how to map among the abstract interface, a service format, and the underlying protocol
- **Message:** Defines the request and the response of the corresponding methods and services
- **Port:** Defines the actual endpoint where the service is located, e.g., the implementing class for Java Binding

WSIF Architecture

The main components of the WSIF architecture are as follows:

- WSIFProvider
- WSIFServiceFactory
- WSIFService
- WSIFPort
- WSIFOperation
- WSIFMessage

The basic idea is to invoke the service by using the abstract description of the WSDL. It is

carried out in the following steps:

- Loading of WSDL Document
- Creation of port factory for the corresponding service
- Port factory is being used to receive the port
- Creation of message
- Invocation of the service by supplying the port with the operation name

The following descriptions pertain to Figure 1:

- **WSIF Service:** The WSIFService interface is used for creating an instance of the WSIFOperation interface to use for a specific invocation of a service operation
- **WSIF Operation:** The run-time illustration of an operation, called WSIFOperation, is used for invoking a service based on a particular binding
- **WSIF Provider:** A WSIF provider is an implementation of a WSDL binding that can run a WSDL operation through a binding-specific protocol. WSIF includes SOAP providers, JMS providers, Java providers, and EJB providers – and it can include all of those concurrently, if required.

WSIF Binding Examples

General Prerequisites

www.apache.org/dyn/closer.cgi/ws/wsif/

- wsif.jar
- wsif-j2c.jar

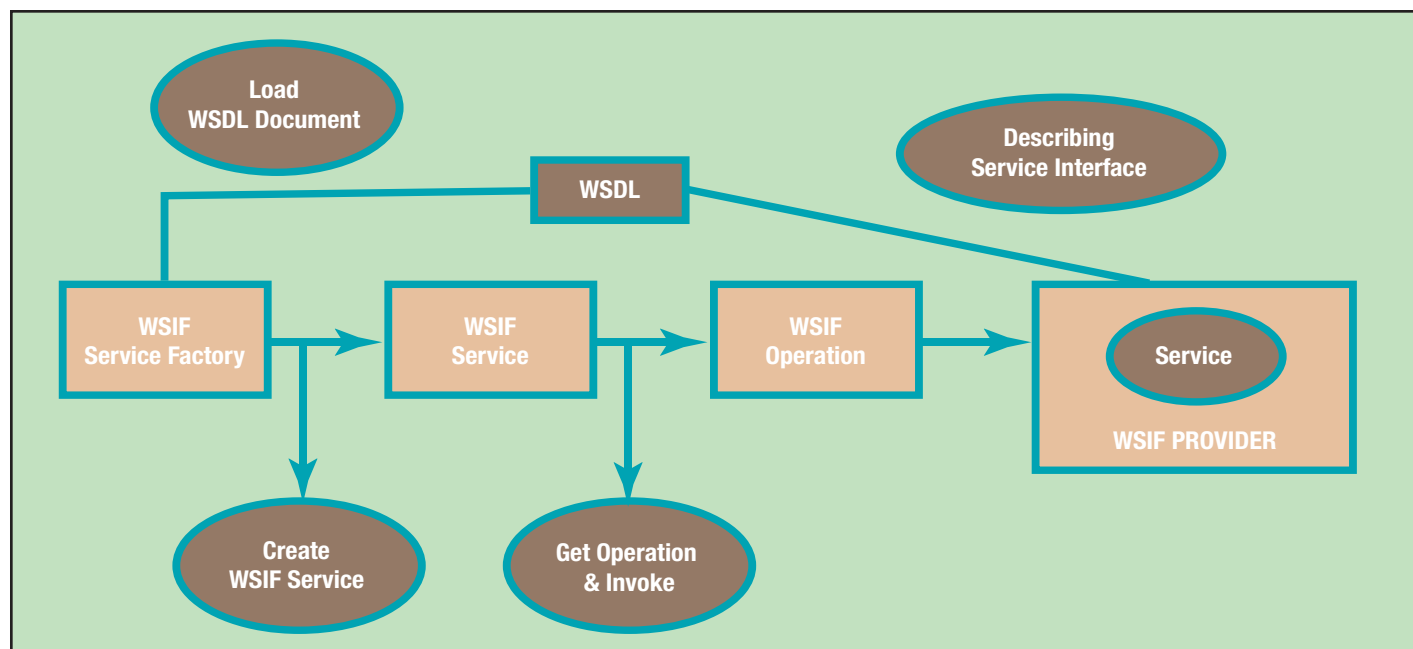


FIGURE 1 WSIF architecture

We need to just add them in the classpath and we are ready to use WSIF.

Java Binding

Description

We will now demonstrate invoking a simple Java Class as a Service using the WSIF APIs.

- We have a Java Class called as CalculatorImpl, which has a method of simple addition.
- Now since we are calling the Java class as a service, we also need to change the WSDL accordingly. All elements of the WSDL remain the same, only the binding tag is changed to indicate that we are using JAVA Binding.

tion of an EJB as a service.

- The EJB that we have developed is used to insert values in an Oracle database
- We have deployed the EJB on the WebLogic Server 8.1
- The binding element in the WSDL indicates that it is EJB binding. The format:typeMapping and ejb:operation elements remain the same as in the case of Java binding
- To specify an EJB object as an endpoint we modify the ejb:address as shown below. The className (optional) attribute gives the fully qualified name of the home interface class of the EJB, while the jndiName attribute specifies the name under which

Client, but why would a client in Java access the service via SOAP binding and not via simple Java binding?

- For the Java Client it is always better to access a Java Class than to access a Web service through a SOAP binding.
- WSIF provides the option of exposing the functionality of a service through multiple bindings, and that too in single WSDL file. Let's see how.
- We have a simple Java Class, which we expose as Web service on WebLogic Server 8.1.
- Now we will describe the same class in a WSDL as SOAP binding (Web service) as well as a Java binding (simple Java Class). Please refer the binding element of MultiBinding.wsdl.
- Now it is up to the client what port to use at what time. If we notice the service element of the WSDL carefully, it shows one service whose functionality with multiple bindings, for instance the Dot Net Client, will be using the SimpleSumServiceSOAPPort, and the Java Client will be using the SimpleSumServiceJavaPort. Let us see the client for the invocation of multiple bindings. Please refer the Dot Net Client "Form1.cs" (source code is available by viewing this article in the WSJ archives, <http://webservices.sys-con.com/read/issue/archives/>, Vol: 5 Iss: 9).

Here WebReference is a Dot Net Skeleton that contains the method of the Web service. In case of the Java Client, we have to use the executeRequestResponseOperation specific to the WSIF Framework.

Advantages

The advantage here lies in the fact that we can have multiple bindings for a single service. The client finds a suitable binding for them and then invokes the service. It is also convenient for building the client on any platform and to access the service.

JMS Bindings

Description

The following service invocation aims at using the message-oriented middleware using Java Messaging Service (JMS) through WSIF Apis. We only want to demonstrate that it is possible to bind JMS using WSIF.

- We have developed a Message Driven Bean

“ WSIF provides a way to call any service regardless of the transport protocol or where it is located ”

- A mapping from the abstract types used in WSDL message parts to Java types that represent the same information is achieved by the "format: typemapping" element.
- To represent Java method in a WSDL in an abstract way the "java: operation" element is used
- To represent a Java object as an endpoint for a service available through a Java binding we use the "java: address." The className attribute is used to give a fully qualified class name to be used for service invocation. Please refer the binding element of "Addition.wsdl."
- Now we take a look into the changes in the client for invoking the simple addition service. We call the method executeRequestResponseOperation, which is method from class WSIFOperation_Java, This class implements the WSIFOperation interface. The method contains three parameters (input, output, and fault), which are objects of the WSIFMessage interface. Please refer the "getAdditionResult" method of Run.java.

this EJB can be looked up in a JNDI context. Please refer "EJBsSample.wsdl"

- As we can see even in this case, it is the WSDL that is changed; the binding name has changed and we have different attributes for the EJB address, which is the endpoint for the service
- For the client, refer to the Run.java of the EJB binding source code

Multiple Bindings

Description

The following service invocation aims to showcase the most important concept of WSIF – multiple binding and protocol with the same WSDL file. Let's look at an example:

- We have a service, for instance, a Calculator. Now it is quite possible to have multiple clients who need to access the service, for instance, a "Dot Net Client" and a "Java Client."
- The Calculator application has been developed in Java.
- Now the only way for the Dot Net Client to access that service Calculator is only when the functionality is exposed via SOAP binding. SOAP binding is fine for the Dot Net

Enterprise Java Beans (EJB) Binding

Description

We will now try to demonstrate the invoca-

(MDB) as a service that has a method that just appends "Hi" to whatever the user enters.

- The MDB is being deployed on WebLogic Server 8.1.
- The main focus of the service is the JMS binding.
- The JMS binding in this case uses JMS text messages for communication
- The WSIF API provides the translation of the invocation to the sending of JMS text message to the particular queue. The specific part of WSDL that deals with the JMS binding is shown below. Please refer the binding element of "JMS.wsdl."
- The service in the WSDL also describes the different parameters that need to be given in order to look up the Queue. The Client for calling the MDB is also a very simple one. Please refer the client "Run.java" of JMS binding.

Advantages and Disadvantages

Advantages

- WSIF allows multiple bindings
- It solves integration problems of various J2EE components using WSDL extensions.
- It doesn't require an abstract description such as CORBA objects at the time exposing it, so it follows most of the rules related to Web services
- WSIF forces the programmer to hide the complicated details from the one who accesses it, as it forces the programmer to write its own WSDL
- Services can be used either by a set of stub classes (static) or by a dynamic interface invocation (dynamic)
- It allows the developer to work with the same programming model regardless of how the service is implemented and accessed
- It enables the developer to move away from the native APIs of the underlying service, and to interact with representations of the services instead
- Also protocols, location, etc. can be changed without having to recompile the client code

Disadvantages

- WSIF also follows the discovery model followed by Web services, so it also suffers

from the performance degradation due to its reliance on XML in discovery

- Centralized registry is for registering all of the new bindings done in WSIF, so this architecture is not good for mobile computing because it would require an accessible registry in every wireless network
- This framework mainly depends on WSDL but WSDL extensions are not open standards, so providers might use multiple specifications of the same binding type, which also causes new interoperability problems
- Tools required for generating stubs and skeleton, which handle asynchronous transport, are not available with WSIF
- It doesn't have built-in support for transactions and security

Summary

We have witnessed that in order to handle different protocols, WSIF implements an extensible framework. It permits new modules to be added to a client runtime to sustain additional protocols. These new modules – WSIF Providers, handle all possible service requests for a given protocol. WSIF currently has providers for SOAP/HTTP, SOAP/JMS, EJB using RMI-IIOP, native JMS messaging, and legacy systems using the J2EE Connector Architecture to access systems, including CICS and IMS. In all of the aforementioned cases, we can see that the WSDL that we are writing keeps on changing according to the binding, but each time the client only has a minor change. The WSIF is just creating a layer on top of the application layer and keeping the WSDL as the centerpiece of information. It would have been better if there had been a way of generating the WSDL depending on what type of binding and protocol the client requires. ©

About the Authors

The authors are interning and/or working as part of the Web Services COE (Center of Excellence) for Infosys Technologies, a global IT consulting firm, and have substantial experience in publishing papers, presenting papers at conferences, and defining standards for SOA and Web services. The Web Services COE specializes in SOA, Web services, and other related technologies.

- ■ ■ anshuk_palchaudhuri@infosys.com
- ■ ■ sandeep_gaikwad@infosys.com
- ■ ■ ambar_verma@infosys.com
- ■ ■ niranjan_v@infosys.com

THREE REASONS TO

blog-n-play.com

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

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

2 Own a most prestigious blog address!

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

3 Best blog engine in the world...

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

www.TAMI.linuxworld.com

"Many blogs to choose from"

PICK YOUR MOST PRESTIGIOUS ADDRESS

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

3 MINUTE SETUP

Sign up for your FREE blog Today!

i-Technology Blogs Read by Millions beta

— This site will go beta February 18, 2005.

SOA in Action

Experiences with caching, transactions, and security in a highly distributed, networked SOA in the travel and leisure industry

■ LibGo Travel, one of the largest privately held travel companies in the U.S., provides vacation packages through its retail stores and wholesale distribution channels to consumers, partners, travel agents, and stores. The company wanted to expand its offerings by adding dynamic, branded, and personalized packages. To help execute this idea, LibGo had to bring together our travel partners, including airlines, hotels, and travel aggregators, as well as LibGo Travel's existing heterogeneous systems environment. As a result, LibGo's Next-Generation Travel System (NGTS) is among the most sophisticated booking systems that are currently being implemented. Instead of building one-off interfaces for each partner – a time-consuming, expensive, and brittle solution – LibGo adopted a modern SOA with shared business services and Web services: data interchange would be XML-based, and WSDL would be the single interface definition standard.

This report from the trenches focuses on our experience of architecting NGTS as a large-scale composite application that is able to deliver more than a million transactions per day. In the process we mastered performance, transactional semantics, and the security challenges that are unique to an SOA environment, which is the focus of this article. The software architecture is shown in Figure 1.

Tracing a Customer Order

The heart of LibGo's booking system, which sits behind the portal, is NGTS, a J2EE-based composite application that ties together a plethora of local back-end systems and remote partner systems, all based on different technologies but accessed through service inter-

WRITTEN BY

**ARMUGHAN RAFAT,
MOHAMAD AFSHAR,
& MARKUS ZIRN**

faces. A single sign-on framework integrated into the booking portal provides agents' security and enables the logging of the agents' activities. Each time a customer quote is generated by the agent, NGTS creates a lead in the Oracle E-Business Suite *sales online* application. To generate a personalized vacation package for a customer, the agent works through each component of the vacation package, such as airline, hotel, and car. A J2EE-based packaging and promotion engine (based on a rules engine) within NGTS gathers the appropriate schedule and pricing information from partners and internal systems, applies appropriate discounts, and delivers the information to the agent through the portal. When the sale is made, the booking is published to the ESB infrastructure where it's propagated to the CRM, Financials,

and Project Accounting modules. The process is shown in Figure 2.

Airlines, hotels, and other partners may change customers' travel schedules. To handle this scenario, LibGo created a callback service within NGTS that the partner systems call via Web services. NGTS processes this information and publishes it to the ESB so it can be matched against a customer booking, thus triggering notification – either by e-mail or by agents calling customers.

Now, the reality-check: not all vendors use HTTP-based communication to send information to LibGo. Pricing and availability information can come in multiple formats from FTP, flat files, message queues, e-mails, and fax. Also, each of our internal and external systems has a slightly different data representation, even for the same entities. In keeping with LibGo's design principles, a service adapter layer encapsulates capabilities that are implemented in both internal and external partner systems, and abstracts the business logic for access through an XML-based interface defined in WSDL. The enterprise service bus (ESB) infrastructure facilitates communications and provides services such as data transformation and enrichment. NGTS also provides key foundational services on its own, such as common security services, utility services such as fax and e-mail, a shopping cart-like service, and an elaborate rules engine. All front-end systems (including the .NET-based Web presence) use the common services and business logic provided through NGTS and the ESB infrastructure.

The rules engine is a truly foundational service used to do all manner of things ranging from implementing the decision logic for alerting (call a customer immediately in case of a same-day flight change), to implementing the result caching policies (refreshes price information for air travel on some routes more often than others), to enabling dynamic pricing and packaging of travel services (mid-week travel and minimum night stays).

Now let's discuss LibGo's experiences with caching, transactions, and security.

Caching Is a Necessity for Real-Time Distributed SOAs

Four key factors drove us to consider implementing caching within the SOA:

1. Each time a customer searches for a flight or holiday package, tens of concurrent queries

are fired across LibGo's internal systems and across the Internet to partners, which produces result sets that can regularly be larger than 1MB.

2. There are transaction fees involved in performing searches across air and hotel partner systems, whether or not the itinerary is purchased.
3. To meet immediate business needs and future plans, performance must withstand at least 50 interactions per second or 1 million per day, and response times must not exceed 10 seconds.
4. LibGo's system must take orders when back-end systems or partner systems are momentarily unavailable and must process them later.

Clearly, such performance can't be achieved by reading data from a distributed data store, let alone one in which data is being pulled from a number of partner systems in XML with Web services. Hence, LibGo needed intelligent caching regimes to make our SOA architecture more resilient.

Web-only travel vendors must cache heavily to avoid partner fees that are levied each time a search is carried out. Often enough, such "over-caching" results in bad data being served, and customers end up being disappointed when offers turn out to be unavailable, or at least not at the promised price, when they press the *Buy* button. For LibGo, the bottom line was that we needed capabilities to define sophisticated and configurable caching rules to deal with price volatility. For example, prices on flights to Hawaii are less volatile than those on Las Vegas flights. Therefore, it might make sense to not cache flights to a particular Las Vegas flight. Cache invalidation rules based on destination location and heuristics helped LibGo optimize the caching strategy. Figure 3 shows the type of data that is cached.

Caching at the UI level. This applies to HTML pages and fragments. Using Oracle Application Server's WebCache, information items such as destination, airline, and hotel information can be easily cached. Since the WebCache capability is closely integrated with Oracle Portal, it can cache output from both .NET-based Web systems and the J2EE system that supports live agents. We used events to manage this content cache; for example, triggers on changes within the content management system invalidate the cache in WebCache and force updates. This

avoids serving up bad data.

Caching of reference data within the composite application (NGTS). This applies to reference data such as state, country, and destination names or different types of attributes that make up 50MB of data located in different data stores that is sourced from remote systems. Using the Java

Object Cache in Oracle Application Server's J2EE container, LibGo cached this data at the mid-tier and avoided the overhead of remote system calls.

Caching of transactional data within the composite application (NGTS). This applies to the most perishable information, such as pricing information that relates to airline tickets and hotels. Each time a search is performed for a travel itinerary, the composite application draws upon the data cached in the Java Object Cache. If pricing information isn't available for part of the itinerary that is directly out of the cache, NGTS pulls the information in from the partner system. The result set is then stored in the cache for subsequent queries for both the same customer (in case the customer decides to change the hotel but keep the air ticket), as well as for other customers.

For LibGo, caching is a business necessity to meet performance requirements and avoid transaction costs. If you're developing an SOA, we strongly

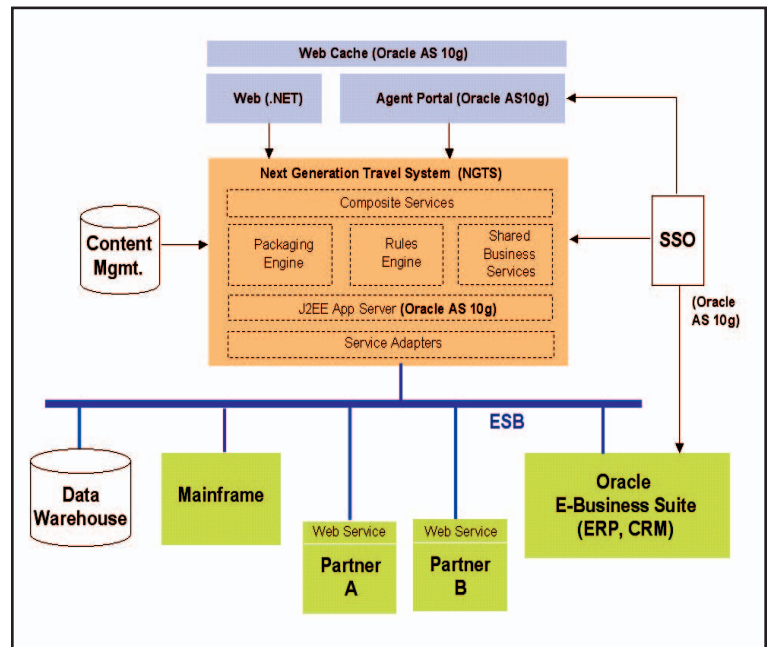


FIGURE 1 LibGo software architecture (simplified)

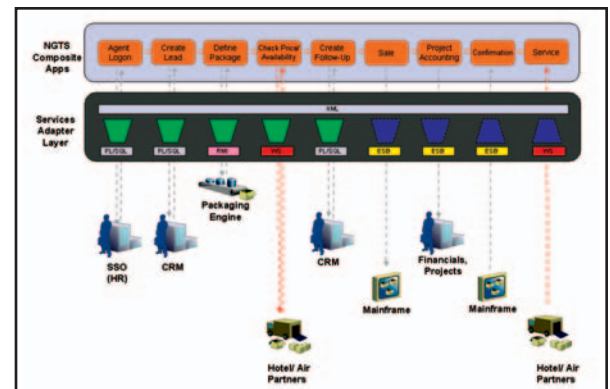


FIGURE 2 Customer order process flow with services adapter layer

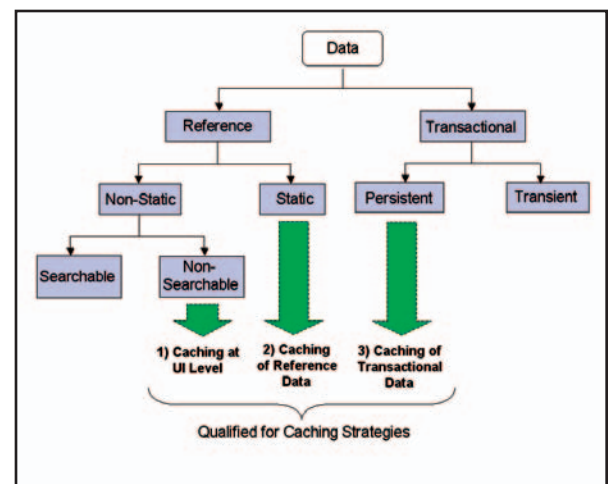


FIGURE 3 Hierarchical caching strategy

recommend that you consider multiple caching strategies to implement a better SOA. Always remember to cache your system codes. As long as you have management rules to avoid overcaching, caching will produce great returns.

Compensating Transactions Are Here to Stay

Remember the ACID properties of transactions (atomic, consistent, isolated, durable) and rollbacks in the database context? How about XA-based transaction managers that enable distributed transaction management across a number of platforms?

LibGo's business transactions are inherently distributed: travel bookings are usually packages, with each component provided by a different vendor. Early on we realized that most of our partners didn't offer sophisticated transactional interfaces to LibGo. Each time a customer request for a complete travel package (airline, hotel, car, and tickets to attractions) was processed, the NGTS system had to ensure that all constituent parts were available before committing the transaction. While standards bodies are addressing the general need to relax ACID properties and manage more complex business transactions within the context of Web services and SOAs, a pragmatic approach to manage our business transactions was needed. The idea to implement an all-purpose Web services transaction manager was quickly dismissed. Instead, LibGo decided to build a custom transaction manager and leverage existing transaction capabilities within their packaged applications, and at the same time, delay compensating transactions to the end of a serial transaction flow.

Avoid Compensating Transactions Where Possible

Our scheme was simple: easy-to-roll-back transactions would be executed up front, while the partner transactions would be delayed until the very end. Since the ERP system had sophisticated transaction capabilities, including rollbacks, LibGo decided that when building the composite application, the ERP-specific steps in the overall process are carried out first. If the ERP transaction didn't fail, the next gating steps would be executed – bookings across partner systems for airline and hotels. The model of sequentially ordered transactions minimizes the number of compensating transactions.

Work with Partners to Implement Compensating Transaction Logic Where Necessary

Some business transactions will need to be compensated. For this purpose, LibGo worked with partners to implement a scheme using unique transaction IDs that were passed in the XML request or SOAP header of each communication. The transaction ID would be set by the booking engine in NGTS each time a booking was to be made. Then, each partner system would append this unique transaction ID with its own information denoting the transaction path. This way, if part of a transaction failed, the compensation handler in the NGTS system has an audit trail that could be used to perform multistep compensating actions. This isn't necessary in the case of internal systems because these typically implement the rollbacks themselves, and so compensating transactions aren't necessary.

Here are LibGo's ground rules for distributed transactions in SOA:

- If only one component can potentially break any two-phase transaction agreement, move it to the end of the execution chain. For example, ERP services provided by E-Business Suite cannot be called from within another transaction because they implement commit and rollback logic internally. Another example is a Web service that does not implement rollback logic. By using this scheme, the chances are that compensating transaction logic will not be executed unless all previous steps that participate in the transaction have completed successfully.
- If more than one component is involved in breaking the two-phase transactions – for instance, both the hotel and airline bookings fail – the (custom) transaction manager must execute a compensating transaction. In the case of remote Web services, that amounts to sending a reverse message. If the compensating transaction also fails (after *n* attempts), then the transaction manager should generate a notification for manual recovery.
- For components that are not designed for two-phase transactions (such as PL/SQL-stored procedures, which are transactional themselves), wrappers written as Java objects must be written. The wrapper participates in container-managed transaction (CMT) and acts as the transaction resource. If a rollback is necessary, the wrapper is responsible for

the reverse/cancel action logic because the component doesn't support rollback. In case of failure, all of the Java (wrapper) objects participating in CMT need to implement the compensation logic. For example, the transaction that creates a customer, creates the lead in the Sales Online application, and places it on the queue, may fail at the final step. Since the customer creation and lead creation activities are transactions in themselves, it's necessary to write wrappers that implement the compensation logic for each of these.

- For those components that support transactions but can't participate in CMT (for example, an external Web service for airline bookings), the transaction should be pushed in the wrapper, which is responsible for implementing transaction logic, including rollbacks. In case of failure, the wrapper needs to issue the rollback command to the underlying component that then implements the rollback. Since the component supports transactions, this is a cleaner model than the point mentioned directly above; however, many systems do not support rollback logic, and instead need to implement compensating transaction logic.

The transaction ID also helps deal with communication failure; for example, if a transaction acknowledgement from a partner is lost in the network, NGTS, which performs automated retries, would resend the communication with the same transaction ID. The partner's system would look up the transaction ID from a local hash map and resend the acknowledgement to NGTS if it had already performed the action, or would carry out the task in the case that it didn't receive the original communication. Implementing standards such as WS-RX for reliable messaging in middleware stacks from the likes of Oracle will also go a ways in making life simpler for practitioners who need reliable messaging.

LibGo also decided on one additional small upgrade: to place timestamps in the SOAP header or XML payload, depending on the service, and to require partners to implement a clock synchronization mechanism along with LibGo. This has greatly helped troubleshoot the complex transactions and pinpoint bottlenecks as part of the distributed transaction environment. In a distributed environment with multiple Web services, it's always hard to pinpoint the performance bottlenecks. However with the

timestamp regime (you'll need a common time server), issues are resolved more quickly because it's possible to pinpoint the culprit right away.

Don't be intimidated by the new challenges of Web services transactions. Yes, you will have to worry about compensating transactions if partners don't participate in a transaction model, but often, a sequential approach to complex business transactions will minimize the chances of compensation. Don't try to solve world hunger; rather, find smart ways to leverage the context of your specific use case to delay compensating partner transactions to the very end. Make a unique transaction ID and a timestamp part of your transaction's SOAP header. Then, you'll be well prepared to execute compensation logic step-by-step to perform transaction playback.

Of course, standards will evolve for handling transaction semantics: WS-BusinessActivity, which formalizes undo semantics for long-running transactions, such as our booking transaction; WS-Coordination, which formalizes transaction IDs used across two or more transaction monitors; and WS-AtomicTransactions, which has stricter transaction semantics than WS-BusinessActivity and ensures strict, guaranteed atomic actions (such as writing a record to both a database and message queue). At LibGo, we continue to have a view to standards when implementing our SOA.

Pragmatism Pays with Security

LibGo has agents in stores, consumers online, and call center service representatives. The agents are responsible for bookings; managers must be able to obtain information on bookings and override policies. Given that the composite application incorporates business logic in NGTS, many ERP modules, and partner systems, the main challenge was to install a common access, authentication, and authorization framework across the applications that would enforce security and also enable auditing and logging (for compliance reasons).

To achieve this, LibGo used the HR model in Oracle e-Business Suite HRMS, along with Oracle Application Server Single Sign-On (SSO) and Oracle's Internet Directory (OID) LDAP store. User, resources, and entitlements from Oracle HR are populated into the OID store, which has application-specific objects. Every application and role has a set of entitlements; for example,

agents may be allowed to accept partial payment for over-the-phone bookings, but customers who use the Web interface cannot do the same. All applications are then registered with SSO to provide SSO and role-based authentication for all applications via JAAS (the Java package that lets applications authenticate and enforce access controls upon users). LibGo uses Oracle Application Server Portal and SSO to consolidate services and bind them into a user interface, and to provide a common security and personalization framework for enabling access to packaged applications, business intelligence and reporting applications, and composite applications in NGTS.

To secure communications between LibGo and external partners, we took a pragmatic approach of using secure frame relay lines with VPN as a backup solution. Such Web-based security approaches are a little heavy-handed because they often secure the entire wire protocol rather than just the SOAP message that is sent over the protocol. Further, for many message-based integration projects, several intermediary steps are necessary before messages arrive at their target endpoint, and transport-level security leaves the messages unsecured at each intermediary checkpoint.

To achieve a finer level of control and to avoid the intermediary security issues, LibGo is moving from today's existing transport-level security to message-level security. WS-Security defines a mechanism for adding three levels of message-level security to SOAP messages:

- **Authentication Tokens:** WS-Security authentication tokens let clients send, in a standardized fashion, username and password or X.509 certificates for authentication within the SOAP message headers
- **XML Encryption:** WS-Security's use of the W3C's XML Encryption standard lets the SOAP message body, or portions of it, be encrypted to ensure message confidentiality
- **XML Digital Signatures:** WS-Security's use of the W3C's XML Digital Signature standard lets SOAP messages be digitally signed to ensure message integrity. Typically, the signature is a computed value based on the content of the message itself: if the message is altered en route, the digital signature becomes invalid.

Though the flexibility and interoperability afforded by WS-Security is ideal, while implementations are developing, our transport-level security

is good enough to secure single conversations.

Conclusion

Building an enterprise-wide SOA is challenging. As more capabilities move into standards and into the middleware stacks of the vendors, however, the task should become easier. For example, when LibGo embarked on this project, Web services orchestration solutions were in their infancy. Now, it is possible to get high-performance, manageability, auditability, exception management, and a framework for building compensating transactions from BPEL Process Manager. In building out our SOA, we had a clear view of the evolution of standards and how capabilities around security and transaction management would work their way into products. When building your SOA, make sure you have this view – so you don't end up producing tomorrow's legacy systems.

About the Authors

Armughan Rafat is the lead architect of LibGo's Next-Generation Travel System (NGTS). Rafat, who has been building large distributed systems for more than 10 years, holds an MS in Software Engineering and Technology Management and is certified for the Microsoft, Sun, and Oracle platforms. Prior to working at LibGo, he led projects at AT&T and Lucent as a lead architect. He specializes in creating Enterprise Architectures for large-scale projects and writes a blog on Enterprise Architecture.

■ ■ ■ rafata@libgotravel.com

Mohamad Afshar, PhD is director of product management for Oracle Application Server 10g. He has core product management responsibilities for the Application Server that is part of Oracle Fusion Middleware. His main focus includes middleware vision, strategy, and architecture, with an emphasis on Web services, SOA, and EDA. Mohamad is a frequent speaker at industry events and is a contributing author to business-oriented, technical, and academic journals. He holds a PhD in Parallel Database Systems from Cambridge University, UK.

■ ■ ■ mohamad.afshar@oracle.com

Markus Zim is a director of Product Management for Oracle Fusion Middleware. In this role, he heads the Strategic Customer Program where he works with Oracle's leading and most innovative middleware customers. Markus has been part of the Enterprise Software industry for more than 10 years. He holds a Masters of Electrical Engineering from the University of Karlsruhe and is an alumnus of the Tripartite program, a joint European degree from the University of Karlsruhe, Germany, the University of Southampton, UK, and ESIEE, France.

■ ■ ■ markus.zim@oracle.com

Enabled SOA Transformation

Leveraging open source principles

■ As numerous organizations are planning to embark on their first endeavors in service-oriented architecture (SOA), it is important to recognize that the necessary organizational transformation has as much to do with cultural transformation, as it has to do with open, Internet standards-based design. In fact, the very nature of how business and IT view each other's role and how the enterprise views its relationships with its marketplace partners and customers is being altered. Such cultural change has never come easily and represents a significant organizational dilemma.

At the heart of an enterprise SOA transformation is also the need to "think differently" about how an enterprise acquires and manages IT systems. Albert Einstein might well have been talking about the challenges of SOA rollout when he said, "No problem can be solved by the same level of consciousness that created it." Building service-based systems involves a shift in thinking from large-scale, centrally planned IT systems to smaller, modular development that requires collaboration and consensus building among all of the stakeholders of an end-to-end IT process.

These aspects of SOA transformation alone



WRITTEN BY
**MICHAEL
KOCHANIK**

represent a large impediment to success for most organizations. Some simple questions that can highlight the importance are: How well do your business and IT stakeholders collaborate in the fielding of new business functions? How well do you execute globally distributed software development – not only within your enterprise, but also in the extended enterprise that includes partners, suppliers, and clients? Do you have a successful culture of component-based design and software reuse? Do you consider the elimination of silos within your enterprise to be a major goal? How well aligned are your operations infrastructure and capabilities with the role of IT service provider?

The aforementioned questions lead to a few aspects of open source principles and characteristics that may mitigate or address some of the significant challenges of SOA transformation. Table 1 lists SOA challenges matched with associated open source characteristics. The list is by no means comprehensive, but it is thorough enough to show the potential relationship in applying open source principles to the business problem of implementing SOA.

Given the success of open source software, the overall premise is that at a minimum there is something that can be learned from the open source development model, which enables organizations to become service-oriented enterprises (SOE).

Problem Illustration

The starting point for many SOA transformations is the desire of the enterprise to define an overall enterprise architecture (EA). EA is often employed as the tool of business modernization. In any EA initiative, the first step is usually an evaluation of the current state of affairs by means of assessment. The EA assessment usually reveals a map of redundancies across business processes and systems. These redundancies become the "low hanging fruit" for the first areas to be targeted for migration to enterprise services.

An oversimplified example might be an organization with multiple business systems, all of which require a software process to calculate loan risk or interest rate. Upon such a discovery, a natural inclination might be to fund a project to create a common shared service that could be leveraged by all such business systems. Although this approach is logical and has an obvious supporting ROI, the actual implementation will require more consideration and effort.

The first problem is the cultural bias of the existing business systems owners. Some will seek exemptions to not participate, usually based upon some notion that their requirements are somehow unique (i.e., somehow the financial aspects of the math are different for them). The real underlying reasons can run the gambit of NIH, concerns over support, or a reluctance to develop dependencies on other organizations or departments. These are some of the cultural dynamics that are often at work, which tend undermine efforts to create reusable SOA components.

A second problem is very tangible in the sense that building a shared service for this calculation may not be the right thing to do in all cases. Assume we deploy an enterprise service and institute a governance model where we instruct engineers to invoke this service directly rather than running their own copy of the software on their own systems. Well, that might work for some applications that only need to invoke that service a couple of times a minute, but what about other applications that need to invoke it hundreds of times per second, or are in latency-critical situations, such as rendering a Web page? In this case we might opt for building the software process as both an "open standard" by giving it a Web services interface and as "open code" by publishing the source and allowing it to be run locally.

This example hopefully provides an appreciation for the application of open source principles in an attempt to address a complex

issue. In addressing the cultural problem, we could adopt a community-based development approach in which business systems owners collaboratively participate in the development of a new solution. Such a shared development model could mitigate perceived risks of support and inflexible organizational dependencies. In the performance-based design problem, having easy access to source code and a supporting community of interest (COI) might enable the best solution to be applied. This would still be in general alignment with the goals of reducing development redundancy and having consistent enterprise architecture.

Some Focus Areas

According to the Yankee Group's "2004 Enterprise Web Service Study," 75 percent of the more than 400 US IT organizations surveyed are planning to invest in SOA approaches over the next 12 months. This seems reasonable because in any large-scale organization, the concept of enterprise shared services that employ component-based design is not new and is easily understood for its cost benefits. Often, "where to begin" is the harder problem and reflects where open source processes might be applied.

Driving Adoption and Utilization

In the end, what makes for a successful Web service is fundamentally a case based on whether or not anyone cares. Success is really a proof point based on actual adoption and utilization levels. Successful open source projects understand this point and invest some significant percentage of effort to support and grow the COI around the project. There is a strong correlation between how open source COIs grow and how to succeed in creating highly successful SOA components.

A case in point is *discovery*. The question is whether or not discovery via UDDI registry or other means is sufficient by itself to drive a service's adoption rate? This problem is analo-

gous to the push to develop reusable software assets based upon traditional component-based design. Many enterprises implemented so called "component repositories" where they encouraged the developers to register their components after development was complete to make them easy to discover.

The theory was that engineering leads and project managers would occasionally search this library of components and reuse them if possible. Given the amount of serious and sustained attempts by qualified organizations and smart people, the fact seems to remain that software reuse initiatives are rarely as successful as has been hoped. Many of these repositories ended up looking like a kitchen junk drawer.

The implication for SOA is that the same fate can befall SOA service registries. Whether it is reusing code or adopting a service, there are a lot of questions that potential users want to have answered before making a commitment. Obvious questions are: What's planned on the roadmap? Who is supporting the service? What are the known bugs and limitations? Who else is using it?

The final decision is really about mitigating the risk of adoption. The consumer might feel more comfortable if, through the interaction with a service's associated COI, they can get a sense for the service, its user base, and possibly even the consumer's own desire to become a contributing member of the project. The probability of adoption and the actual utilization of a service can be enhanced by enabling a discovery registry to provide easy access to the COI that is supporting the Web service.

Application Lifecycle Management

While SOA's impact on culture might be intuitive, what is not so obvious is its critical impact on application lifecycle management (ALM). Existing tools and software development processes may not be well aligned to support the move to SOA. At a macro level, an organization needs to understand that it is essentially migrating from insular LAN-centric development with a captive group of resources to a more open WAN-centric development environment with a diverse ecosystem of stakeholders.

In a recent interview on the importance of ALM, Gartner analyst Theresa Lanowitz

SOA Challenge	Open Source Characteristic
Aligning business and IT	Transparency, community-based development
Driving adoption and utilization	Ease of access, community-based development
Application lifecycle management (ALM)	Project persistence, agile processes, shared source code
Globally distributed development	Ease of access, Internet-enabled tools
Component-based design and reuse	Ease of access, shared source code, peer review

TABLE 1 SOA challenges and open source characteristics

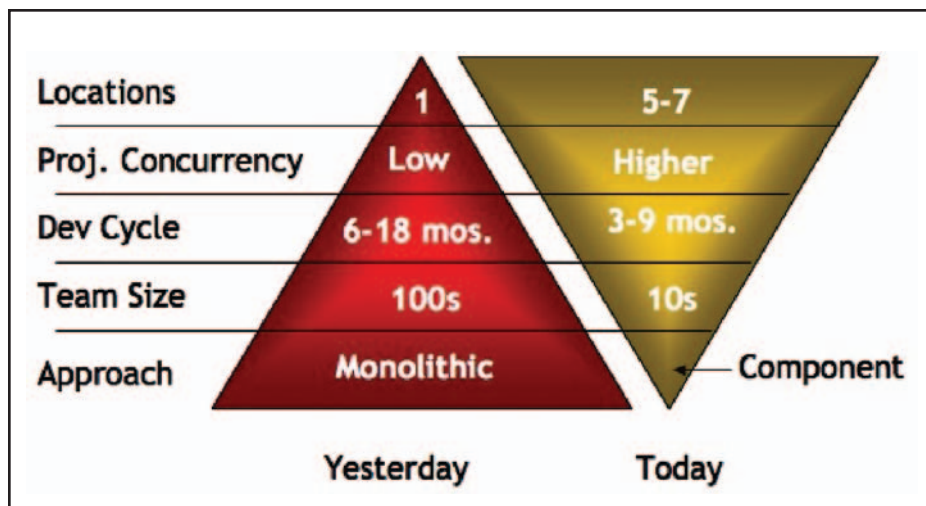


FIGURE 1 | What is changing in enterprise software development

commented that “As we start to bring forward this idea of a services-oriented architecture, and as we start to see services exist in a business-to-business environment, the services exist from two separate companies, and those services possibly are federated together in a loosely coupled environment. So we find the idea of the application ecosystem, the idea of a development ecosystem, is crucial as well, because there are going to be two different organizations creating a service that will have to be used together.”

Many organizations are not prepared to address the need to support across the firewall software development with smaller agile teams from either a technology or process perspective. Figure 1 shows the ALM perspective of the potential impact of moving from monolithic IT applications to granular Web services.

From an open source project perspective, the right side of Figure 1 represents “business as usual” for many open source projects (i.e., smaller and geographically distributed individuals and teams employing agile development processes, working on some modularize software function or component, and using the Internet as the backbone for a software development platform). Project governance is often consensus driven and the project home becomes a focal point for code, context, and community.

A case in point is globally distributed development (GDD). The ALM environment that supports a SOA transformation will have to

enable an end-to-end life-cycle process across a diverse group of stakeholders and locations. In many cases, offshore development partners and business partners will be critical members of the overall ecosystem.

“With SOA you can have business units create point applications, which publish services that are used by others in the same manner as using functionality from an SaaS (Software as a Service) provider such as Salesforce.com. Along with these opportunities, enhanced collaboration between distributed teams becomes even more important. A shared component-based development methodology is essential, and the use of common technical frameworks is highly desirable,” according to Dr. Gautam Shroff, vice president of the Technology Program and head of the Technology Innovation Lab at Tata Consultancy Services, Ltd.

While the image of open source programs from an ALM perspective might be that of an ad hoc collection of developers, the reality is that well-run open source projects have a comprehensive set of ALM processes enabled

by agile tools. Table 2 shows some of the characteristics of integrated ALM for SOA. Organizations can determine by inference the core competencies in software development that they need develop in order to address the challenges of SOA transformation.

Summary

As an organization looks to migrate from a large, monolithic application culture to an agile, Web service-based (SOA) culture, it will have to hone new core competencies in managing globally distributed development across the extended enterprise and across its partner ecosystem. The foundational pillars of these new core competencies are: governance, collaboration, ALM, discovery (UDDI registries, etc.), run time quality of service (QoS), and service level agreement (SLA) management. The good news is that most software development organizations should be able to glean some aspects of open source principles and be able to apply them to drive a logical benefit.

Here are some key capabilities and recommendations to consider for addressing the challenges of SOA transformation:

- **Get Modular:** Maybe the first thing an organization needs to do is to “get modular” about the code it builds. One approach is to use APIs that can be mirrored as a Web service. This allows the user community to decide the correct delivery approach, based on its own voluntary sense of where the control vs. performance balance should be set.
- **Get Good at Collaboration:** Collaboration is a critical skill in sustaining SOA. “The best companies are the best collaborators. In a flat world more and more business will be done through collaboration within and between companies,” as cited by Thomas Friedman in *The World is Flat*, with respect

ALM Functional Area	Requirement/Characteristic
Process	Process-independent, supports Agile methods
Life-cycle phases	Supports full spectrum of definition, design, code and build, test, deployment, and support
SCM	Lightweight and net-centric code management Example: Subversion (www.tigris.org), CVS (www.cvshome.org)
Stakeholder support	Full spectrum support for business and IT users, integrates business, engineering, and operations
Security	Supports role-based access and application-level security for across-the-firewall collaborative development
Deployment model	Net-centric deployment as a service, SaaS vendors

TABLE 2 | Integrated ALM for SOA

to how companies will cope in a new flat landscape. Collaboration is an essential skill both for IT and business.

- **Employ a Community Source Process:**

Having access to source code is a benefit to participants in enabling successful SOA initiatives. Whether this is open source, shared source among partners, or simply access to the code that is associated with APIs, the potential users of a Web service are more likely to participate and become effective contributors in the long term usage and life cycle of a Web service if they are enabled to participate at a source-code level.

- **Adopt a Common ALM Approach:** The ability to apply a common life cycle across the spectrum of definition, design, code/build, test, deployment, and support is an essential part of enabling large-scale SOA implementations. It brings about the required communication and visibility across all stakeholders, including project management, enterprise architecture, engineering, QA, operations, and business end users.

ALM also feeds the necessary compliance hygiene for SOX or other governance compliance programs.

- **Use Network-Centric Tools:** Design teams can no longer be assumed to be collocated. More often than not, today these resources are coming from flexible sourcing partners located in any number of offshore locations. The tools employed have to work efficiently across the Internet, while enabling the necessary security. Where possible, employing WAN-centric open source tools is a benefit as it lowers the cost of entry for participants and does not create a tool barrier to prevent collaboration.
- **Support COI:** Successful Web services will attract and build associated COIs. The ability to support and sustain these communities requires supporting service-oriented infrastructure (SOI), including communication services and collaborative software development services. SOA initiatives should consider the concept of proactive community development and management.

In general, the theme for success is that it is less about *how you want to do it* and much more about *how others want to do it*. SOA implementation turns ordinary IT organizations into suppliers of shared services that are subject to the same constraints as traditional software vendors in terms of delivering expected functionality, support, and QoS. In other words, being successful at SOA means there is some amount of “selling” to be done in order to drive adoption and success. Applying open source principles maybe a beneficial step along the way. ©

■ About the Author

Mike Kochanik is vice president of alliances at CollabNet (www.collab.net), the leading provider of on-demand distributed application life-cycle management solutions for software development. Mike has been a major contributor to the creation of network-centric IT strategies that leverage open source software and community-based development processes at Global 1000 organizations.

■ ■ ■ mike@collab.net

WSJ: FAQs

Orchestration and Choreography

Dear Sir or Madam:

I would like to ask the following: What are the differences and similarities between orchestration and choreography?

Thanks in advance!

Sincerely,
Nayden Nachev

Hi Nayden,

Technology terms are always confusing. In our context, orchestration and choreography both apply to BPM (business process management). Both relate to the concept behind the modeling of a business process and business logic in the process as a series of tasks and activities.

Orchestration assumes the presence of a central director that is orchestrating the tasks and activities in a business process. This is the equivalent of a conductor conducting an orchestra. Orchestration is best suited for long-running business logic that is provided by a combination of services.

Choreography assumes that there is no central point of control, but rather that the tasks and activities in a business process are choreographed together as a group. Therefore, in choreography, the interaction between the members of the group becomes a key concept. Therefore, choreography is concerned with describing the message interchanges between participants. The interaction is between participants who are peers. Choreography applies to the interaction between the business logic of long-running processes in an SOA.

(Names of readers are revealed only upon their consent.)

PRESENTED BY
**WSJ EDITORIAL
BOARD**

EDITED BY
AJIT SAGAR

Aren't SOAP, WSDL, and UDDI all necessary for Web services?

No. Although WSDL, SOAP, and UDDI are considered to be the main building blocks for Web services, all three are not necessary to define a service as a Web service. A service is defined as a Web service if it uses any of these technologies (not necessarily all of them). Therefore, the common building block of a Web service is primarily XML, which is

necessary for WSDL, SOAP, or UDDI.

A Web Services platform provides the foundation for the next generation of distributed computing, thereby enabling the creation of a web of collaborative services that are machine-accessible and peer-to-peer in nature. Services can be exposed via WSDL. Consumption of services is up to the consumer, and does not mandate SOAP or UDDI. SOAP can be used as a transport protocol without UDDI. SOAP and WSDL can be (and are in most cases) used without UDDI.

That said, the true vision of Web services is to use all of these technologies together. Moreover, Web services typically assume the usage of at least WSDL and SOAP. ©

■ 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

Web Services and Enterprise Content Management

Hype vs. reality

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

In the past each group within an organization typically worked with its own content and applications, each with its own server, capabilities, workflows, and user interface: financial personnel with accounting data and systems; knowledge workers with documents in document management (DM) systems; customer service staff with CRM platforms, and so forth. As long as business processes remained confined within these departments, the lack of interoperability among these systems was of secondary concern.

However lately, in order to improve productivity and agility in responding to changing business requirements, organizations are



WRITTEN BY
**CHARLES
HOUGH**

pursuing new initiatives and objectives that increasingly involve multiple departments, data stores, and business processes, as well as moving more content and applications out to internal and customer-facing Web sites. As developers seek to integrate ECM into a broad range of packaged and custom applications and portals, the need for a more flexible approach to integrating ECM technologies across the enterprise becomes increasingly evident. Separate silos for different types of content make composite business applications difficult to develop and deploy, and require custom code to bridge different APIs, interfaces, and platforms. Further, foundation capabilities such as security, access control,

workflow, and renditions may be available only in selected content type-specific offerings.

Web services offer a way to escape these content silos, thereby enabling developers to put specific ECM functionality where it's needed more quickly and in a more cost-effective manner, for example by:

- Displaying the content and metadata stored in an ECM repository within a business interface such as a portal
- Embedding a process for content creation, review, and deployment within a business interface
- Adding an in-house process management toolset to a digital asset management interface
- Aggregating search services for disparate repositories to provide user-friendly content-access capability

By using Web services in this way, companies can streamline internal processes and cut time and costs on the back end so that they can devote more resources to servicing customers.

So far, the actual deployment of Web services-based ECM has been tentative. Gartner reports that North American companies are implementing Web services slowly, as they are still in the "experimentation stages." Gartner also reports that, of those respondents being assisted by a consultant or systems integrator in

the implementation of Web services, 68 percent are implementing fewer than 10 services, largely for intracompany purposes.

However, this slow adoption rate shows signs of increasing in the near future. According to the same Gartner survey, over 50 percent of the surveyed companies reported that they planned to use Web services for IT initiatives such as CRM, ERP, or SCM. In tandem with this growing interest, the ECM Association AIIM (www.aiim.org) has recently unveiled a new project to develop a Web services framework for Interoperable Enterprise Content Management (iECM). The stated goal of this project is to help organizations integrate content into business processes through a common set of standards for accessing enterprise content.

As more developers and companies prepare to take the plunge, they will discover both the significant potential benefits of this approach as well as the work that remains to be done to fully realize these benefits.

Early Potential – and Remaining Challenges

The Web services stack consists of several specifications, including SOAP, WSDL, and UDDI – all of which are based on XML. The recent BPEL (Business Process Execution Language) specification allows process engines to orchestrate Web service invocations. By leveraging these open technologies, providers of Web services can significantly improve upon existing API functionality for system and business process integration. For example, consider the following benefits:

- Parceling out API functionality based on user skill level, helping adoption, and improving the usage of the base application

- Insulating users from minor API changes, reducing the overhead of having to rebuild and deploy client applications
- Providing a layer of business logic to help minimize confusion and/or misuse of the base API and application

Web services also allow different APIs to be aggregated to provide a single, common point of use through an unmodified, client-facing interface. For developers who are creating composite applications, the potential benefits of Web services include:

- Quick access to static information (snapshots)
- Ease of use and access without extensive coding and/or knowledge of the application
- A single point of access that simplifies deploying bug fixes
- Dynamic usage of new functionality without the headache of redeploying new client libraries
- Client applications that are no longer bound by the system requirements of client libraries

While the interest level has been high, there are challenges with Web services-based ECM that remain to be overcome. One of these challenges is limited availability. While some ECM vendors have invested to make their functionality available as Web services, many more have yet to do so, and more progress is needed in this respect. On a technological level, more work is needed to improve the performance and capabilities of Web services and the applications they enable. While Web services are already a highly effective way to exchange information that can be considered static, such as a snap-

shot of the current state of a system, the development of more robust clients will depend on greater interactivity and responsiveness.

While the XML basis of Web services offers many benefits such as cross-platform interoperability and language independence, these benefits do come with a downside, such as the extra burden of processing XML. Attempts to address this limitation include a binary XML standard and Sun Microsystem's Fast Web Services initiative.

Another important issue, which is particularly relevant to ECM systems, is the use of Web services to transfer large files that represent content. Here again there are several approaches, including the use of emerging standards such as WS-Attachment or WS-Transfer, out-of-band techniques with existing protocols such as FTP, custom-built techniques such as HTTP form posts or network file protocols, or even a separate socket.

Strategies have also been proposed to address the current limitations of Web services in issuing long-running requests. These include the use of asynchronous requests, the implementation of client-side threading, and the utilization of emerging standards such as WS-Eventing to move to a more event-driven model. The latter approach can also be effective in enabling the real-time display of dynamic data.

The Road Ahead

For the moment Web services are most suitable for high-level interactions that do not require either a large degree of bandwidth or a large number of transactions – querying an external system for data, for example. The efforts described above will help enable more robust deployments in the coming year or two, as will improvements to performance through the tuning of underlying systems to the traffic patterns that are characteristic of service-orientation. More important, both performance and stability will improve as organizations gain a deeper understanding of the best ways to employ Web services.

One way to think about the progression of Web services is to recall the early days of Web publishing, when advances in site design and technology outpaced the physical limitations of the network, and the most sophisticated sites were also the most likely to bog down over

“

As more developers and companies prepare to take the plunge, they will discover both the significant potential benefits of this approach as well as the work that remains to be done to fully realize these benefits

”

“ To be sure, the ongoing hype surrounding Web services has made some wary of exaggerated promises and overlooked pitfalls, but beneath the hype lies real promise, and a better way to support today's rapidly changing business requirements ”

dial-up connections. Over time though, design principles emerged that were more firmly rooted in practicality, and a focus on optimization led to sites that were both impressive and readily accessible. Similarly, as the development of individual Web services continues, developers will become more adept at techniques for optimizing performance and meeting more demanding requirements.

As this maturation process continues, a new breed of ECM solutions will become possible, offering such benefits as can be illustrated in the context of marketing campaign management. The process: the campaign manager first reviews past campaigns in the CRM system for audience and success criteria. Extracted data and content then become the starting point for the new campaign, supplemented by new text, graphics, and rich media content. This content then goes through a cycle of review, revision, and approval, followed by partner collaboration, testing, and then deployment. Finally, the new campaign is logged into the CRM system for tracking and analysis.

To support this process, a campaign management solution needs to be able to cut across content types – documents, rich media, collaboration, and Web content – that may reside in separate repositories. Moreover, the content must be closely integrated with an external business system, in this case CRM. Such a solution is an ideal candidate for a Web services-based composite application, which could draw on ECM functionality that is exposed as Web services, including core document manage-

ment services (check-in, check-out, categorization), search, content transformation and meta-data extraction, as well as CRM Web services to provide access to accounts, campaigns, contacts, events, and opportunities.

A composite application based on these services would enable the campaign manager to:

- Manage marketing collateral in one place by using integrated ECM functionality
- Schedule and execute targeted marketing campaigns by using integrated CRM functionality
- Use configurable workflows to manage different aspects of the campaign by leveraging the ECM-CRM synergy

Behind the scenes, a business process engine could automate and track the activities of all process participants, including content contributors, approvers, campaign partners, and Web application developers, and even synchronize the data on multiple back-end systems.

As more ECM providers make their functionality available as Web services, and as developers become more adept at maximizing the effectiveness of Web services-based integrations, the benefits of this approach will drive a fundamental shift in the way organizations approach ECM. To be sure, the ongoing hype surrounding Web services has made some wary of exaggerated promises and overlooked pitfalls, but beneath the hype lies real promise, and a better way to support today's rapidly changing business requirements. ©

■ About the Author

Charles Hough is vice president of Technical and Field Marketing at Interwoven. He leads the company's ECM platform marketing, field training, technical marketing, product management, and the Interwoven Developers Network (DevNet), where Web architects, Web managers, and IT decision makers can find answers and solutions to their content management, deployment, and integration challenges. DevNet is also a place where developers can freely access Interwoven Web services for the creation of platform-independent ECM solutions.

■ ■ ■ chough@interwoven.com

WSJ Advertiser Index

Advertiser	URL	Phone	Page
Blog-N-Play	www.blog-n-play.com	201-802-3000	43
Forum Systems	www.forumsys.com	801-313-4400	3
Gartner	gartner.com/us/adea	800-778-1997	11
IBM	www.ibm.com/middleware/flexible		5
Intermedia.net	www.intermedia.com	888-379-7729	6
ISSJ	www.ISSJournal.com	888-303-5282	25
IT Solutions Guide	www.sys-con.com	888-303-5252	29
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	38
SYS-CON FREE Newsletters	www.sys-con.com	201-802-3024	39
SYS-CON Website	www.sys-con.com	888-303-5282	17

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.

This Month

The Semantic Organization

Michael Wacey

Corporations have a tremendous amount of stored information. On top of this, new information is being created every day. A small but critical portion of this information is stored in highly structured and well-defined formats in relational databases. However, most of the information is on paper, in e-mail, in word processing documents, in spreadsheets, in PDF files, in engineering diagrams, and so on.

Ever since the initial XML draft in 1996, there has been an ongoing discussion of the semantic Web. A Google search for the exact expression the semantic Web returns about 1.2 million Web pages. Clearly there has been and continues to be a lot of discussion about the semantic Web. However, the semantic Web is still being worked on.

This is mainly because very little information on the Web has not been semantically tagged. It may be more prudent to start on a smaller scale than the Web.



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:

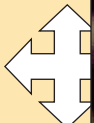
Storage, transformation, representation, and general use of structured and unstructured data

Enterprise Solutions:

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

XML Labs:

Product reviews, book reviews, tutorials, and standards analysis



HOME



ENTERPRISE SOLUTIONS



CONTENT MANAGEMENT



DATA MANAGEMENT



XML LABS

The Semantic Organization

Knowing what you know



WRITTEN BY
Michael Wacey

Corporations have a tremendous amount of stored information. On top of this, new information is being created every day. A small but critical portion of this information is stored in highly structured and well-defined formats in relational databases. However, most of the information is on paper, in e-mail, in word processing documents, in spreadsheets, in PDF files, in engineering diagrams, and so on.

Ever since the initial XML draft in 1996, there has been an ongoing discussion of the semantic Web. A Google search for the exact expression the semantic Web returns about 1.2 million Web pages. Clearly there has been and continues to be a lot of discussion about the semantic Web. However, the semantic Web is still being worked on. This is mainly because very little information on the Web has not been semantically tagged. It may be more prudent to start on a smaller scale than the Web.

There is some clear precedent for this. Web services and the underlying technologies (UDDI, WSDL, SOAP) all started out as having been intended for the Web at large. However, they have been most successfully implemented inside an organization. Similarly, it makes sense to create a semantic organization rather than taking on the whole Web.



Here are two examples of what a semantic organization might look like:

- A new RFP comes in to an advertising agency. It is from a consumer goods company that is looking to launch a new food product. The RFP manager could quickly locate similar RFPs that the agency has received, similar responses, similar project-related documents, and resumes of people who have worked on similar projects. All of these documents would be returned with the rel-

“The first step in implementing a semantic organization is to look for the low-hanging fruit”

evant passages highlighted. Documents for the launch of a food-grade plastic container would be successfully filtered out. In most organizations today, gathering all of this information can be half of the effort, and many relevant documents can be overlooked.

- A human services agency has been asked to provide a summary of the impact of a change in eligibility laws. Searching all case documents provides a summary of the clients that will be impacted by the proposed changes. The specific combination of attributes that are needed can be found, for example, people who have lived in the community for more than 10 years, who have worked for more than 15 years, and who have one of several diagnosis codes. Without semantically tagged case documents, it would require a manual search of the documents.

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



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) \$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

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. - Two Years (12) 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

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

*WHILE SUPPLIES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE

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


SYS-CON
MEDIA

“XML can play a role in three areas of building a semantic organization. These are data tagging, data location, and data relations”

There are seven key areas that are needed to support a semantic organization. The first is a common model of the organization. The remaining six are data tagging, data location, data relationships, data access, data storage, and data transformation. XML can play a role in three areas of building a semantic organization. These are data tagging, data location, and data relations. XML is not the correct tool for the common model, data access, data storage, and data transformation. (See Table 1.)

How do we get from here to there? The first step in implementing a semantic organization is to look for the low-hanging fruit. There are numerous areas in any organization where simple tagging of some information will provide a tremendous benefit. For example, a company could tag all of the proposals that they have sent out with some simple information such as company, project, type of project, etc. In these cases, the common model would be informal. Going beyond this will require a greater effort and support from vendors. The biggest challenge is the creation of a common model.

There are a number of tools on the market that support one or more of the areas required for a semantic organization. However, they tend to be special-purpose tools that require extensive setup. The process of tagging, storing, and retrieving documents should be built into the tools that we use every day. It should be a basic part of what everyone does.

Getting to this will provide organizations with tremendous insight into what they know. All of the information that is being generated will be widely available and useful. The data will be unlocked and available to benefit the organization. Organizations will be able to know what they know. 

AUTHOR BIO

Michael Wacey is a partner with CSC Consulting and has been involved in the data processing industry since 1982. He has worked as a CTO, CIO, and project leader in numerous areas, including the telecommunications, pharmaceutical, chemical, and financial industries.

 mwacey@csc.com

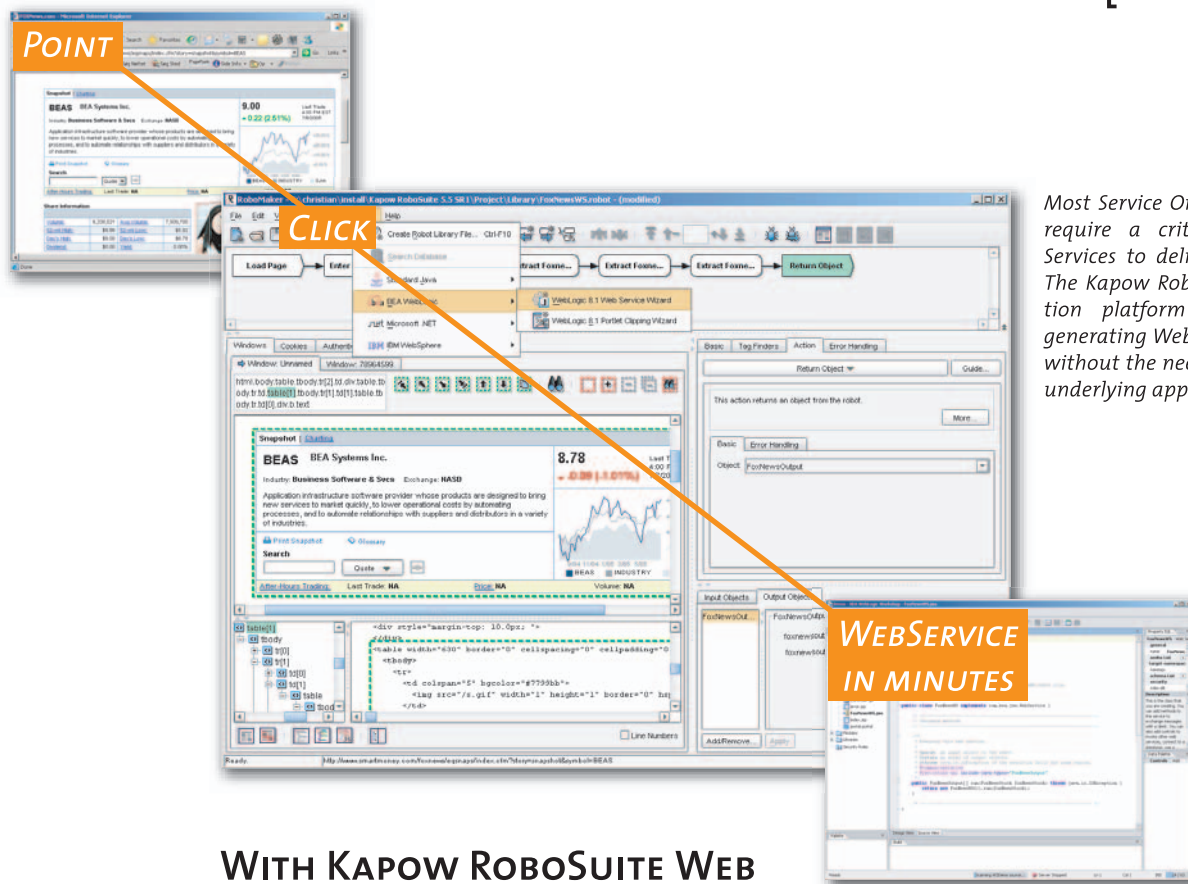
Area	Description
Common Model	A model of the entire organization to which all data can be mapped.
Data Tagging	Marking information with tags that identify how they relate to the common model.
Data Location	Documents that point to the location of other documents.
Data Relationships	Temporal and other relationships of data that are contained in documents. In particular, these are relationships that are not captured in the common model, but that are realized as the documents are created.
Data Access	The ability to retrieve specific documents and the specific data within those documents.
Data Storage	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.
Data Transformation	Facilities for transforming documents from one common format to another (requires a mapping to have been defined).

Table 1 • Key areas for a semantic organization

LEVERAGE EXISTING IT TO GENERATE WEB SERVICES

For
free online
demonstration,
Gartner "Cool Vendor"
release, and software trial:
www.kapowtech.com/wsjprint

[in minutes]



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.

Web Services

.NET J2EE XML JOURNAL

September 2005 Volume 5 Issue 9

The "B" in BPM Stands for Business

Differentiating pure-play
BPM from rebranded EAI
and Workflow technologies
pg.26

Enabled SOA Transformation

Leveraging open
source principles
pg.48

Web Services and Enterprise Content Management

Hype vs. reality
pg.52

The Principles Of SERVICE- ORIENTATION

New research reveals
the fundamental dynamics
behind SOA

PG.10

RETAILERS PLEASE DISPLAY
UNTIL NOVEMBER 30, 2005

\$6.99US \$7.99CAN

09>



71486103420 9