SYS-CON

NOW INCORPORATING XML JOURNAL

# WebServices JOURNAL

.NET J2EE XML

August 2005 Volume 5 Issue 8

**WSRP: Dynamic and Real-Time Integration**
An introduction to WSRP, its usage, and implementation
pg.10

**When Exceptions Are the Rule**
Achieving reliable and traceable service-oriented architectures
pg.32

**Differential QoS Support in Web Services Management**
One service implementation – many levels of service
pg.36

## Myths About Service-Oriented Architecture

PG. 40

Demystifying SOA

RETAILERS PLEASE DISPLAY UNTIL OCTOBER 30, 2005
$6.99US $7.99CAN
08
71486 03420 9

THE WORLD'S LEADING MAGAZINE DEDICATED TO WEB SERVICES TECHNOLOGIES • WSJ2.COM

*Ensure Interoperability.*

*Validate functionality.*

*Eliminate security vulnerabilities.*

*Test performance & scalability.*

*Confirm compliance.*

*Collaborate thru reuse.*

## Ensure Secure, Reliable Compliant Web Services

PARASOFT®
**SOA Test™**

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

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

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

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

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

PARASOFT®
*We make software work.™*

Automated Software Error Prevention™

## 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

WebServices JOURNAL .NET J2EE XML

August 2005 Volume 5 Issue 8

# Inside WSJ

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**

# 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**

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

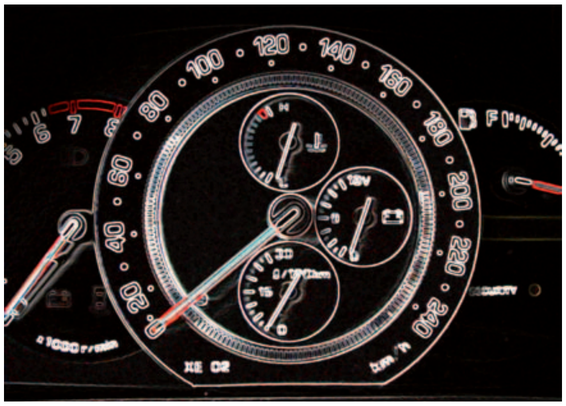
# Hot Wheels

WRITTEN BY
SEAN RHODY

I have a friend who's very into automobiles. He gets a new car every year or two – not expensive ones, but ones that can be used in stock racing (I know I'm getting the term wrong somehow) and time trials. He likes to drive and tries to get the most out of his vehicles. Recently, I went for a ride in his latest purchase. I'm used to his having computers and more gauges than the space shuttle, but somehow this one was a bit different, almost as if it were intended to get the most information to the driver. It even had a way to measure the tire pressure, from inside the car, while moving.

My friend believes firmly in managing his driving experience. One thing that struck me about it – it sure helps when the ability to manage is designed in from the beginning. I thought about what it would take to put that level of information in my Expedition. It would not be feasible.

Web Services Management is very similar to my friend with the car – very useful, and better built in up front. And yet management is often the last thing on someone's agenda – it's next year's to do, if the budget doesn't drop out.



That's a problem for a number of reasons. We've finally gotten to the point where we can talk about services and SOA and start to peel back the covers of the silos and let the true value of the application – its services – show through. SOA enables productivity in a number of ways, including reuse of services, and freedom from having to write the same security, management, data access, and other common code routines that can now be abstracted to a declarative paradigm.

It's understandable that management is not the first thing on the list when starting with Web services – it's a fairly significant paradigm shift to begin with, and everyone is caught up with how to determine the granularity of services, how to protect them, and even how to put a user interface together now that we no longer build applications. These are all challenges that need to be overcome.

However somewhere along the line we need to understand that management of services – where they are, what they are doing, who can access them, how they are performing – is as vital to an enterprise as the ability to invoke a service. Architects and developers sometimes ignore operations, but without it, the corporation grinds to a halt. Operating an SOA requires management. It becomes even more vital when services are shared across organizations that previously built their own. Quality of Service becomes vital, as all it takes is a couple of seconds of delay from an overloaded service to convince a division that the new platform is "buggy" and their old silo applications are really what they need.

I suppose building an SOA is a lot like the evolution of the automobile. At first cars were really "horseless carriages." They were simple devices, comparatively speaking. Then they evolved things, like windshields, automatic transmissions, brakes, shock absorbers, and license plates, and became cars. Then we added air bags, GPS, DVD players, and surround sound stereos. And we needed to be able to monitor all of the increasingly complex functions of a car, so we put in computers, and built a command center so we could monitor the tire air pressure.

Implementation of an SOA is similar to that progression. You start with basic Web services then add transactions and security. But you still need management; otherwise you're racing a model T at the Indy 500. Now where did I put my crash helmet?

## 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

WebServices JOURNAL
.NET J2EE XML

**PRESIDENT AND CEO**
Fuat Kircaali fuat@sys-con.com
**VP, BUSINESS DEVELOPMENT**
Grisha Davida grisha@sys-con.com
**GROUP PUBLISHER**
Jeremy Geelan jeremy@sys-con.com

**ADVERTISING**
**SENIOR VP, SALES & MARKETING**
Carmen Gonzalez carmen@sys-con.com
**VP, SALES & MARKETING**
Miles Silverman miles@sys-con.com
**ADVERTISING DIRECTOR**
Robyn Forma robyn@sys-con.com
**NATIONAL SALES & MARKETING MANAGER**
Dennis Leavey dennis@sys-con.com
**ADVERTISING MANAGER**
Megan Mussa megan@sys-con.com
**ASSOCIATE SALES MANAGERS**
Dorothy Gil dorothy@sys-con.com
Kim Hughes Kim@sys-con.com

**SYS-CON EVENTS**
**PRESIDENT, SYS-CON EVENTS**
Grisha Davida grisha@sys-con.com
**NATIONAL SALES MANAGER**
Jim Hanchrow jimh@sys-con.com

**CUSTOMER RELATIONS/JDJ STORE**
**CIRCULATION SERVICE COORDINATORS**
Edna Earle Russell edna@sys-con.com
Linda Lipton linda@sys-con.com

**sys-con.com**
**CONSULTANT, INFORMATION SYSTEMS**
Robert Diamond robert@sys-con.com
**WEB DESIGNERS**
Stephen Kilmurray stephen@sys-con.com
Vincent Santaiti vincent@sys-con.com

**ACCOUNTING**
**FINANCIAL ANALYST**
Joan LaRose joan@sys-con.com
**ACCOUNTS PAYABLE**
Betty White betty@sys-con.com
**ACCOUNTS RECEIVABLE**
Gail Naples gailn@sys-con.com

**SUBSCRIPTIONS**
SUBSCRIBE@SYS-CON.COM
1-201-802-3012
1-888-303-5282
For subscriptions and requests for bulk orders, please send your letters to Subscription Department
Cover Price: $6.99/issue
Domestic: $69.99/yr (12 issues)
Canada/Mexico: $89.99/yr
All other countries: $99.99/yr
(U.S. Banks or Money Orders)

Worldwide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ
Newsstand Distribution Consultant:
Brian J. Gregory / Gregory Associates / W.R.D.S.
732 607-9941 - BJGAssociates@cs.com

For list rental information:
Kevin Collopy: 845 731-2684, kevin.collopy@edithroman.com;
Frank Cipolla: 845 731-3832, frank.cipolla@epostdirect.com

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

SYS-CON MEDIA

# One Little Service Jumping on the Net...

WRITTEN BY
AJIT SAGAR

As organizations bravely venture into the world of Web services, they grapple with the age-old question – where do we begin? The main challenge that I have seen with key stakeholders looking to move towards the agile enterprise is solving the dilemma of which approach to take, top down or bottom up.

Some key business drivers for considering Web services, which in turn are perpetuated by a desire to add agility to the organization, are reduction of complexity, faster response to change, and reduction in overall cost. This in turn drives the need for reuse. As we all know, the path to reuse eventually leads to services, although unlike Rome, all roads that attempt to get there don't necessarily make it. Since the need for services is driven by business, the understanding of what the service is also comes from the business perspective. The Web service is defined in business terms – it could be a status request on a purchase order, or a request for card activation. The main point here is that the Web services paradigm is meant to be understood by business owners. Service-oriented architecture is meant to bridge the gap between business and IT. The realization and implementation of the service makes its way down the business architecture tiers to the technical architecture tiers to the actual creation of the code that implements the service. The business services are consumed by business components.

However, the gap between IT and business also causes organizations to look at the bottom-up approach. In several cases, the organization may not have all of the pieces in place to translate the requirement of a business service to technical terms. And the problem of integration and interoperability at the IT level is more "real," in these environments. In such cases, the approach is to take a look at what is in place in terms of technology, products, and the implemented architecture, and how it can be simplified, abstracted, and made more usable. The definition of a Web service starts from the back-end technical tiers, driving integration and interoperability between disparate environments and products. This again puts forth the requirement for reusable services that are to be consumed by technical components. The resulting abstraction makes its way up the tiers to be used by the business functions in order to expose the functionality of the application.

So we have two orthogonal paths to reuse, and they should (must) meet in the middle. The problem is that the approach on how to make them meet is not always that intuitive. Successful definition and implementation of the appropriate architectures (business and technical) requires a well thought through strategy for adopting the change throughout the organization. Typically enterprises choose one of the two approaches (top down and bottom up), cross their fingers, and hope that the realization will make it all the way to the other side. Fortunately, there are developments in the areas of WSM (another term in the world of overloaded three-letter acronyms that refers to the management side of Web services), and business process orchestration, which can aid organizations in achieving their objectives. Nevertheless, the challenges are still there.

In the meantime, one of the main risks of adopting Web services faster than they can be consumed by the organization is the creation of several poorly defined, redundant, too fine-grained, or isolated services that will be produced, but will prove hard to consume.

So, as we define a plethora of services as offerings from our individual applications, we need to make sure that the services jumping into the Internet are defined, controlled, and managed properly. After all, I'm sure you all know what happened to each monkey who jumped on the bed. And in this decentralized world of Web services, there may be no doctor to call.

## About the Author

Ajit Sagar is a principal archiect 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**

Mindreef®
www.Mindreef.com

# WSRP: Dynamic and Real-Time Integration

## An introduction to WSRP, its usage, and implementation

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

WRITTEN BY
**RAHUL KUMAR GUPTA**

WSRP stands for Web Services for Remote Portlets. It is an OASIS standard and a dynamic plug-in for portal pages. This allows portal or application owners to easily integrate remotely available third-party portlets or their own portlets using Web services. WSRP provides the dynamic integration in portals and integrated portlets are dynamically updated from the provider's own servers.

Let's understand a few related terms before looking at the inside story of WSRP.

***Web Services*** – Web services provide an application integration technology that can be successfully used over the Internet, which allows business applications to communicate and cooperate over the Internet. Web services allow objects to be distributed across Web sites where clients communicate and cooperate over the Internet. Registry standards enhance this by defining how the Web services may be published, found, and bound with minimal human interaction. Web Services standards are as follows.

***SOAP*** **–** A standard for messaging over HTTP and other Internet protocols. This is a universal service access protocol. Technical features include:

- Structured envelope for XML messages
- Headers…message path
- Protocol independent, i.e., can work with a variety of lower protocols like HTTP, HTPS, SMTP, and RPC
- Data model and encodings

***WSDL*** **–** A language for describing the programmatic interfaces of Web services. This is an XML document that describes the properties (interfaces) of the service. The language does the following:

- Provides the common interface definition language
- Follows XML vocabulary and structure
- Provides operational information about service, interface, implementation details, access protocol, contact endpoints, and security

***XML*** **–** Solves a key technology requirement that appears in many places. By offering a standard, flexible, and inherently extensible data format, XML significantly reduces the burden of deploying the many technologies needed for Web services. WSDL is one of them.

***UDDI/ebXML*** **–** A look-up business registry or database standard for indexing and publishing Web services so that clients (applications and development tools) can locate their descriptions. Typically it uses SOAP messaging (usually XML/HTTP) for publishing, editing, browsing, and searching for information in a registry. Some of the features are given below:

- Share information
- Discover other participants' services
- Define how you can interact across the Internet

Share Your SOAP

Now

LEARN

DEBUG

TEST

TUNE

SHARE

SOAPscope® 4.0

Try it free at www.mindreef.com/tryout

Web Services Diagnostics

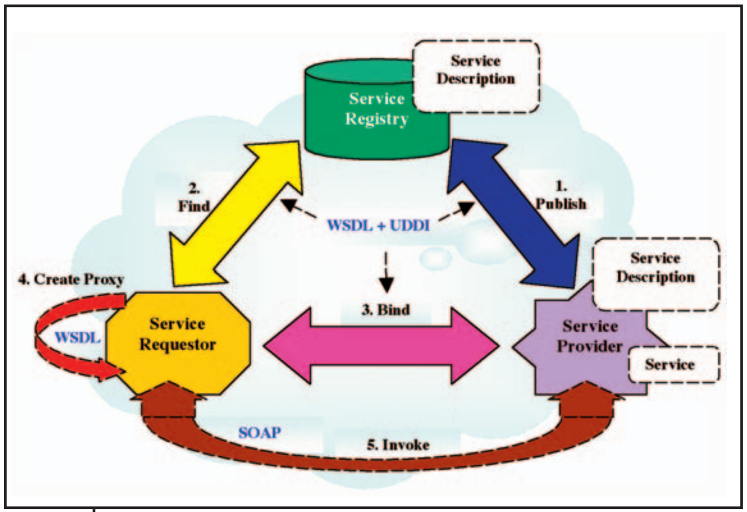


FIGURE 1 **Web service – process of engaging**

ebXML is a suite of XML specification-related processes and behavior designed to provide an e-infrastructure for B2B collaboration and integration. SOAP provides a low-level foundation that we can use to build these extended business exchanges, such as where there is a need for an agreed-upon structure for business transactions, multirequest transactions, schemas, and document flow – wherever SOAP provides some limitations. For more info on ebXML visit www.ebxml.org.

Basically, a Web service's sequence of flow for engaging (see Figure 1) is:

1. A Service provider implements a Web service and describes its interfaces using WSDL. Further, the Web service is published with central Service Registry.
2. A service requestor looks up for the Web services from a centralized Service registry using WDSL.
3. A service requestor binds to a specific service provider for a Web service using a WSDL file.
4. A service requestor creates a service proxy (WSDL).
5. A service requestor now communicates with the server (service provider) using SOAP (i.e., invokes the service via SOAP and receives a response via SOAP).

For more information visit http://developers.sun.com/techtopics/webservices/essentials.html.

### *Portal*

A portal is a Web-based application that commonly provides personalization, single sign on, and content aggregation from different sources for providing useful information to users. Portals uses both "push" and "pull" technologies to transmit information to users through a standardized Web-based interface, and they can be classified as vertical portals, horizontal portals, or corporate or enterprise (intranet) portals. Typically, portals get information from local or remote data sources, for example, from databases, directory services, transaction systems, syndicated content providers, and remote Web sites, etc. A portal is an application that aggregates portlet applications together in a presentable format. They render and aggregate this information into composite pages to provide information to users in a compact and easily consumed form.

### *Portlet*

A portlet is a Java technology-based Web component that is managed by a portlet container that processes requests and generates dynamic content called fragments. The portlet is essentially a small reusable application and it runs under the portlet container. The portlet container provides the runtime environment required for the execution of portlets; it manages the life cycle of portlets and provides a persistent storage mechanism for portlet preferences. Portlet technology is similar to Servlet/JSP, which is managed by servlet container. A fragment is a piece of markup (e.g., HTML, XHTML, WML) that adheres to certain rules. Fragments can be aggregated with other fragments to form a complete document. The portlet is accessible to users via a portal interface and portal uses them as pluggable user interface components that provide a presentation layer to information systems.

A portlet looks like a small window application within a portal page (see Figure 2). Portlets support some modes such as:

- View Mode – This mode provides the portlets with their actual functionality
- Edit Mode – In this mode we can edit the portlet's instance data
- Help Mode – Explains how the portlet should be used
- Design Mode – Used for changing the appearance of the portlet
- Preview Mode – Used for previewing the portlet's appearance

Portals also support window states like *Minimized*, *Normal*, and *Maximized* to display portlets in different sizes (www.jcp.org/aboutJava/communityprocess/final/jsr168/).

Current portal implementation follows the component model architecture that allows the plugging of components in infrastructure, which are referred to as Portlets (see Figure 3). Let's examine a case in which a company plans to build an intranet portal to provide the facility for viewing the latest stock trends, news, and weather information to its employees.

In order to implement it they have to provide presentation layers in News Portlet, Stock Portlet, and Weather Portlet, which run locally on the portal server and access remote Web

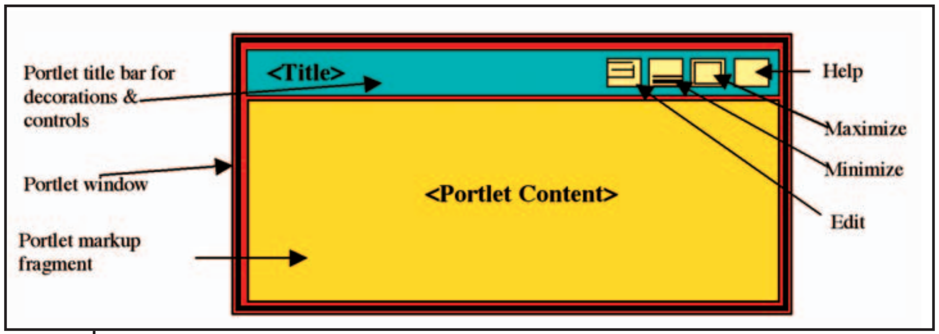


FIGURE 2 **Portal window**

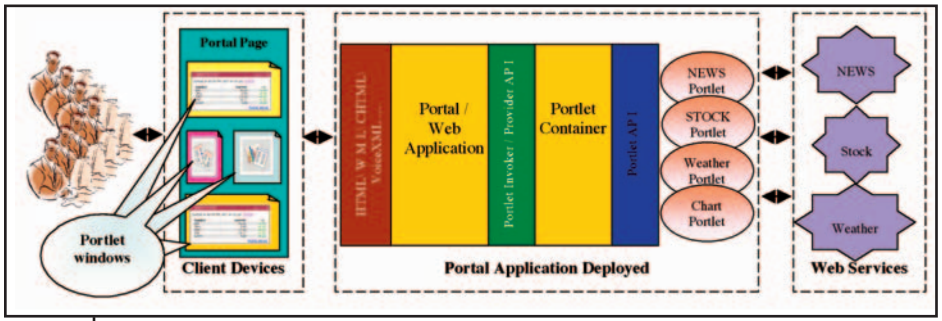


FIGURE 3 **Portal application deployment architecture**

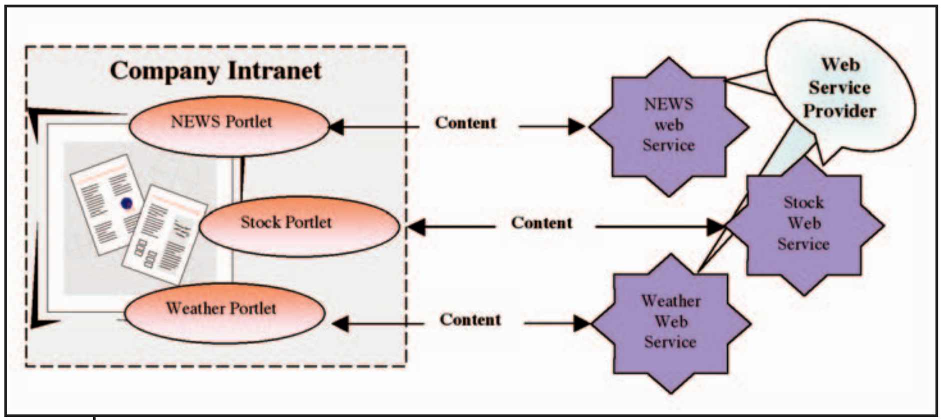


FIGURE 4 **Case study – portal application**

services to obtain the required information (see Figure 4). Later in this article we will see the benefits and drawback of this approach and how can we use WSRP in this type of scenario.

Now we have some understanding of various terminologies related to WSRP. To understand WSRP we have to find the answers to the some of the following questions.

- What is WSRP?
- Why and where would we use WSRP?
- What is the significance of WSRP?
- How can we provide the support for WSRP?
- How can we create and use WSRP?
- What are the limitations of WSRP?

## What Is WSRP?

Web Services for Remote Portlets (WSRP) was created by OASIS and it's a Web services protocol for aggregating content and interactive Web applications from remote sources and it operates over connectionless technologies. WSRP builds on a few fundamental standards, most notably XML, SOAP, and WSDL, while allowing for the implementation of evolving standards to deliver a protocol rich in the abstractions and operations that Web service implementers and portlet consumers require.

The WSRP standard defines presentation-oriented, interactive Web services with a common, well-defined interface and protocol for processing user interactions and for providing presentation conventions for publishing, finding, and binding such services. Because of these common, well-defined WSRP interfaces we can plug 'n' play any WSRP services with any WSRP-compliant portals/applications.

In a nutshell, WSRP allows enterprises to consume and publish portlets as Web services. It decouples our portlet applications from portals and instead of bundling all of our portlets with the portal in a single application, we can choose to deploy our portlets in individual portlet applications, and let the portal consume those portlets using WSRP. This results in easier team development, upgrades, administration, low development cost, and usage of shared resources.

In a typical WSRP implementation three parties are involved (see Figure 5).

1. ***End User*** – The user who is actually accessing the Portal and taking its services.
2. ***WSRP Consumer*** – This is a Web service client that invokes the WSRP Web services offered by the WSRP producer and also provides an environment for users to interact with the portlets offered by one or more such producers. A typical WSRP consumer is a portal, which mediates the markup and the interaction with this markup between end users and presentation-oriented Web services.
3. ***WSRP Producer*** – It offers one or more portlets for consumption and implements various WSRP interfaces/operations. Typically it provides a runtime (or a container) for deploying and managing several portlets. Producer implements set of WSRP-defined Web services that allow for producer and portlet description, portlet and consumer management, and portlet markup.

A WSRP consumer can also be a WSRP producer, and a WSRP producer can also be a WSRP consumer.

### *Why and where would you use WSRP?*

Let's reconsider the company intranet

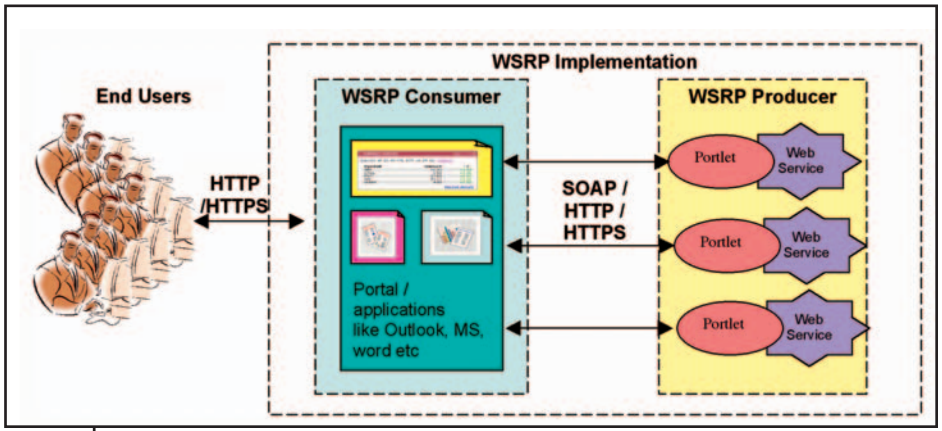


FIGURE 5 **WSRP architecture**

portal case study that we had discussed in the portal section. That approach works well and has no problems per se apart from certain limitations such as:

- This works only if all portlets are physically installed at the company intranet portal
- The company intranet portal team is responsible for designing and developing the presentation layer for each portal
- Most important, the company intranet development team is responsible for developing those local portlets as per the guidelines and in accordance with the interface description of the respective Web service provider

These shortcomings involve significant cost and loss of time – not to mention the maintenance overhead. Now the question is what is the solution for these kinds of scenarios. What if the company portal development team somehow can include application *and presentation* logic instead of just content? Wouldn't that be great? So, instead of just getting raw data and/or invoking business functions that still require special rendering on the portal side, it would be great if we could use the presentation services. See Figure 6 to understand the application architecture that uses WSRP.

WSRP has answers for all of our queries. As we know the WSRP are presentation-oriented, interactive Web services, thus they allow us to include the entire application, which adheres to the WSRP specification as it is in our customized application.

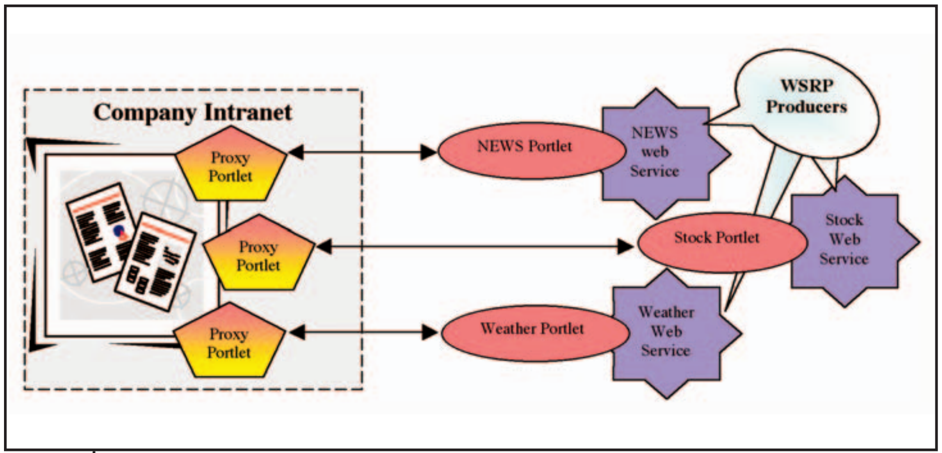


FIGURE 6 **Case study – WSRP application**

In this approach generic portlet proxies that consume all WSRP services conforming to the common interface eliminate the need to develop service-specific portlets to run on the portal. In this kind of architecture we have some major benefits such as:

- It is flexible and open ended – in the future if we find some other service provider is providing better services, the process and complexity involved in migrating to a new vendor is easy and low
- The complexity is low, which results in less time required and low cost
- There is low cost and less time required for development because we don't have to build the presentation layers or write any service-specific adapters for the concerned portlets/services
- It permits the usage of existing available services from different vendors
- Its implementation requires little or no programming

There are two major usage scenarios in which WSRP can be used:

1. ***For publishing content.*** Content providers can use WSRP to surface their content as remote portlets and publish them as WSRP services in public registries. In doing so they expose their services as per the WSRP specification.
2. ***For publishing existing services or local Portlets.*** In today's world we need to emphasize the notion of "REUSE AND DON'T REINVENT," and WSRP is a good technology for integrating the existing installations. Portals (intranet/extranet) can integrate remote portlets like any local portlet and put them on their pages, and applications may embed WSRP services through a plug-in mechanism, like COM components or ActiveX controls.

Some of the business scenarios where WSRP can be effectively used are:

- Intranet portals
- Content publishers
- Value-added service providers
- Information service providers
- Billing industry
- Enterprise Integration (EAI)
- Multimedia sports portal/mobility
- Travel and CRS

### *What is the significance of WSRP and what are its goals?*

With earlier portal vendors or organizations running portals, application integrators had to write special adapters to enable communication with applications and content providers using a variety of different interfaces and protocols. However, with this standard the entire approach is changed. Some of the WSRP features are:

- **Remote Invocation** – Can be used to include the remote applications as it is, which follows WSRP specification i.e., it delivers both data and that data's presentation logic.
- **Business Enablement** – Designed to enable businesses to provide content or applications in a form that does not require any manual content or application-specific adaptation by consuming applications.
- **Plug 'n' Play** – Easy to aggregate and can be invoked through a common interface using generic portlet-independent code that is built into the portal.

Some of the WSRP goals are:

- To provide interoperability, portability, and options for deployment
- To decouple the deployment and delivery of applications
- To remove duplication of functionality from otherwise discrete applications
- To enable interactive, presentation-oriented Web services to be *easily plugged* into standards-compliant portals

- To ensure concepts and data exchanged are *aligned with other standards* in both the portal and Web service arenas
- To make the Internet a *marketplace* of visual Web services, ready to be integrated into portals

### *How can we provide the support for WSRP?*

Now we know what WSRP is and what its benefits and goals are. Now let's understand the inside story of WSRP from a WSRP implementer (producer and consumer) perspective in the context of the end user. In this section we will discuss the following:

- Layered architecture (which includes the end user also)
- Remote portlets publishing and access mechanism
- List of interfaces (API) (and their brief descriptions) implemented by producers and consumers
- Sequence of flows of interaction between the end user, consumer, and producer

A portal can be accessed through HTTP protocol directly or indirectly i.e., a client using the handheld devices uses portal via WAP gateways and a client using telephone uses it via Voice Gateways. Based on the user agent (client devices), accordingly the appropriate markup language is used.

To better understand how to design the application that supports disparate clients I would suggest you read the "Building Internationalized J2EE Web Applications for Disparate Clients" article (www.devx.com/DevX/Article/9778/0/page/1).

The portal page that the end user sees is an aggregation of different set of portlets. Table 1 shows how WSRP portlets can be categorized into two types and shows the difference between them as well.

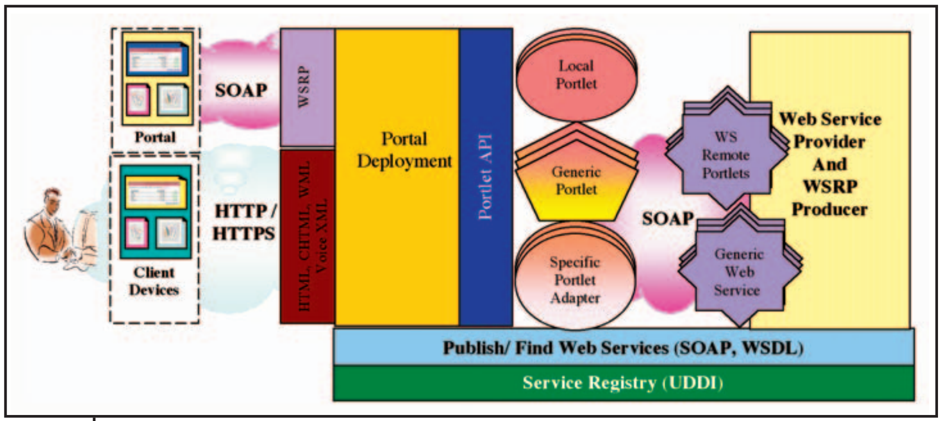


FIGURE 7 **WSRP-based application deployment architecture**

Local portlets can be accessed as remote portlets by some other portals, if our portal sever supports the WSRP producer specifications and WSRP services can be integrated into portals by wrapping them in adapter code written to the local portlet API. JSR 168 defines the standard API for this in Java programming language.

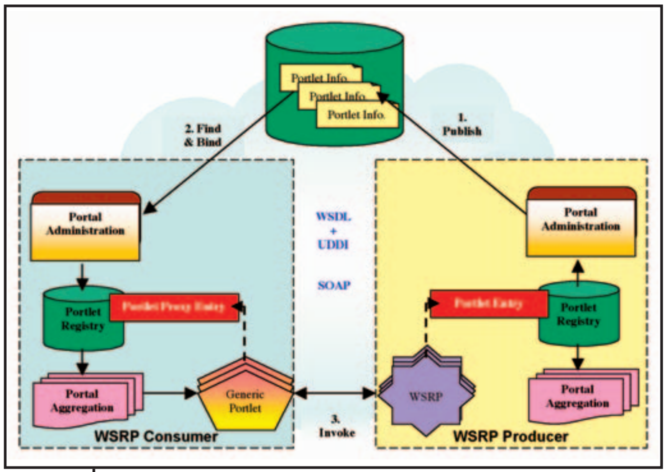


FIGURE 8 **WSRP – process of engaging**

As we know Registry (UDDI), SOAP protocol, and markup languages are the backbone for any Web service architecture. Here it's also the same. The process of engaging a WSRP is shown in Figure 8.

***Publishing Web Services for Remote Portals*** – The WSRP producer must publish WSRP to a Registry (UDDI directory). All WSRP must follow the WSDL interface and behavioral contracts standards.

| | Local Portlet | Remote Portlet |
|---|---|---|
| Deployment | They are deployed locally | They are deployed as web services at remote servers |
| Access Mechanism and Protocols Used | Local Portlet API is Used | Generic portlet proxies will invoke WSRP services to which they are bound via the SOAP protocol. UDDI directory is used to publish to easy finding & binding |
| Functionality | They are meant to provide the base functionality for portals in use. | To provide large number of additional and generic functions for a domain / business scenario. |
| Execution | Execute locally | Execute in remote server |

Table 1 **Two WSRP portlet types**

***Finding and Binding WSRP Services*** – A WSRP consumer can find the WSRP by querying the directory for registry (UDDI directories). During binding the WSRP consumer will typically store WSRP information for administrative purposes and in order to keep information on how to invoke the service. During this time the WSRP consumer creates/uses the generic proxy portlet.

***Invoking WSRP Services*** – The WSRP consumer passes the information about the user's request and portlet handle (a unique identifier for portlet) to the WSRP service and returns to end user the response it receives from the WSRP producer.

Now let's understand implementation process of WSRP. Since we are a technical people we should also understand how the features have been implemented as well as the major sequence flow(s) for a clear view and better understanding.

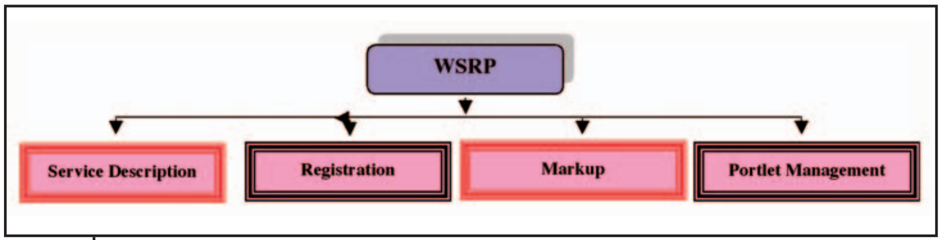


FIGURE 9 **WSRP API classification (interface)**

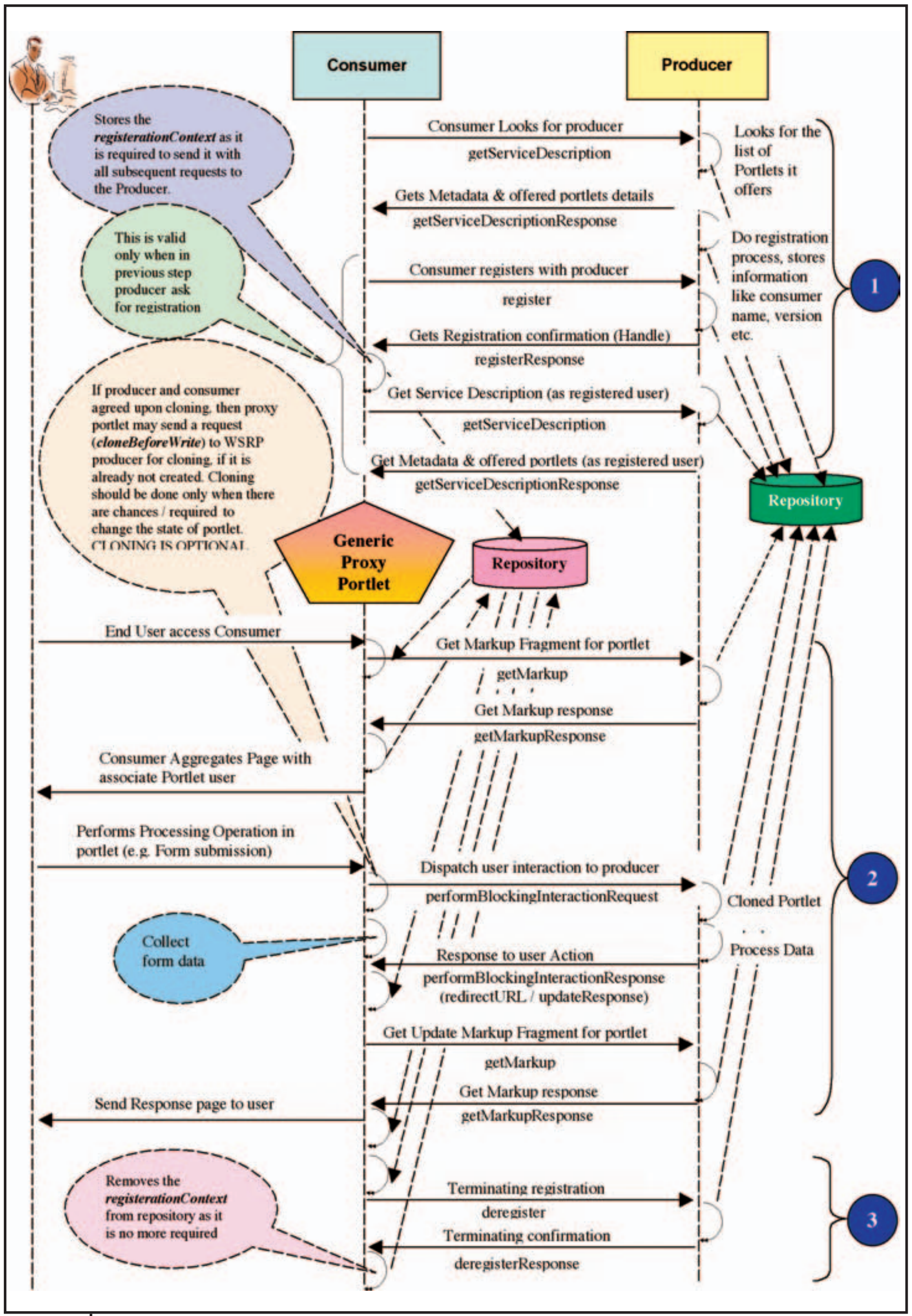

Consumer
Producer
Stores the *registerationContext* as it is required to send it with all subsequent requests to the Producer.
This is valid only when in previous step producer ask for registration
If producer and consumer agreed upon cloning, then proxy portlet may send a request (*cloneBeforeWrite*) to WSRP producer for cloning, if it is already not created. Cloning should be done only when there are chances / required to change the state of portlet. CLONING IS OPTIONAL
Consumer Looks for producer
getServiceDescription
Gets Metadata & offered portlets details
getServiceDescriptionResponse
Consumer registers with producer
register
Gets Registration confirmation (Handle)
registerResponse
Get Service Description (as registered user)
getServiceDescription
Get Metadata & offered portlets (as registered user)
getServiceDescriptionResponse
Looks for the list of Portlets it offers
Do registration process, stores information like consumer name, version etc.
1
Repository
Generic Proxy Portlet
Repository
End User access Consumer
Get Markup Fragment for portlet
getMarkup
Get Markup response
getMarkupResponse
Consumer Aggregates Page with associate Portlet user
Performs Processing Operation in portlet (e.g. Form submission)
Dispatch user interaction to producer
performBlockingInteractionRequest
Cloned Portlet
Process Data
2
Collect form data
Response to user Action
performBlockingInteractionResponse (redirectURL / updateResponse)
Get Update Markup Fragment for portlet
getMarkup
Get Markup response
getMarkupResponse
Send Response page to user
Removes the *registerationContext* from repository as it is no more required
Terminating registration
deregister
Terminating confirmation
deregisterResponse
3

FIGURE 10 **WSRP – Basic Sequence Flow**

As we know WSRP producers are modeled as containers of portlets. The producer provides a set of Web service interfaces and by implementing these interfaces, and agreeing to conform to WSRP, both producers and consumer can use a standard mechanism to offer and consume portlets. The following are the WSRP interfaces defined in the WSRP specification (See Figure 9):

- ***Service Description Interface:*** This is a *required* interface and it must be implemented by a WSRP producer to provide metadata of itself and the list of portlets it offers.
- ***Registration Interface:*** This is an *optional* interface. It is used to establish a relationship between a WSRP producer and a WSRP consumer. This interface provides the mechanism for a WSRP consumer to register with a WSRP producer and lets the producer customize its behavior for each WSRP consumer based on the registration information.
- ***Markup Interface:*** This is a *required* interface and it must be implemented by a WSRP producer to generate markup and to process interaction requests.
- ***Portlet Management Interface:*** This is an *optional* interface and it is implemented to grant access to the life cycle of the hosted portlets. This interface also includes property management, which enables programmatic access to a portlet's persistent state. In addition to this it also allows WSRP consumers to clone/destroy portlets, and also to customize portlets by changing any associated properties.

In order to understand the significance and usage of these interfaces, let's have a look at the sequence of basic flow (see Figure 10). We will be discussing the very basic and minimal basic sequence flow and some key points. For more details such as the method signatures, attributes, data structures to be used, etc., refer to the WSRP specifications. In the sequence diagram every sequence arrow in between consumer and producer has two parts:

1. First (shown above the arrow) – represents the functionality/activity
2. Second (shown below the arrow) – represents the method name/XML attribute name that comes into the picture while actually performing the activity defined above the arrow

The entire basic sequence flow is divided into three sections.

1. ***Initialization Section*** – Discusses the sequence of events that occurs between consumer and producer for handshaking/registration process for opening a communication channel
2. ***Interaction Section*** – Discusses the sequence of events that occurs between end user, consumer, and producer
3. ***Termination Section*** – Discusses the sequence of events that occurs between consumer and producer for termination/deregistration of the communication channel

## Initialization Section

1. The WSRP consumer obtains a description of the producer, and the list of portlets that the producer offers in order to aggregate portlets offered by a producer. For this the WSRP consumer sends a request for the *getServiceDescription* operation of the *service description interface*.
2. The WSRP producer looks up its repository and sends back a response (*getServiceDescriptionResponse*) to the WSRP consumer. This response holds the information such as whether registration is required or not, Locales supported, list of offered portlets, supported MIME types, and Window and Modes supported.
3. If the registration is required then WSRP consumer sends a request for the *register* operation of *registration interface* to the WSRP producer. Currently WSRP 1.0 supports two types of registration:
   - ***In-band registration.*** Here consumer provides all of the required information to the producer using the Registration Interface (also shown in sequence diagram).
   - ***Out-of-band registration.*** To establish registration, in this case both parties follows the business process that they mutually agreed upon such as Web applications or e-mail, etc. It is not standardized in the WSRP specification.

   For registration, the consumer provides DUNS (Data Universal Numbering System) and Service ID, which they received as a part of *getServiceDescriptionResponse*. In addition to this the consumer name, version, and Method GET supported information are also sent to the producer.
4. The WSRP producer validates this request for registration, creates a registration context, and sends back registerResponse to the WSRP consumer. The registration context contains a unique *registrationHandle*, which is valid for a lifetime of WSRP consumer's registration, and *registrationState* (optional).

*Note:* It is advisable that the WSRP consumer should store this registrationContext as it is required to provide this with all subsequent requests to the WSRP producer.

After this Generic proxy portlet is used for each end-user request. The generic proxy portlet is stateless in nature and it gets instantiated either when the first portlet is registered or at the first time a request comes for any of the portlets and it is dependent upon the WSRP consumer implementation.

Generic proxy portlets are generic portlets that eliminate the need to develop specific portlets for each Web service to plug into the portal. Because of these types of portlets, remote Web service portlets get included into the portal and the user accesses them as a local portlet. All user requests for any operation on any portlet of any WSRP producer go through generic proxy portlets. Basically a generic portlet does the same thing that HttpServlet does for us because it routes the request to the corresponding WSRP producer for a requested portlet.

From an implementation point of view, the generic proxy portlet in the WSRP consumer can be implemented in different ways. Some of the scenarios described below.

- ***Many to Many:*** There is only one instance of a generic proxy portlet that is being shared among all end users and the WSRP producer
- ***One to Many:*** There would be one instance of a generic proxy portlet that corresponds to each end user
- ***Many To One:*** There is only one instance of a generic proxy portlet corresponding to each WSRP producer

Each of these approaches has its own benefits and limitations in terms of performance, handling requests, etc., and although it depends on the WSRP consumer implementa-

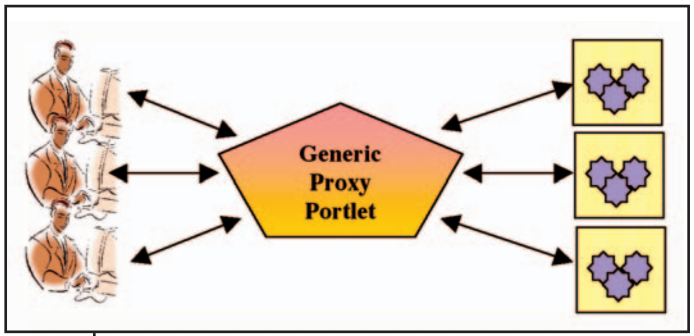


FIGURE 11 Generic portlet – many to many

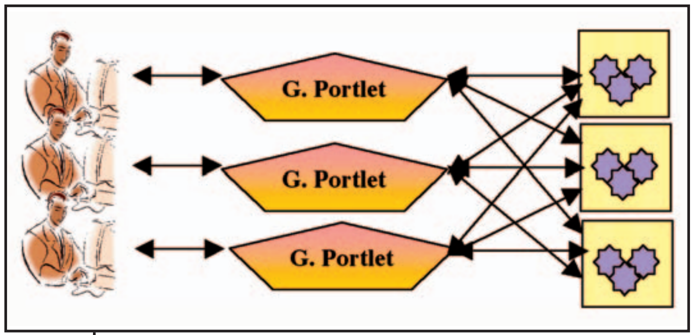


FIGURE 12 Generic portlet – one to many

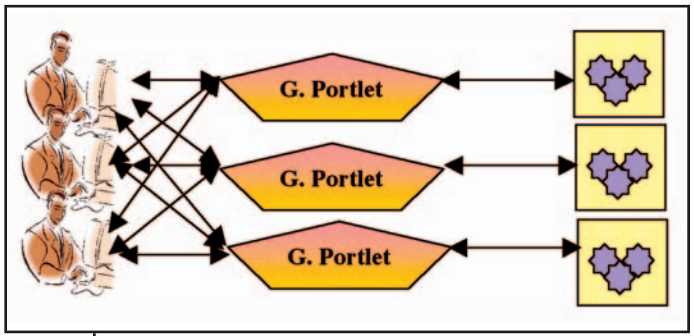


FIGURE 13 Generic portlet – many to one

tion, in most cases the WSRP consumer follows "many to many" approach.

## Interaction Section

Now by this time the WSRP consumer has established the communication with the WSRP producer and has gathered all of the information about services. Now in order to display an aggregated view to end users, it is required that the WSRP consumer must first obtain the markup of each portlet from each WSRP producer.

1. When end users access the portlets on the WSRP consumer portal/Web site, the WSRP consumer sends the request for the *getMarkup* operation of *markup interface* to the WSRP producer. While making the request along with *registerationContext*, the WSRP consumer needs to send information like end-user preferences, the locale and MIME types supported, user agent information (i.e., PDA, WEB, etc.) and protocol used (e.g., HTTP, HTTPS).
2. The WSRP producer sends back *getMarkupResponse* to the WSRP consumer. This response contains markup information (MIME type, locale, title, etc.) and session information (i.e., sessionID), and finally WSRP consumer sends the proper response to the end user for its request.
3. Now consider a scenario in which the end user does some processing information (i.e., form submission). For this WSRP consumers can send the request for *performBlockingInteraction* operation of *markup interface*, the submitting end user's form data to WSRP producer. During this time the portlet can perform various activities such as:
   - It can access and process the request data. It can change its navigational state and can make persistent changes to its state.
   - It can change its own current mode and/or window state.
   - The WSRP producer can choose to generate and return the portlet's markup as an optimization.
   - It can request the WSRP consumer for *redirectURL*.

   If there is a need, or it's required, or there is a chance to make changes in portlet state, then the WSRP consumer sends '*cloneBeforeWrite*' as a value for the *portletStateChange* element in the *performBlockingInteractionRequest* to the WSRP producer for creating the clone of a portlet. The WSRP producer creates a clone and returns a *portletContext* with a new *portletHandle* and/or *portletState*.
4. The WSRP producers send back *PerformBlockingInteractionResponse* based on the processing. This response contains *sessionContext* (i.e., new sessionID), *markupConext*, *redirectURL* (absolute URL)/*updatedResponse* (i.e., new *navigationalState* or we can say expected response), and finally the WSRP consumer sends back an appropriate response to the end user.

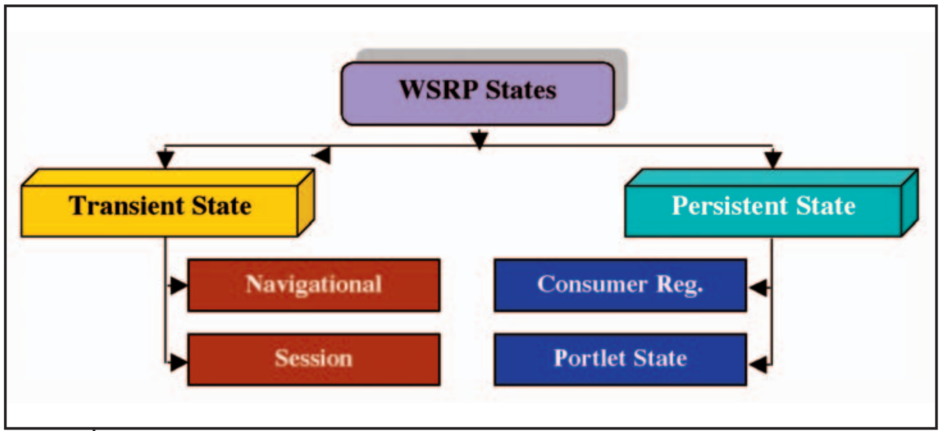


FIGURE 14 WSRP States

## Termination Section

Registration relationships are not permanent in nature – either of the two WSRP producers or WSRP consumers can terminate the registration. The termination process can be triggered because:

- The WSRP consumer no longer wants to aggregates the portlets provided by the producer in consideration
- The WSRP producer is no longer interested in providing the services to the WSRP consumer.
- There is need for re-registration because of the changes in the WSRP consumer environment.

1. The WSRP consumer sends a request for the *deregister* operation of *registration interface* to the WSRP producer.
2. In response to *deregister* request the WSRP producer sends the *deregisterResponse* after performing the cleanup process i.e., releasing the resources/states created for the WSRP consumer.
3. Upon receiving *deregisterResponse* the WSRP consumer is advised to perform the cleanup process i.e., releasing resources/removing the *registerationContext* from the repository, etc.

*Note*: Termination doesn't mean that the WSRP consumer can't re-register with the WSRP producer, but earlier states can't be restored.

## States Management

As we know, the WSRP protocol operates over connectionless technologies, so there is a need to maintain states that span across the different requests of an end user. WSRP provides support for two major types of states (See Figure 10).

### *Transient state*

This represents application-level state. There are two types of transient states.

- ***Navigational State:*** WSRP Producers require some data to generate markup for a given portlet several times during its interaction with a WSRP consumer. This state encapsulates data required and avoids the need to keep track of the interaction that caused the current state of the portlet. During the navigation the WSRP producer state doesn't hold transient state locally and even the user can store/bookmark the portlet navigation state. A Web application's URL plays key role in this because state is stored with the URL only so that both page refresh and bookmarked pages will generate what the end user expects. The WSRP producer returns this state to the WSRP consumer as navigational state so that it may satisfy these expectations of the end user.

- ***Session state:*** It is maintained with the help of *sessionID*, which is generated when portlets initialize the session. This sessionID moves to and fro in between WSRP producer and WSRP consumer.

### *Persistent state*

As the name suggests, this persistent in nature and will exist until either the WSRP consumer or producer explicitly discards it. Persistent state could be properties exposed by the WSRP producer via the *portlet management interface*, or some opaque state. There are two types of persistent states.

- ***Consumer Registration:*** It is a representation of a relationship between a consumer and a producer. Registration state is maintained with the help of *registrationHandle* generated during the WSRP consumer's registration process.
- ***Portlet State:*** The WSRP consumer allows creating unique configuration for portlets for its own use. Portlet state is maintained using *portletHandle*.

## How Can We Create, Use, and Test WSRP?

Various application server vendors and open-source projects provide support for WSRP (e.g., WebLogic 8.1, Oracle, Apache's WSR4J, etc.). As a practical example, to create a portal page that aggregates WSRP from different WSRP producers with WebLogic Workshop, you would take the following steps:

1. Open WebLogic Workshop.
2. Create a new Portal Application – enter the application name as *TestWSRPApp*.
3. Create a new Portal Web Project – right-click the *TestWSRPApp* directory in the Application window, choose *New ->Project*, and enter the project name as *TestWSRPProject*.
4. Create a new Portal – enter the portal name as *TestWSRPPortal.portal* in your project.
5. Create a new Portlet – enter portlet name as *TestWSRPPortlet.portlet* in your project.
   a. Select Portlet Type – Remote Portlet
   b. Find Producer (WSRP producer) – enter WSDL associated with the producer, for example:
      - BEA WSDL: http://wsrp.bea.com/portal/producer?wsdl
      - Oracle WSDL: http://portalstandards.oracle.com/wsrp/jaxrpc

   c. Select Portlet from List – select the portlet from list appears
   d. Proxy portlet Details – to enter Portlet Title say *BEA:eWorld*, to enter Producer' handle say *TestBEAHandle*

6. Select created portal and insert the selected remote Portlet, for example,
   select *TestWSRPProject -> TestWSRPApp->TestWSRPPortal.Portal* in left pane and insert the *BEA:eWorld* portlet in the portal page.
7. Start/run the project – this will open WebLogic's test browser, in which you can see your portal page. The URL for your portal page would be http://localhost:7001/TestWSRPProject/TestWSRPPortal.portal.

If we want to add some more portlets from the same producer, repeat steps 5c, 5d, and 6. If we want to add remote portlets from some other WSRP producer then repeat steps 5b, 5c, 5d, and 6.

## What Are the Limitations of WSRP?

We have read some great thing about WSRP – does that means this technology doesn't have any limitations? No, this is not true. WSRP also has some limitations, some of which are listed below.

- ***Security*** – WSRP doesn't provide any standardization for security and it only relies on the lower-level protocol. All security mechanisms applied for Web services are applicable for this also. For example, for secured transmission it has to depend on HTTPS. WSRP producers are responsible for authentication and authorization, etc. Keep one thing in mind: "Threats to Web services involve threats to the host system, the application, and the entire network infrastructure."
- ***Transaction Handling*** – There is no standard mechanism defined for handling the transactions.
- ***Response Time*** – There may be issues in response time. As portal page may aggregate portlets from different WSRP producers, and if any one of the WSRP producers is not on par with the others in terms of responsiveness, it definitely would effect response time.
- ***Reliability*** – Fault tolerance issues need to be handled at both the WSRP consumer level and the WSRP producer level. Normally, if some WSRP producer goes off then error is displayed in the portlet window. However it would be good if the WSRP consumer would define its own fault tolerance policy so that in case any of the WSRP producers goes off the portal page can still be rendered properly.

• • •

Many thanks to my colleagues Nitin Khare, Saurabh Bhatnagar, and Atul Singh Chauhan for their valuable contributions as reviewers and for their technical input in this article.

## References

- WSRP Specification: www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf
- WSRP FAQ: www.oasis-open.org/committees/wsrp/faq.php
- WSRP Overview: http://xml.coverpages.org/wsrp-overview200206.pdf
- WSRP Primer: www.oasis-open.org/committees/download.php/1237/wsrp-primer-draft-0.3.pdf
- IBM Document: www-106.ibm.com/developerworks/webservices/library/ws-wsrp/?Open&ca=daw-ws-dr
- JSR168 Specification: www.jcp.org/aboutJava/communityprocess/review/jsr168/
- Cover Pages: http://xml.coverpages.org/wsrp.html

### About the Author

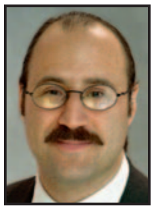
Rahul Kumar Gupta has more than seven years of experience in the IT industry with core expertise in designing and developing enterprise applications and middleware based on JAVA, J2EE, and Web technologies. He works with Indian IT giant HCL Technologies Limited, NOIDA (INDIA). He is also a co-technical reviewer for the Wrox Publications books Professional Java Ecommerce, Professional EJB, and Professional JSP Site Design, and is involved with other technical writings as well. He has eight certifications, including Sun Certified Enterprise Java Architect and Java 2 Programmer. For more details visit his profile at www.jguru.com/guru/viewbio.jsp?EID=4809 and view his publication profile at www.jguru.com/guru/viewpubs.jsp?EID=4809.

■ ■ ■ rahgup@mailcity.com

# The Well-Spoken SOA

## Understanding the elements of an SOA in the context of management, security, governance, and the power of words

The American comedian and actor Steven Wright once said, "It doesn't make a difference what temperature a room is, it's always room temperature." Words are wonderful that way. They can give you a little blast of pleasure when used cleverly, but like everything else they are subject to fashion. For example, I was speaking at a technical conference recently when I overheard a person whom I know, who is well-respected in this field, say something along these lines: "You have to know how well your SOA is running. Knowing the overall health and responsiveness of your SOA is very important. You've got to get a handle on your governance." The goal was laudable, but the wording was off target.



WRITTEN BY
**PAUL LIPTON**

I've heard the word *governance* fall from people's lips with increasing frequency recently, which is a good thing. Lately though, it seems to me that there has been an unfortunate blurring of the usage and definition of the word governance with another important word that also ought to be on the tip of the tongues of most people involved with SOA today, and that word is *management*. Monitoring and controlling the overall health and responsiveness of your SOA is largely a function of management, not governance.

The person whom I mentioned above probably knows this, at least in his better moments, but fashion is a powerful force. Trust me on this. You may consider yourself an up-to-date person both technically and in your style of language and dress, but I assure you, fashions change. Many years from now, photos of you wearing cloths that were once considered the height of fashion may cause your very own children to turn on you. There is no defense against the younger generation when they sense vulnerability any more than you can convince a shark in the midst of a feeding frenzy to try tofu. Speaking from a theoretical perspective, naturally, my advice is to be prepared for the likes of "Gee, Dad, how could you have possibly gone out in public dressed that way?"

A good response is to flash your progeny a peace sign and beat a hasty retreat.

Similarly, in order to spare our dear readers the potential embarrassment of explaining to future generations of telepathic IT people what an SOA was and why we even cared about it, it seems prudent to review and solidify our own architectural understanding. Let us consider the functional elements of an SOA starting with those elements responsible for the actual creation and execution of services. Later, we will focus on other essential elements such as management, governance, and security, and we'll examine their role in the SOA and their relationship with the rest of the IT infrastructure upon which the SOA depends, as well.

### Creation and Execution of Services

Many well-known types of enterprise software such as application servers, integration servers, and large business systems have evolved to provide the essential elements needed to create and run services in an SOA. Most often these are Web services based on protocols such as SOAP, but can include other types of services based on technologies such as CORBA or Java RMI as well. These newly evolved entities are often called *service platforms*. Service platforms minimally provide a service runtime environment for the execution of services, but are often bundled with tools that provide many other capabilities. Most commonly, they include development tools that provide the ability to develop and deploy services to that same runtime environment. Therefore, it is no surprise that most application servers and their associated development environments have been transformed and remarketed as service platforms.

SOA is also about breaking down the barriers between previously isolated legacy application silos and reusing these capabilities in new, more flexible ways. Therefore, both integration servers and messaging middleware vendors, which often have more specialized mechanisms in order to work with legacy systems, have joined the service platform game as well. In fact, a wide range of diverse platforms and technologies are transforming themselves into services platforms. For example, many application vendors such as SAP are also offering service platforms that provide the added benefit of leveraging the business application itself.

Many of these service platforms feature embellished tools that are helpful in designing and creating a modern SOA, including support for many Web services standards. These platforms are usually capable of composing simple Web services into more complex composite ones, and frequently provide orchestration engines so you can more easily create high-level business processes out of these services. They are designed to aid reusability by making it easy find new services via discovery mechanisms (typically UDDI registries), another element of SOA, which they often include as part of a complete service platform package.

Despite their architectural, technical, and functional diversity, one thing that many service platforms have in common these days is that they increasingly follow the current fashion of calling themselves an Enterprise Service Bus or ESB (a previously fashionable word was "fabric," but that has now fallen into disfavor). In my opinion, this was a smart move from the marketing perspective because it creates the impression of an indivisible and essential component. After all, what computer can operate without its bus?

However, unlike a computer bus, elements of an SOA related to the service-oriented applications themselves such as development, runtime, orchestration, transformation, guaranteed message delivery, or registry can also be provided by more specialized stand-alone products, depending on the needs of the organization. As these capabilities become increasingly mature and commoditized (a challenge that J2EE application servers started to face a few years ago), many organizations have already found that they have multiple ESBs and point-products with overlapping capabilities.

## Service Platform Limitations

For many organizations, the success of the SOA may ultimately be more dependent upon other SOA elements, such as operations management and security management. As different types of service platforms proliferate, the management and security challenges become more difficult. Why can't service platforms easily provide these capabilities in an SOA? They often promote themselves as "all you ever need to build an SOA." In fact many service platforms do provide limited management and security capabilities. However many service platforms are quite rightly focused on maximizing the benefits of their own technology stack, rather than leveraging and increasing the value and utility of all service platforms that participate in an SOA. Indeed, the service platform vendor may have limited experience or incentive to leverage the management and security capabilities of any platform but its own.

Theoretically, you could use any type of service platform in an SOA because your loosely coupled service consumers should not need to know what platform you are using, anyway. Also, as SOAs become increasingly complex and volatile, with messages being dynamically routed to services based on content, load, identity, and even the current prices or service levels of particular services, it might very well be that the underlying platform of a particular service is not the same from one day to the next. Under these circumstances, depending upon any one particular service platform to consistently apply your enterprise management and security policies across all other types of service platforms is likely to be problematic.

## SOA Management and Security

Platforms specific to SOA management and security have evolved to meet these challenges. In an SOA, services are message-centric. Most SOA management and security products function at the service-message level (for example, by monitoring and controlling SOAP traffic). They are designed to transcend the management and security limitations inherent in service platforms, and to supply more sophisticated capabilities while working well in a heterogeneous SOA. They typically offer support for SLA (Service Level Agreements), QoS (Quality of Service), fault reporting, policy definition and storage, auditing, and related capabilities across multiple service platforms.

Many SOA management systems also share some characteristics with service platforms, showcasing message translation or routing capabilities as "active management." Strictly speaking, this is not really management, per se. This is a capability shared by many elements of the SOA today including service platforms and even hardware. In my personal opinion, careful architecture and design, rather than dogmatic adherence to the idea of one central translation or routing point, is likely to serve most businesses better in the long run. Where you put your routing and translation may very well vary according to the task and the requirements.

"Monitoring and controlling the overall health and responsiveness of your SOA is largely a function of management, not governance"

At any level of the technology stack, management is about visibility and control. Historical record keeping and auditing are also important. Security and management often use similar technologies and techniques, but view things from a different perspective. For example, a denial-of-service attack is clearly a security issue (it may actually be intended to draw attention from a coordinated internal attack and is clearly an attack on the business in its own right), but it is also a management issue impacting load, performance, reliability, and more. Thus, management and security are closely related and some SOA management products are beginning to combine functionality in both areas, thus enabling SOA management and security policy to be defined and coordinated using a common interface, and providing a unified administrative perspective.

The market is crowded with numerous startup companies that sell various products in this space, although some have been acquired or have chosen to reinvent themselves in areas outside of management and security, in response to increasing pressure from the leading Enterprise Management vendors. At the time that I am writing this, CA has had a solution on the market for over a year and HP is believed to be preparing to ship a product of its own very soon.

## The SOA Does Not Exist in Isolation

It is important to note that most low-level services are themselves only a thin tier that encapsulates and depends upon a much deeper layer of existing business processes and logic. These business processes depend upon custom applications as well as on packaged applications such as ERP and e-mail systems. These systems work both in tandem with and depend upon other IT infrastructure such as application servers, Web servers, messaging and integration middleware, operating systems, storage, servers, routers, and so on. If any one of these diverse entities experiences a problem, it can have an effect on services at the SOA level.

The problem is that while an SOA management solution can certainly identify a problematic service by monitoring message traffic, it is not able to trace the underlying cause of a service's problem down to a particular infrastructure entity; nor can SOA management software monitor or control the lower-level infrastructure entities themselves to dig more deeply into the problem. The challenge becomes even more daunting when multiple infrastructure entities are contributing synergistically to a problem. In other words, the underlying business logic and the supporting IT infrastructure are completely invisible to the SOA management platform. How can the business determine the true root cause for SOA-level service problems caused by the underlying IT infrastructure?

The answer lies in the existing enterprise management and security systems that are already responsible for the overall health and security of the enterprise IT infrastructure. These existing enterprise systems have sophisticated event correlation and root-cause analysis that they apply with good effect to the existing IT infrastructure. In fact, the need for comprehensive, multitiered management and security is one reason why it is very common for well-managed and secure businesses to have sizeable investments in these enterprise systems already in place. In short, these systems often have established event correlation capability and are already helping to run the existing business processes even before most IT organizations began to consider an SOA.

It is these underlying enterprise management and security systems that must be leveraged to do the heavy lifting in order to perform the necessary event correlation and analysis for all of the parties invested in the SOA's success, including operations, security administrators, development, and line-of-business personnel. When SOA management software is appropriately integrated with existing enterprise management and security systems, it becomes possible to explore and to truly understand the operational and security state of the entire business from one end to the other, from the services that constitute the enterprise SOA down to the lowliest network device. But the bottom line is that in order for the existing enterprise management systems to perform this comprehensive event correlation and root-cause analysis, it is essential for SOA management and security systems to function, not in isolation, but fully integrated with the enterprise security and management systems that are already helping to run the business.

Many stand-alone SOA management systems available today maintain, at best, a shallow form of integration with enterprise management systems through the use of simple SNMP traps, but this is not adequate. It allows for things like simple text information to be conveyed through the enterprise management system to its central operations console. However, such limited information is usually insufficient for the sophisticated automated event correlation needed to determine the true root cause of high-level service disruption or delay.

Today's complex distributed systems generate a vast cascade of interrelated events that must be analyzed and understood by enterprise management systems, and SOA will add to this. SOA management systems that are not deeply integrated with existing enterprise management systems cannot easily determine the underlying cause of a problem. While SOA management solutions from enterprise management vendors are, by their nature, deeply and appropriately integrated and aligned with the rest of that vendor's enterprise management solution as a matter of course, other SOA management vendors will have to consider the huge development overhead of attempting to integrate with each of the unique interfaces of the leading enterprise management vendors, while at the same time responding to increasing pressure to differentiate themselves in more and more esoteric ways.

SOA management solutions from enterprise management vendors also benefit from natural alignment with the semantics, message terminology, and user interface provided by the enterprise management system – since a business's operations staff is already familiar with the enterprise management and security system. If an enterprise decides to deploy a second system for SOA that is not fully aligned with the existing enterprise system, this forces the operations staff to deal with error messages, terminology, and a user interface design that is different from the familiar enterprise systems that they are already using. In this case, it is important to factor in the added complexity, overhead, and cost of training operations staff in the particulars of two different management systems: one for the SOA and one for the rest of the business.

## Governance

Governance is an increasingly popular term and with good reason. Recently, some management vendors have tried to leverage the upswing in interest in governance by coining terms such as "runtime governance" for their management products, but to me those terms merely confuse the issue. Management and security, as we have discussed, are essential cornerstones of the SOA. The runtime health, safety, performance, and reliability of your SOA are most significantly a function of effective SOA management and security, not governance.

This invites the question of what exactly

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

# Gartner Application Development Summit 2005

**Service-Oriented Architecture and Agility:**

Are You Ready for the Next Step?

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

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

**Four Unique Tracks for 2005:**

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

**Register Now!**

**Visit gartner.com/us/ad or call 1 800 778 1997**

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

**Priority Code: ADWSJ**

**Special Keynote Guests**

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

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

Gartner Application Development Summit

is governance. A very simple but practical explanation is that governance is the management of development artifacts (some people would use the word *assets* instead), such as Java code, HTML, XML, COBOL copybooks, deployment descriptors, WSDL, etc. Governance is primarily about tracking and controlling development artifacts through their life cycles; from creation to archiving (it is usually not a good idea to destroy an artifact). Good governance is an important part of creating an effective SOA. While good governance is also dependent upon development discipline, architecture, and best practices at a number of levels, governance products can also add significant value and an important level of enforcement to that development discipline, and to the quality and usefulness of an SOA, especially as it evolves and responds to changing business conditions.

Good governance products manage the entire process, including human aspects such as human approvals for moving something from one stage (like test) into another stage (like production), checking that only the right persons can use or make an artifact public, and more. Governance products usually can make sure that artifacts that are deployed or created at certain stages of development for particular projects conform to company standards for that type of artifact and that project, and can track the location of those artifacts, which may reside in change configuration systems, databases, and other repositories. Some governance systems also provide their own repository, thus allowing certain or all artifacts to be centrally stored. The ability to interoperate with other repositories, registries, and other tools is also important.

In an SOA, registries based on UDDI are already starting to be used to facilitate discovery of services and obtain other relatively static information about those services, such as security policy requirements. In the future, I believe that more distributed, peer-to-peer mechanisms based on specifications such as WS-MetadataExchange will afford services the option of sharing information about themselves directly with their consumers at runtime without the imposition of a central intermediary. Depending upon business and architectural needs, many implementations of SOA may eventually use both approaches.

UDDI registries already contain pointers to important artifacts related to SOA such as WSDL documents, and can also point to many other types of artifacts. Because of this, many UDDI registry vendors have recently begun to market themselves as a governance solution. Their advantages include the fact that they are Web services standards-based and are already useful for discovery in an SOA. Of course, as they start to extend their core UDDI capabilities with custom governance enhancements such as approval screens and life-cycle processes, repository capabilities to store artifacts centrally, integrate with multiple development environments, and so on, they transform themselves from a pure, simple, standards-based discovery tool to a more complex Swiss Army knife of both standards-based and proprietary capabilities.

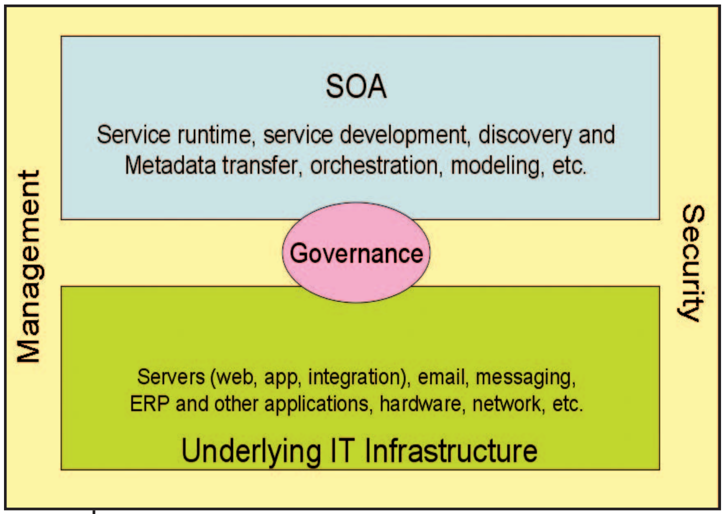

SOA
Service runtime, service development, discovery and Metadata transfer, orchestration, modeling, etc.
Governance
Servers (web, app, integration), email, messaging, ERP and other applications, hardware, network, etc.
Underlying IT Infrastructure
Management
Security

FIGURE 1 **Elements of an SOA in the context of the rest of the supporting IT infrastructure**

More traditional and established IT governance vendors are coming from the opposite direction. They often offer better integration with development tools, more sophisticated process mechanisms to manage the artifacts, and superior support for legacy artifacts such as mainframe systems, diverse languages, etc. However, they are newer to the world of Web services and SOA. They typically do not offer UDDI capabilities themselves and may not support or may not be as up-to-date on some Web services standards. Rather, they often choose to interoperate with UDDI repositories much as they do with other mechanisms that are external to their systems, but the focus is usually on governance and the development organization itself, rather than on the SOA and standards.

SOA management and security systems can add value to governance systems by providing relevant historical information about service levels, security, quality of service, and fault detection to the governance system. Since governance systems are about development artifacts, this allows developers to understand the long-term performance and reliability of their artifacts. In short, they can more easily ask questions such as which of my services is actually managed, which has the slowest response time, which has had the most security violations, or which matches certain service-level requirements that I need.

From the architecture perspective, the SOA management and security solution provides a centralized repository and mechanism for defining management and security policy for the entire SOA, across all service platforms. Governance platforms may potentially refer to and track changes to these management and security policies, which would be particularly useful to enterprise architects and designers already using these governance tools on a daily basis. However these policies still need to be stored and optimized for the use of the SOA management and security systems that are managing the environment in real time. Therefore, these policies will be most often stored within the management system's own repository. Similarly, because the SOA does not exist in isolation from other enterprise entities, these policies should be considered specialized cases of more general enterprise policies that already secure and manage the entire enterprise through existing enterprise systems.

Governance solutions, particularly UDDI-based repositories, can help management and security systems discover new or changed services, but there is nothing actually forcing service providers to publish WSDL descriptions of their services in a registry, so management and security systems must also be able to discover services based on message traffic, as well. Over time, management and security solutions will increasingly leverage information about development artifact changes provided by governance products in order to more effectively monitor the impact of these life-cycle events on the performance and reliability of the SOA at runtime.

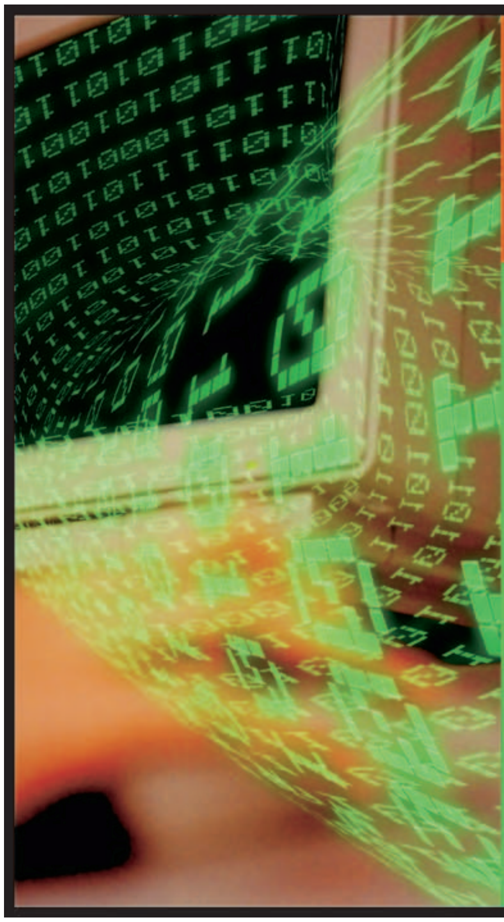
## Conclusion

Henry David Thoreau once said that "If you have built castles in the air, your work needs not be lost; that is where they should be. Now put the foundations under them." Similarly, many IT people have been focused on building their castles in the air by creating the beginnings of an SOA through the creation of carefully designed, loosely coupled services, often deployed on multiple service platforms. Many are starting to consider the role of dynamic service discovery through a registry, as well. This is all fine. But, as Thoreau said, now is the time to make sure you don't lose those castles. Be sure to build the needed foundations with effective SOA management and security. Even better, if you really don't want those castles to fall, make your foundations deep and strong by carefully considering the relationship and level of integration between your SOA management and security solution and the rest of your enterprise management and security systems. When you have a plan for complete, end-to-end management, and security covering every important aspect of your business processes and every supporting IT system, then your service-oriented castles are truly reliable, safe, and ready to add value to your business.

## About the Author

Paul Lipton is a senior architect in the Web Services and Application Management Team at Computer Associates (CA) and is affiliated as a strategist with the Office of the CTO. He has been an architect and developer of enterprise systems for over 20 years, and has worked closely with key CA customers to solve important business challenges through the creation of manageable, mission-critical distributed solutions. Paul has represented CA in numerous standards organizations, such as the W3C, OASIS, and the Java Community Process, and he is currently serving on standards committees involved in the definition of new Web services standards for management, orchestration, choreography, service state, and notification. He is a highly sought-after author and conference speaker, and has shared his knowledge with appreciative audiences around the world on such diverse topics as service-oriented architecture, Web services, Java/J2EE, .NET, management and security, EAI, On-Demand/Grid computing, XML transformation frameworks, databases, distributed systems, and wireless.

■ ■ ■ paul.lipton@ca.com



# 10 Things to Think About When Building the Perfect SOA

## Why we're learning more about making SOAs work the first time

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

WRITTEN BY
DAVID LINTHICUM

It's also important to note that our thinking is always evolving. As we learn what works, or perhaps more importantly, what does not work, we get closer to near perfect implementations that bring huge value to their enterprises or problem domains. Granted, some of this is trial and error. Let's explore each emerging pattern.

**1. Focus holistically, act locally**

SOAs are all-encompassing architectures, not mere projects, and those who consider them "projects" are doomed to failure. You're in for a pound when implementing an SOA due to the fact that, the more penetration the architecture gets, the more value it has within the enterprise.

However, since an SOA is the sum of its parts, you must consider the component parts as well, including notions such as identity management, semantic management, orchestration, etc., and how each part makes up the larger solution. An SOA is only as good as the weakest component, and neglecting a component will kill an SOA before it gets off the ground.

**2. Define the value**

As technologists we don't always want to make business cases for the technology. Indeed, we have a tendency to leverage the most popular technology without regard for how its use will affect the bottom line – not a good practice if you want IT to have a strategic position within an organization.

We build SOAs because they provide an infrastructure for agility, or the ability to change processes in support of a changing business, or both. They also allow for reuse: the ability to reuse application behaviors from system to system without having to port and retest code. Your ability to define a business case around these notions needs to occur before you begin your movement toward an SOA.

**3. Don't neglect service design**

At the end of the day, services are small, specific applications and need just as much attention paid to design. If SOA supports composite applications and composite processes made up of services, then the overall design of services will determine the overall success of things made out of those services…it's just that simple. Tried and true design techniques are applicable for service design as well. Please use them.

First and foremost, services should be designed for *reuse*. Services become a part of any number of other applications, and thus must be designed to provide behavior and information, but not be application specific.

Services have to be designed for *heterogeneity*. Web services should be built so that there are no calls to native interfaces or platforms. A Web service, say one built on Linux, may be leveraged by applications on Windows, Macs, even mainframes. Those that leverage your service should do so without regard for how it was created, and should be completely platform independent.

**4. Leveraging a legacy is unavoidable**

Embrace your legacy systems. In fact, they may be the majority of services that you leverage within your SOA. This means service-enablement of systems that you would consider old and outdated, but indeed serve a purpose within the business and thus serve a purpose within your SOA. Those who attempt to displace and replace existing systems, just to support new technology, are destined to make their work much more complex and far reaching than it need be.

**5. Remember the semantics**

If you don't understand application semantics, or, simply put, the meaning of data, then you have no hope of creating a proper SOA. You must understand the data to define the proper integration flows and transformation scenarios, and provide service-oriented frameworks to your data integration domain, which means levels of abstraction, as well as how data is bound to services.

You must always deal with semantics, and how to describe semantics relative to a multitude of information systems. There is also a need to formalize this process, putting some additional methodology and technology behind the management of metadata, as well as the relationships therein.

**6.The proper place for orchestration**

For our purposes we can define orchestration as a standards-based mechanism that defines how Web services work together, including business logic, sequencing, exception handling, and process decomposition, including service and process reuse. Orchestrations may span a few internal systems, systems between organizations, or both. Moreover, orchestrations are long-running, multistep transactions that are almost always controlled by one business party, and are loosely coupled and asynchronous in nature. While all SOAs don't need orchestration, most do, and you must find the right fit and application.

**7. Security soaks in as you execute; it's not an afterthought**

So, why do we need identity management? Also, why do we need to think about this stuff during the creation of our SOA? It's a fact that Web services are not for internal use anymore, and those who leverage Web services (consumers), or produce Web services (providers), need to be known to each, else we risk invoking malicious or incorrect behavior, which could cost us dearly.

Identity is important in the growth of sensitive data and confidential relationships online. If lacking identities, there is no way to provide certain users with access to certain resources. These relationships are complex, and can't be understood and created after the SOA is complete. The design of security is systemic.

**8. Classify the patterns of use**

As we build an SOA, we need to determine how the SOA will be leveraged within the enterprise – not only now, but in the future. Thus we need to determine patterns of use for several reasons, including:

- Understanding the value of the SOA
- Understanding how we will test the SOA
- Understanding how we can improve the SOA

**9. Persistence is important**

You need to think about persistence for a few reasons, including *federation of services* around the SOA. When building an SOA you may end up with composites or processes created out of services that may exist over a dozen or more different systems, and as such, persistence becomes more complex if done at the points. So, in these types of situations (which are becoming more common), it makes good sense to centralize the persistence for the composites and processes, as well as some supporting services to a *central data tier* or *central data service*. This data tier exposes a custom schema view or views to the composites, and may also abstract some data at the points as well.

“ If SOA supports composite applications and composite processes made up of services, then the overall design of services will determine the overall success of things made out of those services…it's just that simple ”

**10. Don't skimp on testing**

To insure proper testing, a test plan will have to be put in place. While a detailed discussion of a test plan is beyond the scope of this article, it is really just a step-by-step procedure detailing how the SOA will be tested when completed, using all of the patterns already discussed. A test plan is particularly important because of the difficulty of testing an SOA solution. Most source and target systems are business-critical and therefore cannot be taken offline. As a result, testing these systems can be a bit tricky.

New patterns continue to emerge, and each SOA is unique and deserves careful consideration. However, these patterns should provide a good base on which to plan your next SOA.

## About the Author

David Linthicum is the CTO at Grand Central Communications (www.grandcentral.com), and a leading expert in the application integration and open standards areas. He has held key technology management roles with a number of organizations, including CTO of both Mercator and SAGA software. David has authored or coauthored 10 books, including the groundbreaking and best-selling *Enterprise Application Integration* released in 1998. His latest book is *Next Generation Application Integration, From Simple Information to Web Services*.

■ ■ ■ dlinthicum@grandcentral.com

# Cordys BCP

## Building collaborative systems at the business level

Building truly collaborative systems relationships between organizations is a daunting challenge in today's business environment. While technologies such as Web services have risen to assist, true collaboration requires a far greater set of functionality.

WRITTEN BY
**BRIAN BARBASH**

The Cordys Business Collaboration Platform (BCP) provides a set of components designed to address the challenges of collaborative systems relationships across disparate businesses. The platform comprises a set of components that provides the ability to deliver orchestrated applications in a services-oriented architecture. Cordys BCP is composed of:

- **Cordys Studio** – The business process design component that provides for the modeling and design of business logic through a set of systems and applications.
- **Cordys Orchestrator** – Provides the tools to coordinate business processes and functions across organizations. Some of its key functions include the handling of process implementation, monitoring and tracking, and persistence management.
- **Cordys Integrator** – A component not unlike an EAI or ETL tool that provides the connectivity and transformation capabilities for communicating with existing systems.
- **Cordys Portal** – The main user interface for applications created in the Cordys environment.

For the purposes of this review, I will focus on the Studio and Integrator components of Cordys BCP.

### Working in Cordys Studio

Cordys Studio is the main development environment in Cordys BCP. It provides the environment used to model collaborative relationships between organizations in a top-down fashion, thus keeping the focus on business rather than on technology.

Three different types of models exist within the Cordys Studio environment:

1. ***Value Chain Models:*** These models represent the highest levels of an organization and are used to illustrate its core business relationships. They are analogous to Use-Case UML diagrams for software development (see Figure 1).
2. ***Business Context Models:*** The primary business function of an organization is represented in these models. They provide the basis for business process models and are in some ways similar to data flow diagrams (see Figure 2).
3. ***Business Process Models:*** The lowest level of model for an organization, they describe discrete processes within a value chain and business context and they are the programmatic elements instantiated at runtime.

For the purposes of this review, I have created a fictional business – BBConsulting – that will provide services to clients utilizing resources from subcontractors.

### Value Chain Models

Value chain units comprise value chain models in Cordys BCP. In the BBConsulting scenario shown in Figure 1, one unit represents the BBConsulting business while other units represent individual clients and subcontractors. These additional client and contractor value chain units are classified as business partners in Cordys Studio. Assuming BBConsulting is a healthy business, many value chain units are required to model all clients and contractors. To support large numbers of additional entities, Cordys Studio allows business partners to be categorized; in this case categories have been created for the client and subcontractor entities. The value chain model may then be composed of higher-level business partner categories, which provide for abstraction and generalization.

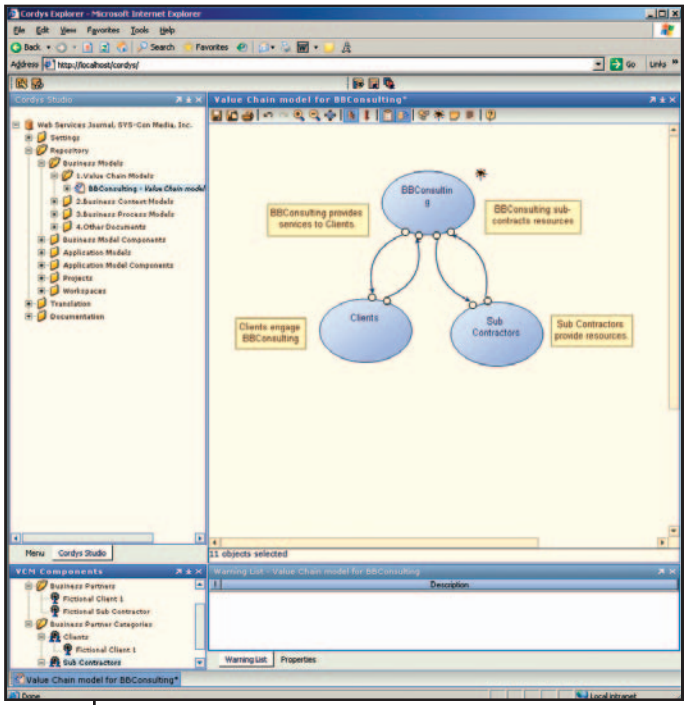

Cordys Explorer - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/cordys/
Go Links
Cordys Studio
Web Services Journal, SYS-Con Media, Inc.
Settings
Repository
Business Models
1.Value Chain Models
BBConsulting - Value Chain model
2.Business Context Models
3.Business Process Models
4.Other Documents
Business Model Components
Application Models
Application Model Components
Projects
Workspaces
Translation
Documentation
Menu
Cordys Studio
Value Chain model for BBConsulting*
BBConsulting
BBConsulting provides services to Clients
BBConsulting sub-contracts resources
Clients
Sub Contractors
Clients engage BBConsulting
Sub Contractors provide resources
11 objects selected
VCM Components
Business Partners
Fictional Client 1
Fictional Sub Contractor
Business Partner Categories
Clients
Fictional Client 1
Sub Contractors
Warning List - Value Chain model for BBConsulting
Description
Warning List
Properties
Value Chain model for BBConsulting*
Done
Local intranet

FIGURE 1 | Value chain model for BBConsulting

With the appropriate value chain units modeled, connectors are built between entities to represent the information flow between them. Finally, a View Point icon is placed above the BBConsulting value chain unit to identify the point of view of the model.

## Business Context Models

The business context models define the primary business function of an organization and serve as the path between high-level value chain models and low-level business process models.

In the BBConsulting example modeled in Figure 2, the primary function of the organization is to deliver services to its clients. Service delivery involves incorporating and utilizing resources from subcontractors. All activities are supported by an account management function.

With the context model in place for BBConsulting, it may be linked back to the BBConsulting value chain unit in the value chain model. The end result is a definition of BBConsulting's business relationships and primary business function. Similarly, business context models are developed for and linked to the client and subcontractor business partner categories.

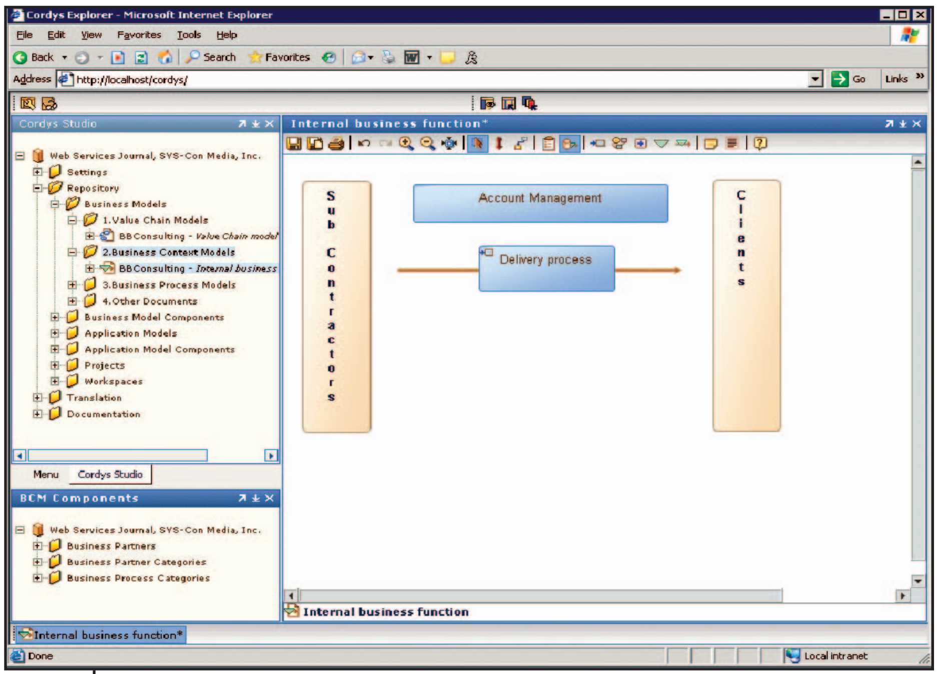

Cordys Explorer - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/cordys/
Go Links
Cordys Studio
Web Services Journal, SYS-Con Media, Inc.
Settings
Repository
Business Models
1.Value Chain Models
BBConsulting - Value Chain model
2.Business Context Models
BBConsulting - Internal business
3.Business Process Models
4.Other Documents
Business Model Components
Application Models
Application Model Components
Projects
Workspaces
Translation
Documentation
Menu
Cordys Studio
BCM Components
Web Services Journal, SYS-Con Media, Inc.
Business Partners
Business Partner Categories
Business Process Categories
Internal business function*
Subcontractors
Account Management
Delivery process
Clients
Internal business function
Internal business function*
Done
Local intranet

FIGURE 2 Business context models

## Business Process Models

Business process models are the lowest level entities within Cordys BCP. Modeled as flowcharts, they represent individual threads of activities that respond to certain business events. Process models contain the following basic components:

- **Start** – The starting point or trigger of the business process. Processes may be spawned upon the receipt of a message, they may be scheduled, or they may be initiated manually
- **Activity** – An activity is a set of actions that must be undertaken to achieve an intermediate goal within a business process. Either systems or users perform activities
- **Sub-Process** – A container used to model nested business processes. These are useful for breaking processes down into reusable or independent components
- **Decision** – Provides access to multiple pathways within a business process based on a defined set of criteria
- **End** – The end point of a business process

Figure 3 shows the business process model for the Invoicing process of BBConsulting. The process is initiated every 30 days and performs the following basic tasks:

1. Gather all Time and Expense reports
2. Determine if all resources on a project have entered Time and Expense reports
3. If any reports are missing, send a reminder to the appropriate resource and return to step 1
4. If all reports have been entered, generate an invoice for the client
5. Electronically send the invoice to the client

In this example, all activities are performed by a system. However, Cordys BCP also supports activities that require manual input. By assigning a role to an activity, it is marked for manual intervention.

Some of the unique characteristics of business process models within Cordys BCP include the ability to attach documents to activities. For example, the business process model that might be created for resources to enter their Time and Expense might include a link to the T&E policy document.

With all three levels of a business and its relationships modeled, the focus of the effort transfers to establishing systems integration touch points, and message formats and transformations within Cordys BCP.

## Systems Integration in Cordys BCP

Cordys Integrator is the main systems integration component within Cordys BCP. It provides a variety of application connectors that may be used with several external systems, some of which include:

- Databases via OLEDB or JDBC
- XML
- WebEx
- FTP servers
- Web services

Cordys Integrator also provides a Java interface for communicating with systems directly in Java, and a custom Application Connector framework for connections to systems that are unsupported by any of the other provided mechanisms.

For the BBConsulting example, two integration touch points are required: the BBConsulting Time and Expense system and an e-mail interface to route invoices to clients.

The BBConsulting Time and Expense system is available as a Web service hosted on a local machine. Cordys Integrator can build references to Web services either by

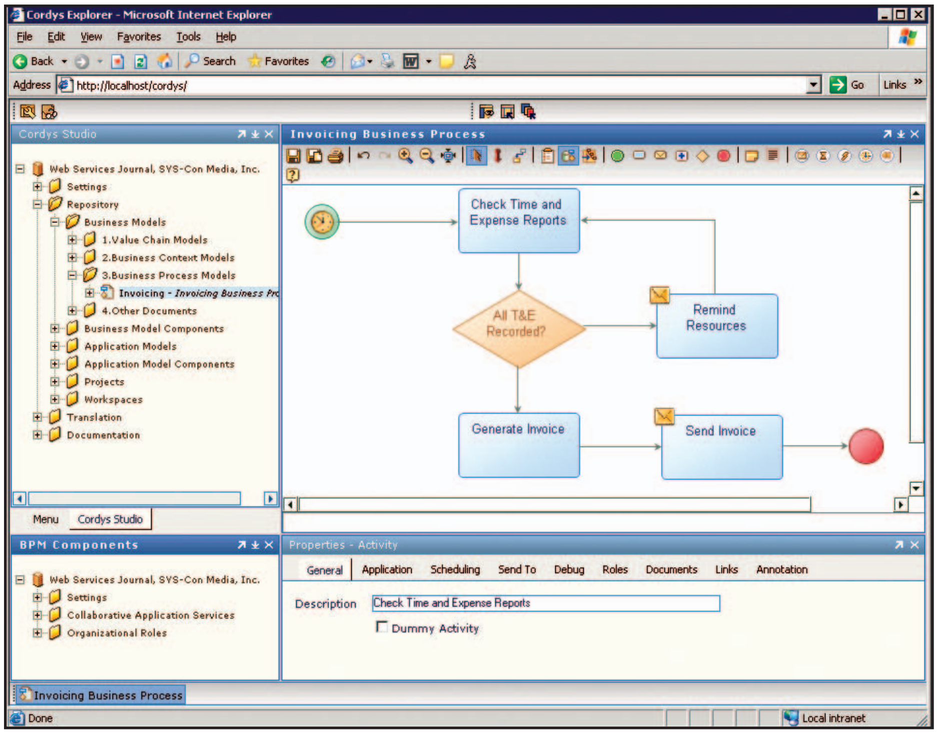

Cordys Explorer - Microsoft Internet Explorer
Address http://localhost/cordys/
Cordys Studio
Web Services Journal, SYS-Con Media, Inc.
Settings
Repository
Business Models
1.Value Chain Models
2.Business Context Models
3.Business Process Models
Invoicing - Invoicing Business Pro
4.Other Documents
Business Model Components
Application Models
Application Model Components
Projects
Workspaces
Translation
Documentation
Menu
Cordys Studio
Invoicing Business Process
Check Time and Expense Reports
All T&E Recorded?
Remind Resources
Generate Invoice
Send Invoice
BPM Components
Web Services Journal, SYS-Con Media, Inc.
Settings
Collaborative Application Services
Organizational Roles
Properties - Activity
General
Application
Scheduling
Send To
Debug
Roles
Documents
Links
Annotation
Description
Check Time and Expense Reports
Dummy Activity
Invoicing Business Process
Done
Local intranet

FIGURE 3 **Invoicing business process model**

communicating with a UDDI server, or by downloading the WSDL directly. For the purposes of this example, the WSDL approach will be used.

Within Cordys Integrator, the Method Generator tool may be used to read a WSDL file from a specific URL. By entering the WSDL URL of the Time and Expense service, the generator displays the operations defined by the Web service. Importing the operations to Cordys is as simple as selecting them from the list. In this case, the operations GetTimeSheetsByProject, GetExpenseReportsByProject, and IsProjectReadyToInvoice will be utilized.

Once the Web services operations have been imported to Cordys Integrator, they automatically become available in Cordys Studio. Within Cordys Studio, Web service operations must be imported to the business process model that will utilize them. Operations may then be assigned to the appropriate activities simply by dragging and dropping the definition from the Repository to the activity.

With all external operations made available to the business process model, the final piece of the integration puzzle is defining the messages that pass among the activities. Since all messages within Cordys BCP are defined as XML documents, message inspection and transformation is handled via XPath statements using the Message Map tool.

For the BBConsulting invoicing process, the key point in the model is the Decision-gate that identifies whether all Time and Expense data has been recorded. Each output connector provides a Condition statement that must evaluate to true for the process flow to proceed down that path. The Condition will be an XPath statement that determines whether to invoice or remind project resources to complete their reports:

```
Proceed to Generate Invoice: /Projects/
Project/TimeAndExpenseComplete = True
```

```
 Proceed to Remind Resources: /Projects/
Project/TimeAndExpenseComplete = False
```

Once the business process model is complete with all message transformation and integration, the model may be deployed to the runtime environment for invocation.

## Summary

Building a collaborative relationship between organizations at a business and systems level is a highly complex challenge. Each partner must fully understand the business processes to be integrated and the impact of those processes on the parties involved. Systems and data must be woven together in such a manner that effective and efficient exchange of information takes place with enough flexibility that each partner may evolve and expand its business without impacting the other. Technologies such as Web services are available to technically facilitate such relationships, but a broader set of tools and collaborative capability is required at a business level. The Cordys Business Collaboration Platform is built with this complexity in mind. It allows business partners to visualize their relationship at the highest levels, understand the key functions and strengths of each business, and then model and integrate the individual business processes appropriately. Cordys BCP is a solid solution that can address the challenge of business collaboration.

## About the Author

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

■ ■ ■ bbarbash@sys-con.com

“Importing the operations to Cordys is as simple as selecting them from the list”

# When Exceptions ARE THE RULE

## Achieving reliable and traceable service-oriented architectures

Every now and then, an IT glitch makes national news. Just a few weeks ago, I read in the paper about an airline that mistakenly sold thousands of roundtrip tickets online at a fare of just a few dollars each. The airline lost hundreds of thousands of dollars from the mistake, though that's really just the tip of the iceberg. The National Institute of Software Technology (NIST) estimates that application errors cost the U.S. economy $59.5 billion per year. Because nearly 80 percent of such errors are discovered after applications have been put into production, exceptions also have a significant impact on the productivity and effectiveness of your IT staff and production support teams. And that's to say nothing of foregone revenue due to poor customer service.

WRITTEN BY
SEAN FITTS

We call these unexpected conditions "exceptions," though they happen all the time. They are as unavoidable as they are harmful.

Unlike my highly publicized airline example, the people who know about exceptions are usually limited to customers, IT teams, and line-of-business managers – and they typically find out about exceptions in that order. It's the customer or the end user of an SOA-based business application who is usually the first to witness the consequences of an exception. Common symptoms include opaque messages (such as "Sorry, unable to process request at this time") on Web sites. Such seemingly mild errors eventually translate into more disruptive business exceptions such as delayed orders, lost packages, rejected insurance claims, and so on.

Due to their distributed and heterogeneous nature, services-based systems are inherently vulnerable to exceptions. Exceptions in SOA environments can be broadly categorized into three classes:

- **System-Level Exceptions** result from XML/SOAP processing errors or transmission failures. They surface as SOAP faults with inconsistent fault codes.
- **Application-Level Exceptions** often come from incorrect message semantics or logical errors within the application. Incorrect data, unchecked boundary conditions, unexpected service, and client responses can be the cause.
- **Business-Level Exceptions** denote unacceptable business states in a transaction and are not necessarily technology-related issues. These surface as events that violate best practices, compliance laws, regulations, or business policies mandated by business managers. They include both technology-driven issues (such as an important order not processed within 24 hours) and human errors (such as an incorrect shipping address in a purchase order).

## SOA: A Haven for Exceptions

Any developer will tell you that 80

percent of the time required to diagnose an exception is spent simply trying to replicate the scenario. That's due to all the effort of searching through log files and iteratively recoding to add more information to the log. With distributed SOA, possibly stretched across geographies, this task is even more challenging.

Managing exceptions has traditionally been an expensive, extremely manual effort performed by often dedicated application maintenance teams. Since their clues reside in multiple messages that span different services in the business application, exceptions in SOA systems are even harder to detect and diagnose. To begin with, the applications themselves are seldom instrumented to proactively alert on specific exceptions. At best, an application surfaces exceptions as incoherent entries in an error log, such as "Error 00021C: Transaction rejected." These exceptions might be uncovered during routine maintenance of the application. However, as noted earlier, it's the phone calls from vexed customers that usually make IT staff aware of exceptions. Business operations teams seldom come to know about exceptions until it is too late to respond.

So what are you to do about it? You could just shrug your shoulders, accept that exceptions are going to happen, and hope you're not next week's headline news. Or you could look for ways to detect, diagnose, and remedy exceptions before they bring your business to a standstill. Let's explore ways to go about the latter option.

## Managing Exceptions in Services-Based Systems

To understand how to manage exceptions in SOA-based environments, start by considering the requisite capabilities. Exceptions must be detected as they occur. To do so, IT and business operations must be able to specify the criteria for spotting exceptions in live business transactions. Typically, you'd look for message patterns that indicate unusual business activity. These may include incongruent reference data, discrepancies in data fields, error messages, and error codes. Sometimes, criteria can be crafted for detecting very specific conditions – for example "raise an exception if a premier customer's order is rejected due to mainframe error code D234200." Other times it's the absence of a message or pattern that's the symptom of an exception.

Since it's impossible to anticipate all patterns, operations teams need to cast as wide a net as possible across their business systems to trap the maximum number of exceptions. As a fallback, they must be able to trace and record all distributed transactions and diagnose this data for the root causes of exceptions.

IT and business teams must know about exceptions immediately. It might be important to alert one or more individuals across different teams based on the nature of the exception. For example, the error code is of interest to the IT staff, while the rejected purchase order and the customer details are important to business types.

The notified personnel must then be able to quickly analyze the situation, understand its cause, and implement a cure. To accomplish this, they need to know not only the exception message pattern but also the context of the business transaction in which it occurred. IT operations must be able to diagnose and resolve the exception in minutes and seconds instead of days and hours. Similarly, business operations must be able to learn about exceptions in real time in order to formalize a resolution before customer service is affected.

For some exceptions, the resolution is clear. In such cases it's important to resolve the exception in-flight by applying automated exception-handling actions.

## Why Traditional Approaches Don't Work for SOA

Programmatic exception-handling models have been the mainstay of exception management in business applications. The compilation stage detects and eliminates syntactic errors. Anticipated anomalous business conditions are detected and handled via embedded logic, either in the application source code or in the business process driving the application. Business Process Management (BPM) systems often handle exceptions in process definitions by hardwiring the process definition with corrective actions for a well-defined set of exceptions that might occur while executing the process. Unanticipated conditions and process exits are handled by writing the condition to a log file.

Debugging and testing practices aim to isolate and eliminate logical errors. Quality assurance teams spend countless hours putting the software through scripted production simulations. Then it's up to the consumers of the production systems to report any exceptions to the technical support organization. IT operations staff members depend on applications and system logs to diagnose problems reported by customers. Patches are applied to applications if problems are deemed severe. Additionally, Network Systems Management (NSM) software is used to isolate runtime failures in the hardware or in elements of the physical layer and to trigger alerts.

It's important to note, however, that these approaches are based on the following assumptions:

- The entire system is monolithic
- The functional requirements are driven by a single business application
- A central managing entity controls the message flow in the environment
- Changes in the application components are administered centrally
- Exceptions are expected to be limited and well defined

The problem is, SOA-based applications are distributed, heterogeneous and federated. Specifically:

- Application components are shared and reused by many applications
- Not all application code used in an application has been tailor-made for that application
- SOA-based systems are dynamic and real time; application components can change independently

Traditional exception-handling approaches rely on the developer's ability to anticipate all possible exceptional conditions and embed custom instrumentation (as code) into the application to detect and handle them appropriately. Yet, it's impossible for anyone to imagine all of the types of interactions an end user might have with the services of an SOA-based application and code to prevent every possible runtime glitch that might occur.

Coding-based approaches are inapplicable to services you don't own or don't control. Such hardwiring also means that any change or update to the exception-handling capabilities already fabricated within an application requires programmatic changes to the application. IT is unable to respond in a timely fashion to the vagaries of a real-time business environment. Business requirements remain unmet, while IT maintenance overheads rise.

Distributed systems introduce a new order of complexity in detecting exceptional conditions. Since the custom instrumentation is isolated to the service into which it has been programmed, it has no context of the actual business transaction into which it is participating. Exceptional conditions that are related to business transactions spanning one or more Web services across different business applications cannot be detected by exception handling localized within one of the participants. This is particularly true of exceptions that materialize as business events.

What's more, traditional approaches rely on the developer, leading to resource-intensive IT fire drills. To the IT staff, exceptions appear as entries in an error log that simply state something along the lines of "Error 00021C: Transaction rejected." The amount of information available to diagnose the problem is limited, and is often solely dependent on whether or not the developer had been meticulous about exporting all of the available information to an application "log." There is no way to know how a Web service was consumed, when the exception occurred, and in what context. Also, there's no easy mechanism for the IT staff to capture the flow of business transactions end-to-end within the system, as they occur.

In a distributed environment, log entries are likely to be collocated with the Web service in question. Thus, diagnostic information is fragmented across one or more logs, potentially at geographically different locations, or at least on different servers within the same organization. Typically, IT operations teams spend hours sifting through different logs trying to manually piece together the different pieces of the puzzle.

## Exception Management Befitting SOA

For dynamic, service-oriented business environments to work, runtime exceptions must be detected and intercepted in-flight and localized resolutions must be executed to either eliminate these exceptions or to mitigate their effect on the rest of the system. The question becomes how to accomplish this. What's needed is a dedicated piece of infrastructure (whether you build it yourself or purchase it as a dedicated solution) that allows you to detect, diagnose, and resolve exceptions. This is the Exception Manager.

The Exception Manager spans a network of services, and each service serves multiple applications within the business environment. It is chartered to:

- Identify the set of services that serves each business transaction
- Identify the distributed interactions within the scope ("context") of the business transaction
- Provide a "safety net" to actively spot exceptions and automatically respond
- Provide a record of interactions and tools for after-the-fact diagnosis

The Exception Manager has the ability to detect both software errors (system or application errors) and business events in services-based systems and then act upon these conditions in real time.

Abstracting of exception detection and handling from the actual application creates a rapidly adaptive business system. It also allows you to deal with exceptions in system components that are beyond your direct control, such as a partner's Web service.

Now let's take a look at how an Exception Management System might be implemented.

SOA offers a standards-rich environment that enables "intermediaries" to gain runtime insight into the behavior of an application at both the syst em level (such as message sequences and formats) and the business level (such as order number or order amount), via machine-readable XML message content. These intermediaries can also participate in services interactions in real time. A combination of these capabilities can allow "intermediaries" to facilitate intelligent management of exceptions.

Intermediaries can take many forms. An autonomous software agent is probably the best suited to the role of an intermediary. It is able to intercept the live messages, understand their content and context, and initiate appropriate management activities upon detecting exceptions. A collection of such intermediaries can record interactions as well as detect and act upon exceptions in applications spanning multiple services, thus laying the foundation for a comprehensive Exception Management System.

Such an intermediary-based system offers many advantages over traditional approaches for managing exceptions. Unlike a proprietary application execution server or message bus, the agents do not require control of the environment. Furthermore, this type of "on-demand" participation can be administered into this environment without writing new code or altering existing services and applications. Finally, the intermediaries are able to listen for system errors and business events generated by incumbent IT systems while allowing the delegation of appropriate system-level tasks or business activities to these systems via SOA standards.

The time and expense of implementing an effective Exception Management System pays for itself in short order. The maintenance overhead saved when detecting, diagnosing, and resolving the multitude of exceptions that can occur is significant. Business systems become more dynamic, thereby enabling your organization to more quickly and reliably adapt to narrow windows of market opportunity. Customers are better served, and your systems are less likely to lose your company revenue or gain your department coverage on CNN due to an unanticipated glitch.

### About the Author

Sean Fitts is the chief systems architect for AmberPoint, Inc., the leading provider of SOA management software. Prior to AmberPoint, Sean held positions as lead architect and engineer at Forte Software, where he guided the overall architecture of the SynerJ product suite. Sean also held positions as senior software engineer at Sybase and Management Dynamics. He has a Bachelor of Science in Electrical Engineering/Computer Science from Princeton University. He has been awarded a patent, and has another pending.

■ ■ ■ sfitts@amberpoint.com

# Differential QoS Support in Web Services Management

One service implementation – many levels of service

Web services are gaining acceptance as the prime technology used to interconnect disparate applications and ease interoperability between heterogeneous and autonomous systems both for internal and external integration. The variation of contexts in which shared Web services could be used and the resulting variation in functional and Quality of Service (QoS) requirements motivate extending Web services management platforms with more sophisticated control mechanisms to cater to differentiated service offerings.

WRITTEN BY
ABDELKARIM ERRADI,
AKASH SAURAV DAS,
ABHISHEK CHATTERJEE,
ANSHUK PAL CHAUDHARI,
TERANCE DIAS

However, most Web services platforms are based on a best-effort model, which treats all requests uniformly, without any type of service differentiation or prioritization. This article explores the typical generic requirements for differential QoS support in Web services management. We then evaluate various emerging management frameworks to assess the degree to which they meet the identified requirements. Finally we present the typical architecture for priority-based differentiated QoS for Web services.

## Web Services Management

Web services management (WSM) provides the necessary infrastructure to help enterprises monitor, optimize, and control the Web services infrastructure by using a policy-oriented approach to ensure that QoS objectives are met. A WSM system provides visibility into the Web services runtime environment to enable:

- Monitoring of availability, accessibility, and performance of Web services
- Service level agreement (SLA)–compliance tracking
- Error detection, resolution, and auditing

OASIS Web Services Distributed Management (WSDM) is a key standard for Web services management. It allows exposing management functionality in a reusable way through two specifications: one for Management Using Web Services (MUWS) and the other for Management of Web Services (MOWS). The MUWS specification provides a framework that defines how to represent and access the manageability interfaces of resources as Web services. MOWS builds on MUWS to define how to manage a Web service as a resource. It defines WSDL interfaces, which allows management events and metrics to be exposed, queried, and controlled by a broad range of management tools. However, WSDM does not provide any support for differential QoS.

## Differential QoS Requirements in Web Services: One Service Implementation – Many Levels of Service

In the business world differentiated services are very common, just as gold card holders get preferential services compared to silver or bronze card holders. Leveraging the notion of differentiation to vary the level of service offerings in Web services environments is a challenging task. For example, a Stock Quote Web service could report prices with different levels of timeliness, ranging from real time to fifteen-minute delay to a 24-hour time lag. Service request metadata such as the grade of the service customer (gold, silver, etc.), or the security level (employee, customer, partner, etc.) could then be used to determine the service level to offer to the user at the time of request.

However, current Web services management platforms do not address differentiated services requirements comprehensively, and requests are often processed according to their order of arrival. Figure 1 summarizes the typical architectural requirements for differential QoS.

**Generic service-level offerings –** The first requirement to achieve differentiated QoS is the ability to encode QoS assurances in machine-readable format. An expressive language is required to specify the various service levels offered by the service provider. For example, services can be offered using an Olympic classification such as gold, silver and bronze.

**Individualized service-level offerings –** Besides the support for defining generic service levels, sometimes it is useful to be able to define custom service levels (SLAs) per service consumer.

**Matching and negotiation of service levels –** Particularly for custom SLAs, sometimes it is useful to be able to dynamically negotiate service levels depending on the requester's requirements, the service provider's capabilities, and the dynamic runtime conditions such as service load.

**Admission control –** Admission control regulates the "intake" and manages the acceptance of new requests while taking into account current service policies and system load. In case of overload, a request may be either rejected or have its QoS level downgraded (through negotiation), so that it can be accepted at a lower service class.

**Request classification –** Request classification is responsible for receiving incoming requests and assigning a service class to requests to enable prioritization according to a predefined classification schema. The request is then placed on the appropriate priority queue according to its assigned service class.

**Requests policing –** Requests policing is required to make sure that the number of requests per customer is within a predefined limit. Requests exceeding the maximum throughput limit are assigned a low priority class.

**Differential QoS dispatcher –** Dispatcher uses scheduling algorithms such as weighted round-robin scheme to dispatch requests for processing, while ensuring that the number of dispatched requests to each service does not exceed its capacity.

**QoS measurement and monitoring –** A QoS metrics engine is required to gather QoS metrics, monitor the workload, and readjust the class-of-service weights for request queues. This allows the Service provider to ensure that the promised performance is being delivered, and to take appropriate actions to rectify non-compliance with an SLA such as reprioritization and reallocation of resources.

**Policy manager –** For implementing differentiated services, there is typically a set of rules or policies that control the QoS variations. From an architectural point of view, this requires a separate policy management layer to allow separating out the policy aspects from service implementation.

**Automated resource management –** The most important and challenging requirement for differential QoS management is still QoS delivery through automated resource management, particularly arbitration in allocating resources to client requests (e.g., providing more resources to process higher priority requests) and mapping QoS requirements onto the configuration of underlying QoS provision technologies across all abstraction layers.

## Evaluation of Differential QoS Support in Emerging Web Services Management Frameworks

So far there are no established standards for formally specifying QoS assurances and classes of service in machine-readable ways. However, various XML-based languages have been proposed such as Web Service Level Agreement (WSLA) and Web Services Offerings Language (WSOL). Such languages help define contracts to specify agreed-upon, nonfunctional characteristics of Web services (in the form of Service Level Objectives – SLOs) as well as a model for measuring, evaluating, and managing the compliance with these characteristics.

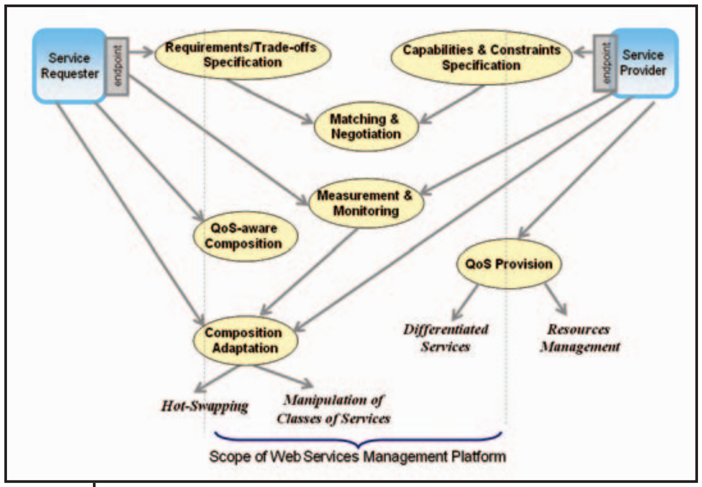


FIGURE 1 Web services QoS dimensions

The WSLA framework consists of a flexible and extensible XML language for the specification of custom-made SLAs as well as a management infrastructure that comprises several SLA enforcement and monitoring services both for dynamic allocation of resources and for compliance checking.

WSOL enables the specification, monitoring, and management of classes of service for Web services, instead of custom-made SLAs. WSOL is the only framework that has an explicit notion of class of service. A class of service refers to a formal representation of a discrete variation of QoS guarantees provided by a Web service. WSOL also provides mechanisms for the specification of static and dynamic relationships between different classes of service. The latter are used in the algorithms and protocols to dynamically modify the QoS levels offered by a service. The corresponding Web Service Offering Infrastructure (WSOI) enables monitoring of Web services described in WSOL and implements the dynamic adaptation algorithms and protocols.

In terms of supporting negotiation of service level, WSOL only allows basic negotiation before switching to a different service level at runtime. The "autoManipulation" attribute determines whether the provider is allowed to perform switching to a different service offering without asking the consumer for explicit confirmation. If the value of this attribute is "True" the provider does not need to ask for consumer confirmation before switching. However, the provider has to inform the consumer after the switching. On the other hand, WSLA has more powerful negotiation capabilities that are further explored in the WS-Agreement specification.

For admission control, WSLA uses advanced control mechanisms that take into account the current load and available resources, whereas, WSOL uses simple access rights to specify conditions under which a consumer of a service offering has the right to invoke a particular operation. For example, an access right can limit the time of day when an operation can be invoked. It can also limit the number and/or frequency of invocations.

## Inferences

WSLA was developed as part of IBM's autonomic computing initiative, and seems to be the most complete differential QoS framework. WSLA is extensible and allows logical expressions such as trade-offs between QoS parameters to be expressed; it also allows failure behavior to be specified. It defines a framework for providing differentiated levels of service through automated management of resources according to the negotiated SLAs. Some extended sets of WSLA abilities have been already integrated in IBM WebSphere Extended Deployment (XD). The WSLA project spawned into a number of efforts like WS-Agreement standardization in the Global Grid Forum. However WSLA is relatively more complex and introduces high negotiation and management overhead.

WSOL on the other hand is a simple and lightweight framework used to describe and manage multiple classes of service for a Web service, but it does not address how to provide the offered service levels.

The proposed approaches for expressing differential QoS assurances are still immature in terms of software implementation and experimental evaluation. Also they are silent when it comes to QoS provision, except for WSLA. Furthermore, the absence of a common QoS management standard might impede interoperability. The differential QoS space remains fragmented and a unifying framework is highly needed to tie up all of these approaches into a coherent framework. For example the distinctive features of WSOL could be added to WSAL and the two can be integrated with WS-Policy, with the latter serving a container for assertions.

The requirement of automated enforcement of machine-readable policies that control the variation of differentiated services has not been addressed in any of proposed solutions. However, IBM's Policy Manager for Autonomic Computing (PMAC) can be seen as a step towards this.

The biggest challenge facing the provision of differentiated services is the automated mapping of service offerings or SLA requirements to resources requirements across abstraction layers in order to determine the detailed resource configuration necessary to meet the promised QoS assurances. This requires accurate service performance modeling, capacity management, and timely information on available resources. The second challenge is the integration of differential QoS mechanisms that are available at different levels along the message path: network, application servers, and SOAP layer. These challenges are further complicated by the virtualization of services and the composition of Web services into added-value aggregate services. Grid and autonomic computing initiatives are the space to watch for potential answers to these challenges.

| Emerging Standards | | |
|---|---|---|
| **Differential QoS requirements** | **WSLA** | **WSOL** |
| Generic service-level offerings | ✗ | ★★★★★ |
| Individualized service-level offerings | ★★★★★ | ✗ |
| Matching and negotiation of service level | ★★★☆☆ | ★☆☆☆☆ |
| Admission control | ★★★★☆ | ★☆☆☆☆ |
| Request classification | ★★★★☆ | ✗ |
| Requests policing | ★★★★☆ | ✗ |
| Differential QoS dispatcher | ★★★★☆ | ✗ |
| QoS measurement and monitoring | ★★★★☆ | ★★★☆☆ |
| Policy manager | ✗ | ✗ |
| Automated resource management | ★★★☆☆ | ✗ |

FIGURE 1 **The degree of support for differential QoS requirements by key emerging differential Web services management frameworks**

## Typical Architecture for Web Services Management with Differential QoS Support

The state-of-practice regarding QoS differentiation is to provide different service interfaces for different service levels or to specify an additional parameter in the operation signature. Differentiation in terms of performance and throughput is usually achieved by using hardware-based techniques to either allocate dedicated resources for different QoS levels (i.e., physical partitioning) or by the over-provisioning of resources to absorb traffic fluctuations. The major drawbacks of these mechanisms are added complexity, higher cost, and inefficient resource utilization.

Supporting differential QoS for Web services should apply differentiation techniques at the SOAP layer rather than at lower protocol layers, such as HTTP or TCP. This allows content-based request classification while leveraging the metadata carried in the SOAP requests. Furthermore, differential QoS support should be transparent without requiring any changes to the application server, the service client, or the service implementation. Hence, unmodified commodity Web services platforms can be used without costly changes while avoiding specific hardware or OS prerequisites.

This section discusses a typical architecture that involves differential QoS support added to WSM platforms (see Figure 2). The following subsections describe key components of such an architecture.

### *Request Classifier*

A key requirement for supporting differentiated QoS is the ability to classify the incoming requests according to a classification schema. For example, a classifier component can use the requester attributes metadata, which is generated by the authentication handler, to map the request to a service class. SAML could play a key role in service classification because of its wide acceptance as a standard in identity management to encode the user's access rights, authentication information, and other user attributes such as the user's role and service level (e.g., gold, silver…).

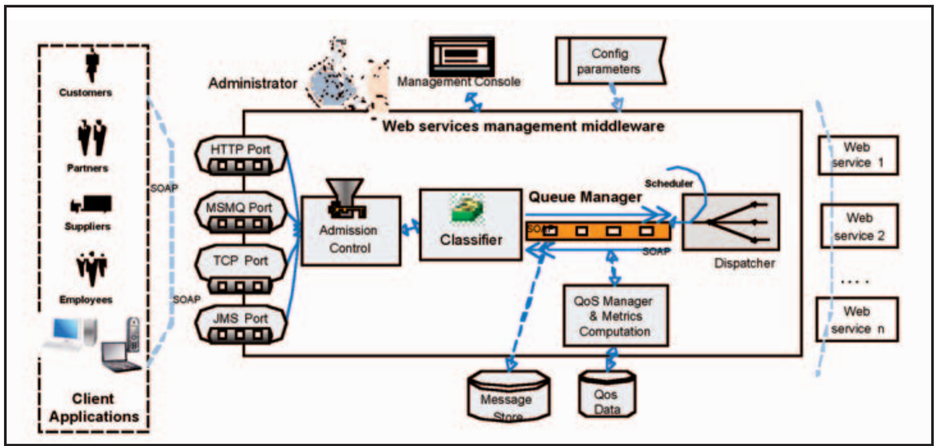


FIGURE 2 **Typical Web services management architecture for differential QoS support**

### *Admission Control*

Admission control is used to control throughput and to prevent the system from being overwhelmed in the presence of overload by keeping the maximum amount of load within the system capacity. Admission control requires two prerequisites: determining the load that a particular service call will generate on a system, and knowing the capacity of the service implementation. Admission control is also responsible for identifying appropriate scheduling algorithms to enable differentiation in service such as weighted round-robin scheduling.

### *Request Dispatcher*

There are various ways to implement the request dispatcher, particularly the queue manager component:

- The *look-ahead parameter approach* uses a single queue where requests are queued according to their arrival order. The scheduling algorithm can use a look-ahead parameter (k) that specifies the number of positions that will be searched from the head of the queue looking for requests of a given priority. If no request of the desired priority is found, the algorithm is repeated for the next lower level and so on. The higher the value of k, the better is the service given to higher priority requests. If k = 1, requests will be serviced according to their arrival order.
- *Multi-queue dispatcher with a queue manager* implements a set of logical FIFO queues, one for each service class. The scheduler then dispatches the pending requests according to the configured scheduling scheme.

## Summary

The notion of differentiated services is relevant for both business services and third-party Web and Grid services to meet the need for variety and flexibility. Web services that are capable of supporting different levels of service provide greater flexibility and achieve higher levels of reuse and adaptability. Advanced service-level management techniques that provide support for differentiated services are needed because the best-effort delivery model is not enough to satisfy various categories of client applications that access services in different contexts with varying service-level expectations. This article presented a generic and practical architecture to add differential QoS support to Web services management platforms; the remaining challenges were also discussed.

## References

- Erradi, A., Varadharajan, N., and Dr Padmanabhuni, S., "WS-DiffServ: A Middleware for Differentiated Service Delivery," submitted to the International Conference on Service-Oriented Computing (ICSOC 2005).
- Tosic, V. 2004, "Service Offerings for XML Web Services and Their Management Applications." PhD dissertation, Carleton University, Ottawa, Canada. http://elab.njit.edu/vladimir/TosicThesis-Final.pdf
- *IBM Web Service Level Agreements (WSLA):* www.research.ibm.com/wsla
- *Web Services Distributed Management (WSDM):* www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm

## 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.

■ ■ ■ aerradi@cse.unsw.edu.au
■ ■ ■ Akash_Das@infosys.com
■ ■ ■ Abhishek_Chatterjee@infosys.com
■ ■ ■ Anshuk_PalChaudhuri@infosys.com
■ ■ ■ Terance_Dias@infosys.com

## *WSJ* Advertiser Index

| Advertiser | URL | Phone | Page |
|---|---|---|---|
| Forum Systems | www.forumsys.com | 801-313-4400 | 3 |
| Gartner | gartner.com/us/adea | 800-778-1997 | 23 |
| IBM | www.ibm.com/middleware/flexible | | 5 |
| Intermedia.net | www.intermedia.com | 888-379-7729 | 6 |
| ISSJ | www.ISSJournal.com | 888-303-5282 | 46 |
| IT Solutions Guide | www.sys-con.com | 888-303-5252 | 35 |
| KaPow Technologies | www.kapow.com | 800-805-0823 | Cover III |
| Mindreef | www.mindreef.com/tryout | 603-465-2204 | 9 & 11 |
| Parasoft | www.parasoft.com/products | 888-305-0041 | Cover II |
| Skyway Software | www.skywaysoftware.com | 813-288-9355 | Cover IV |
| SYS-CON FREE Newsletters | www.sys-con.com | 201-802-3024 | 25 |

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.

# Myths About Service-Oriented Architecture

## Demystifying SOA

Service-oriented architecture (SOA) refers to an architectural solution that creates an environment in which services, service consumers, and service producers can coexist, and still have no dependence on each other. SOA enables an enterprise to increase the loose coupling and the reuse of frequently used software assets. These software assets, together with the functionality that they provide, are called services in the SOA terminology. By nature, SOAs are complex and are typically applied to solutions with highly volatile requirements.

WRITTEN BY
RAGHU ANANTHARANGACHAR

As SOA concepts are increasingly utilized and applied to solve software integration problems, there is confusion as to where SOAs are applicable, where they would yield the best results, and where the use of an SOA would be overkill. There are also misconceptions as to what the benefits of using an SOA are, as well as what the potential approaches to realizing an SOA solution are. In fact, there is no clear and consistent understanding about SOA across the different functions within an enterprise such as marketing, sales, delivery team, and so on. Similarly, even within the software industry these misconceptions are present.

This article expresses these misconceptions and areas of confusion in the form of myths and aims to provide a voice of reason and reality in response to these myths.

## SOA Solution Overview

An SOA solution refers to a solution built using the concepts of service-oriented architecture. As mentioned earlier, SOA is an architectural concept, and hence to realize an SOA solution, one must map the architecture to an implementation using a specific set of technologies/products/platforms. As with any other solution, an SOA solution is characterized by a set of mandatory (and optional) components. Figure 1 shows the main components of an SOA solution.

A complete SOA solution consists of the following main components:

- ***Producers:*** A producer is an entity that offers a specific service or functionality
- ***Consumers:*** A consumer is an entity that makes use of the service offered by the producer
- ***Services:*** A service is an entity that performs a specific task when invoked
- ***Contract:*** A contract or an interface specifies a format in which the data needs to be provided to the service to perform the specific task
- ***Repository:*** A repository is a glorified version of a registry and includes the metadata relevant for the solution, namely service, service contract, data/object model, and so on

### *Myth #1: SOA is a solution (panacea) to all software problems*

SOA is an architectural approach used to build solutions that are characterized by the presence of a set of services, service consumers, service producers, and service contracts. The approach of SOA needs to be used in light of business processes to arrive at a solution that can provide business benefits. Though SOA provides a sound architectural foundation to the overall solution, the specific problem regarding domain/business needs to be solved using the domain/business expertise, over and above the SOA solution.

As Figure 2 shows, the solution to the SOA consists of multiple components – an infrastructure layer, a set of applications, and a set of business processes and best practices. Though the expertise in the SOA architecture is the foundation for the overall solution, the other components, namely the business process expertise and the applications expertise, are absolutely required to arrive at an end-to-end solution to the problem at hand. The SOA principles are applied at each of the layers in the solution to arrive at a complete SOA solution. Only by doing so is it possible to reap the benefits of SOA completely.

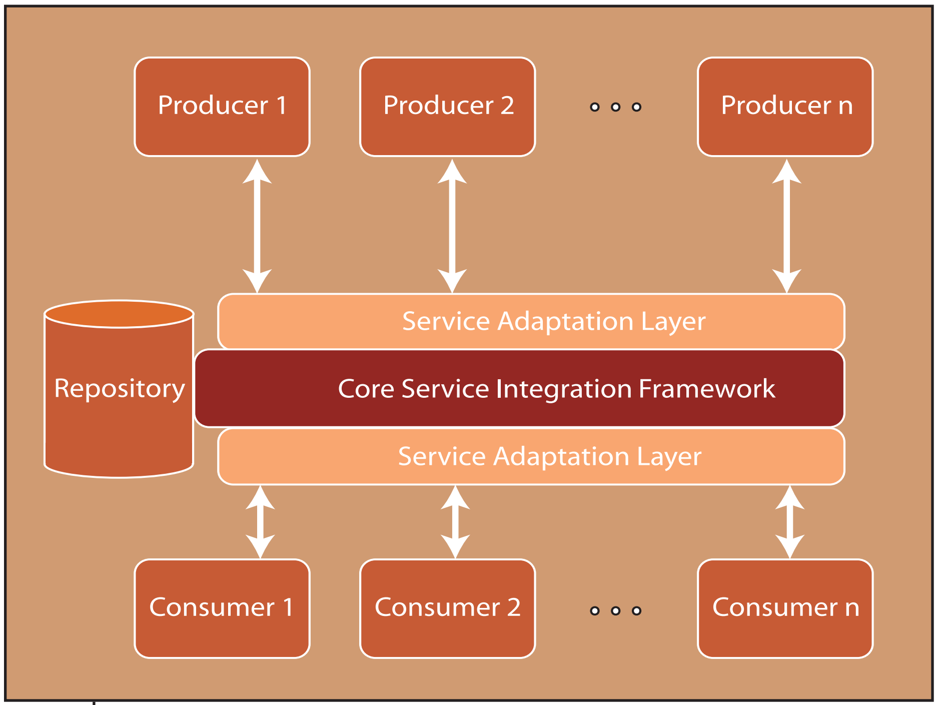


FIGURE 1 **SOA solution components**

***Myth #2: SOA is like a product, and can be downloaded for trial***

SOA is an architectural approach for building solutions that are loosely coupled in a stepwise, phased manner, resulting ultimately in the realization of a complex, federated, service-oriented enterprise. The business-specific services are initially identified over the SOA architecture, and then mapped to a set of technology-specific implementation architectures for the purpose of realization.

Though SOA concepts are reasonably simple to understand and apply, it is a rather involved process to build an SOA-rich enterprise, covering all aspects of the SOA characteristics. These solutions evolve over time, and they need to be crafted carefully and jointly along with the customer to ensure that the journey in the SOA architecture evolution is progressing in the "correct" direction. Due to this very nature, many SOA solutions typically do not fall into the category of "products."

***Myth #3: SOA is a complete, off-the-shelf solution***

SOA solutions are composed of prefabricated building blocks that typically represent the services identified during architecture workshops. The concept of prefabricated building blocks (services) reduces time to market, risk, promotes reusability, and provides a head start. While the generic, technology-neutral SOA approach continues to strive for increasing levels of reusability, the technology-specific SOA solution always requires some degree of customization. The degree of customization is based on factors including if the customer's environment already has a set of services or an environment with different degrees of legacy applications and integrations. The service-oriented approach provides a significant differentiation in building a federated service-oriented enterprise, and helps in realizing the business services reasonably quickly.

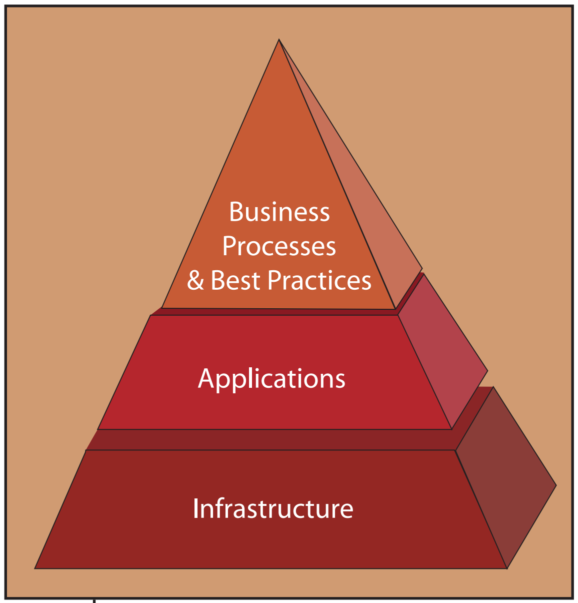


FIGURE 2 **SOA solution view**

***Myth #4: SOA software always needs to be developed using Web services***

SOA is a technology-neutral architecture, and can be realized using any technology. The selection of technology is performed by considering the various possible factors such as the functional requirements to be addressed, the performance and reliability requirements, the available budget, and so on. Based on these factors, the technology is chosen. Web services offer just one such

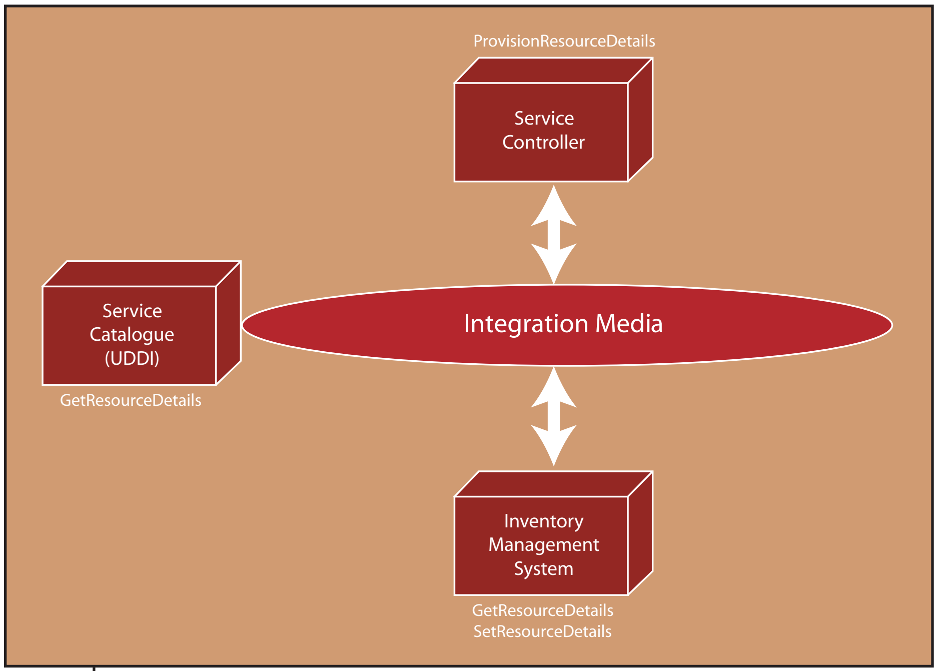


FIGURE 3 **Inventory management example**

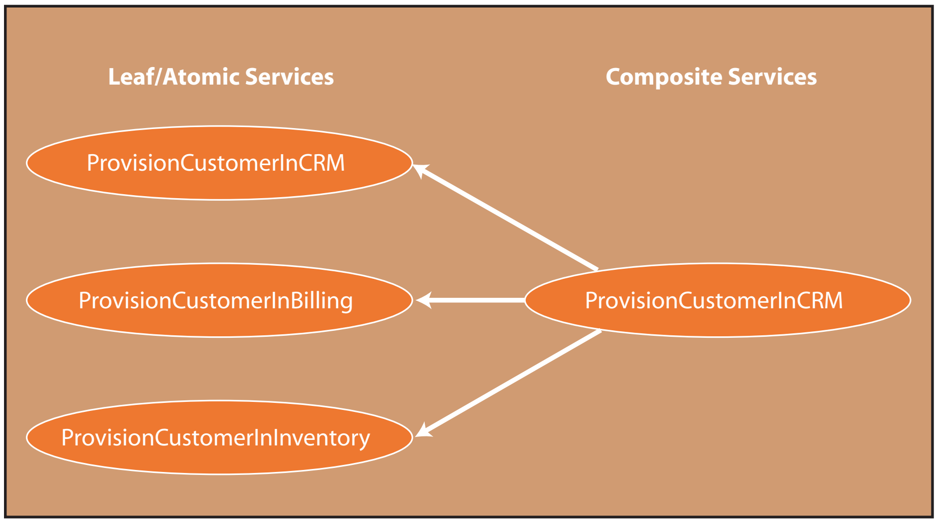


FIGURE 4 **Service types: atomic and composite**

technology option that is used to realize SOA solutions. However, it is possible to use other alternatives (apart from Web services), and still realize SOA solutions.

For example, consider Figure 3. In this case, the service controller could be a process engine that implements an end-to-end automated use case. The inventory management system provides services, namely "GetResourceDetails" and "SetResourceDetails." The details of the contract specification for each of these services are stored in the service catalogue. It is possible for the process running in the service controller to verify if a specific resource is available in the network inventory using the GetResourceDetails service, and then to set the status of the appropriate resource using SetResourceDetails. The service controller looks up the service details in the service catalogue, which could potentially be realized using UDDI. In this case, the service controller invokes the specific services by publishing events or messages on the integration media. It is possible to use a middleware (like TIBCO) for realizing the integration media, and SOAP/XML messages/events can be published to initiate the services.

### *Myth #5: Any software development using Web services is aligned with SOA*

Web services, coupled with the other relevant tools and technologies, offer one option that can be used to build and realize an SOA solution. However, a solution cannot be classified as SOA just by virtue of being built using Web services. A solution is compliant with SOA if it meets the following requirements:

- Interaction between service providers and consumers
- Usage of service contracts
- Usage of metadata

This can be compared to object-oriented architecture (OO). It is possible to use an OO language such as C++ and still end up in non-OO architecture, unless the necessary characteristics of an OO solution are addressed while building the solution.

### *Myth #6: Each service is always atomic in nature*

Services in the context of SOA represent the functionality provided by software assets. These services, when invoked, perform a specific task. At the lowest level, these services are mapped to a specific task. The services that always perform one atomic task are referred to as "leaf" services. The services that are created by the federation of other services are called as "composite" services. In other words, it is possible to define services in the SOA context, which are in turn composed of other SOA atomic services. Figure 4 shows that there are three atomic services (namely ProvisionCustomerInCRM, ProvisionCustomerinBilling, and ProvisionCustomerInInventory). Each of these services provisions the customer in the specified subsystem – for example, the ProvisionCustomerInCRM provisions the customer in the CRM (customer relationship management) subsystem, and so on. However, these atomic services are aggregated and a new composite service is created, called as ProvisionCustomer, which basically provisions the customer in all three subsystems.

***Myth #7: SOA is not aligned with any standards***

SOA is based on several industry-standard initiatives, namely the OASIS working group, the Web services standards bodies, and so on. Figure 5 shows the exact scenario of the associated standards body.

***Myth #8: SOA is the same as EAI***

There is a general misconception that SOA is the same as enterprise application integration (EAI). EAI is the integration approach in which various applications are integrated using a middleware, through the use of a set of connectors (or adapters). These adapters provide access to and exposure of all of the atomic interfaces of the underlying applications.

However, SOA is not the same as EAI. SOA is based on service aggregation that is based on functionality, and not on atomic APIs. SOA can be visualized as a further evolution of EAI, as shown in Figure 6.

SOA advocates integration based on services rather than on atomic APIs. SOA integration is similar to a richer form of ESB (enterprise service bus) integration, and represents a significant evolution from traditional EAI integration. Using SOA as an architectural approach results in significant improvement in the performance, flexibility, usability, and TCO (total cost of ownership) of the overall solution.

SOA is more sophisticated than "classical/traditional" EAI in several ways. First, SOA provides an aggregation capability (support for composite services) that is lacking in EAI. EAI deals with basic atomic APIs and data. Second, SOA provides support to work with service-level data, whereas EAI always deals with application integration using atomic API (application programming interface). Also, most important, SOA provides support for transformations and mappings, whereas EAI does not support these directly. Keeping all this in mind, it is possible to say that SOA is a more advanced architectural methodology.

***Myth #9: SOA is a very expensive solution***

SOA solutions are deployed in an evolutionary, stepwise manner that requires incremental investments. However, the framework allows for the consistency across the incremental solution.

The cost of the solution depends on several factors, among which the level of automation and the level of sophistication required in the solution are foremost. It is possible to arrive at a reasonable level of automation, and design and build an SOA solution that is cost-effective. Also, the cost depends on the choice of the other parameters such as the technology chosen, the products chosen (in case of green-field customers), and so on. All of the factors that contribute to the cost need to be considered

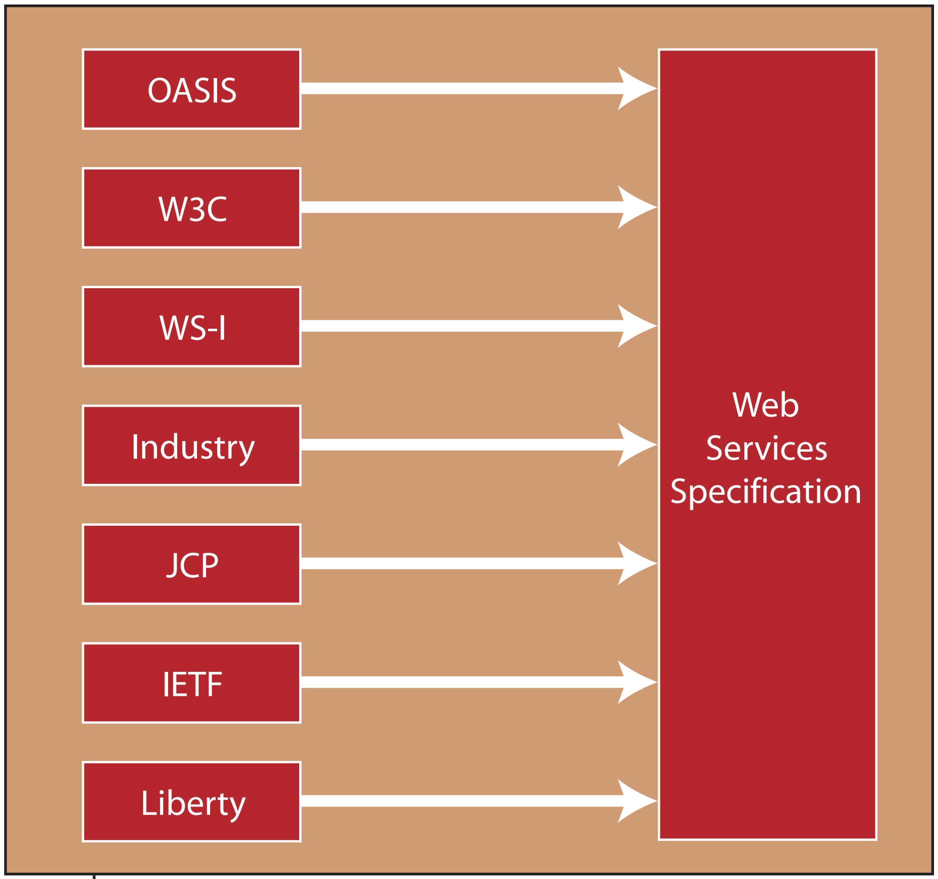


FIGURE 5 **Standards in SOA**

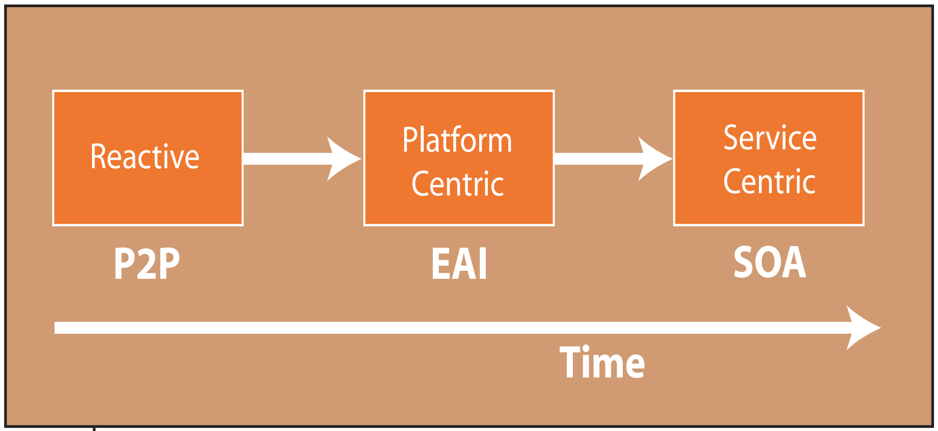


FIGURE 6 **Evolution of SOA**

carefully, and appropriate choices need to be made in order to reduce the cost. By doing so, it is possible to build a reasonably feature-rich and yet cheap solution. The enterprise architecture plays a crucial role in the SOA roadmap for the enterprise and precedes any major commitments. The concept of service and a means of interaction are more important than changing technologies overnight.

***Myth #10: SOA solution components (services, contracts, and data model) are completely reusable***

SOA strives for the highest possible amount of reuse, and the amount of reuse achievable increases over time.

In terms of the service, a large amount of reuse is possible in the technology-neutral representation. However, as the implementation is associated with the chosen technology, the reuse is limited if the technology is changed. However, when newer services are designed using existing services, a large amount of reuse is possible. In any case, the learning and the knowledge can definitely be reused, in addition to possible code reuse.

For example, the telecom customers would have their own legacy systems/applications in place, which would dictate the details about the services and the service contracts. It would be required in these cases to align the new definitions of the services and the service contracts with the already existing ones, and the flexibility to reuse something already existing reduces this drastically. In other words, only green-field customers can reuse the SOA components to a large extent.

In a legacy environment, it is required to "wrap" the legacy applications that provide the services, in appropriate SOA wrappers that can expose a service view of the existing legacy applications. It is also required to define appropriate service contracts to provide an indication of the details for service invocation.

***Myth #11: Services in the context of SOA are not explicit***

Services are autonomous, and they are explicit with specific boundaries defined for each. These services are built to perform specific tasks or to provide specific functionality. Services are discovered using service contracts, and are invoked using the information in the service contract. Each service has a specific life cycle, as shown in Figure 7.

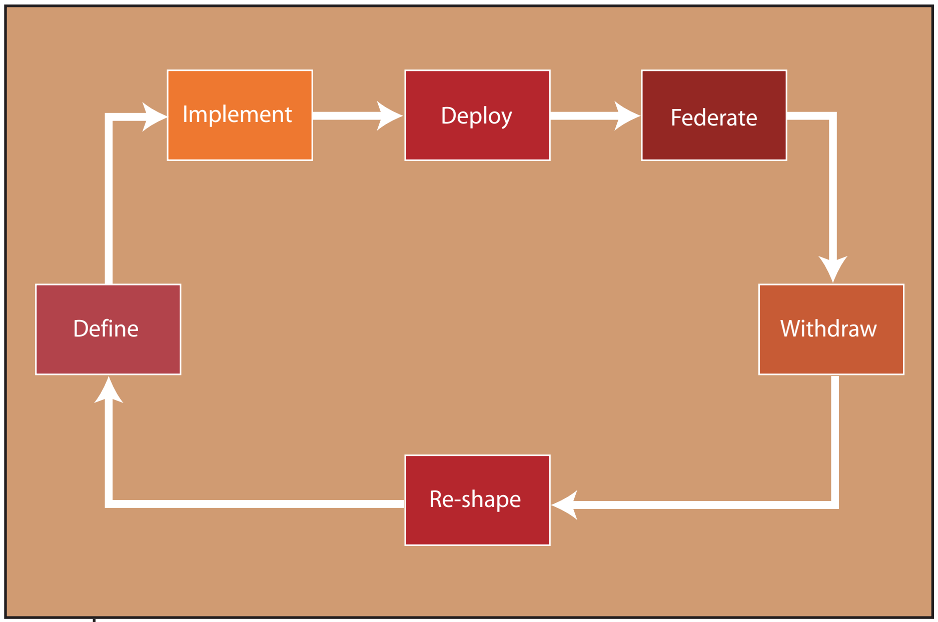


FIGURE 7 **SOA service life cycle**

Initially, the services are identified along with the customer. Once the services are identified, they need to be defined, built, and deployed. The federation of services is basically the collaboration between the various services that results in a composite service that provides an aggregated functionality. The services are used extensively once they are deployed. Subsequently, when the reuse level of the services drops or a newer service replaces an already existing service, then the service enters the withdraw phase. At this time, it is possible to reshape the service, which might result in the identification of a new service.

"Because it is not possible to sell a theory in isolation, SOA needs to be coupled with a set of practical examples and products before it can be successfully demonstrated to customers"

***Myth #12: SOA is applicable only to specific industrial domains such as Internet Data center***

SOA is an architectural concept, and does not have any dependency on any specific industrial segment. Though it has evolved in the context of an e-service–based industry or an e-commerce–based organization, it is equally applicable to any other industry. The concepts of an SOA can be applied to a telecom domain, financial services domain, and so on. For more details on HP's SOA approach, see http://h20219.www2.hp.com/services/cache/242475-0-0-225-121.html.

***Myth #13: SOA can be sold to customers as is***

While SOA is bound to help in the sales of several software products and best practices, it is preferable to tie it up with practical examples and case studies, enriched using the business processes. Because it is not possible to sell a theory in isolation, SOA needs to be coupled with a set of practical examples and products before it can be successfully demonstrated to customers.

"In fact, there is no clear and consistent understanding about SOA across the different functions within an enterprise such as marketing, sales, delivery team, and so on"

## Summary

In today's context, it is good to understand clearly what SOA can do and what it is. There are perceptions about SOA that are not always accurate. This article is an attempt to clarify those perceptions and put things in proper perspective. Once the concept of SOA is clear, it helps both the field teams to sell it appropriately and the delivery teams to set the expectations as to what exists and what can be reused.

**About the Author**
Raghu Anantharangachar is a solution architect with the Hewlett Packard Global Delivery India Centre, Bangalore. He has over 15 years of experience and has worked on porting, network management, and system integration projects in the past. He has a Bachelor's degree in Computer Science and Engineering from Bangalore University and a Master's degree in Industrial Management from the Indian Institute of Science, Bangalore.
raghu.anantharangachar@hp.com

# Subscribe Today!

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

FREE RESOURCE CD INCLUDED!
information STORAGE+SECURITY journal
www.ISSJournal.com
PREMIER 2004 VOL.1 ISSUE 1
In this issue:
- IP Storage
- SAN NAS
- Disaster Recovery
- Enterprise Security
- Infosecurity

Microsoft RSA Security
SecureID for Windows
SAN NAS
Emerging technology trends and market maneuvers
$9.99US $9.99CAN

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

## Editorial Mission

Greater Collaboration and Efficiency Through Education

✓ ***ISSJ*****'s editorial mission is to showcase proven solutions** that will guide, motivate, and inspire senior IT and business management leaders in the planning, development, deployment, and management of successful enterprise-wide security and storage solutions.

✓ ***ISSJ*** brings together all key elements of data storage and protection, and presents compelling insight into the benefits, efficiencies, and effectiveness gained by focusing on these two critical areas of IT simultaneously.

✓ ***ISSJ*** is an objective, critical source of information that helps storage and security managers make informed management decisions about what is working today, and what they need to plan for tomorrow, and is the only publication that focuses exclusively on the needs of IT professionals who are driving the enterprise storage architecture/infrastructure while pursuing and incorporating the latest security technologies.

✓ ***ISSJ*** achieves our mission by delivering in-depth features, practical "how-to" solutions, authoritative commentary, hard-hitting product reviews, comprehensive real-world case studies, and successful business models, written by and for professional IT storage and security practitioners.

**SAVE 50% OFF!**
(REGULAR NEWSSTAND PRICE)

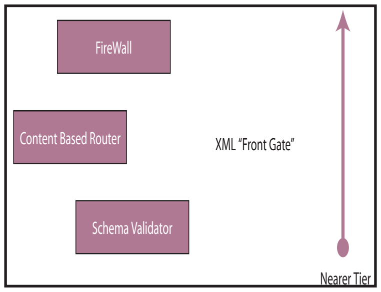
**Only $39.99** ONE YEAR 12 ISSUES

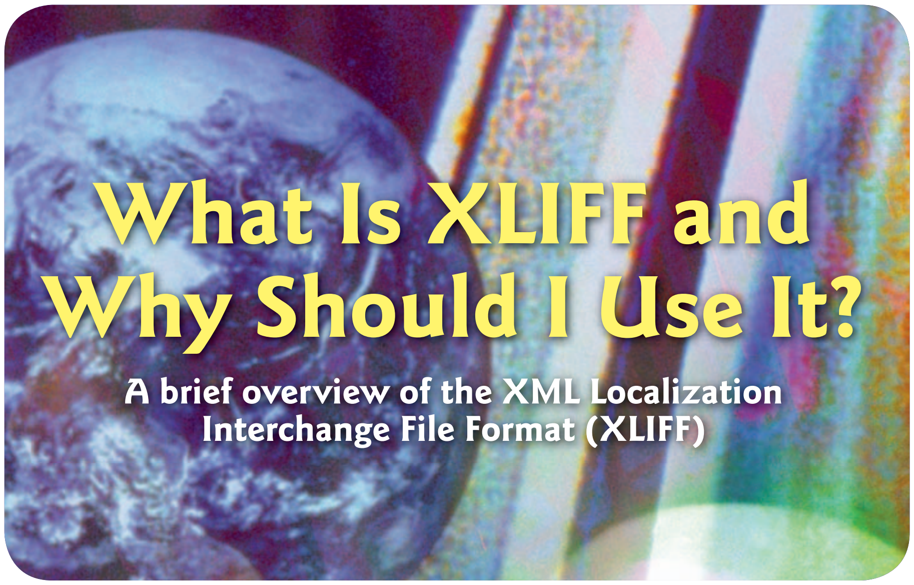
**www.ISSJournal.com**
or
**1-888-303-5282**

SYS-CON MEDIA
*The World's Leading i-Technology Publisher*

# XML JOURNAL SUPPLEMENT

## This Month




What Is XLIFF and Why Should I Use It?
A brief overview of the XML Localization Interchange File Format (XLIFF)

## 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

# What Is XLIFF and Why Should I Use It?

## A brief overview of the XML Localization Interchange File Format (XLIFF)

WRITTEN BY
**Tony Jewtushenko & Peter Reynolds**

**Much time, energy, and commitment are required to develop and sell successful software products and Web-based services. Most products of this type are initially developed for a specific language and locale (e.g., U.S. English). To maximize return on investment, products can be customized so they may be available to the largest possible market – the global market. This customization process is known as localization.**

Localization includes not only translation of the displayed text, but also adaptation of a product to comply with a country's cultural and legal practices.  Examples of cultural conventions include date/time formats, postal address formats, font sizes, appropriateness of colors, numeric or currency formats and symbols, culturally appropriate icons or graphics, etc. The diversity of software platforms and technologies means that tools and technologies that support localization are also diverse and are frequently incompatible with each other. Industry standards drive process and technology efficiencies, and OASIS XLIFF (XML Localization Interchange File Format) has emerged as a standard interchange file format for localization-related data and metadata. This article will introduce the process of localization and summarize the challenges and issues facing those who localize. It will illustrate how XLIFF addresses many of the challenges and issues with descriptions of its architecture, provide examples of how to use it in real life, and discuss how it was developed and where it goes from here.

## Localization: A Brief Overview

Having a product available in a different language increases the potential market into which it can be sold. Microsoft, IBM, and Oracle earn more than 60 percent of their sales from international markets. IBM recently reported that 70 percent of Web users speak a primary language other English. Within the United States, 18 percent of the population speak a language other than English at home. The Canadian Translation Bureau has reported that Canada represents 4 to 8 percent of the global translation services market with only 0.5 percent of the world's population.

The software localization industry was born in the mid 1980s as a result of the personal computer revolution. Early projects were very engineering intensive. Software developers would usually throw the final version of the application over the proverbial wall and expect the localization team to do the rest. The user interface and functional code were in the same file, which meant that functionality problems would often be introduced during the translation process. Before translation could begin, a software engineer would first need to change the character set code pages to ensure that software could be translated into the target language. Separating code from the UI required significant software development and localization resources. Physically and logically partitioning the UI data into separate resource containers reduced the severity and frequency of functionality defects introduced by the localization process.

The introduction of the Unicode character set standard and the growing emphasis on internationalization as a standard component of the software engineering practice was a very significant benefit to reducing the time and cost of localization. Internationalization is the process of developing software in such a way that it can run in different international environments without adapting or recompiling the code. Unicode introduced a common character set code page, and it is one of the corner-stones of software internationalization. The World Wide Web Consortium's (W3C) specifications and guidelines have addressed localization issues such as the separation of code and UI, character set issues, and date and time formatting. Cultural and economic factors are motivating many organizations to design and adapt their products for delivery to the global markets.

Today global markets are not content to wait months or even weeks for locally adapted software to be made available to them, so localization must now be done in parallel with the core product development. "Throwing software over the wall" to be localized after a product is sold is no longer an option.

The Internet introduces new challenges into the localization process by enabling more complex distribution models. Products and content are rolled out simultaneously throughout the global markets. The proliferation of diverse architectural frameworks and technologies upon which Internet and e-business applications are built has resulted in different degrees of complexity in both the core development and in localization.

Throughout its history, the localization industry addressed these business challenges through improvements in tools and processes delivered by competing vendors. Their solutions addressed specific technical or process requirements, but were rarely interoperable, which often resulted in vendor lock-in. With XLIFF, the localization industry got together to find a solution to some of the challenges that the industry is now facing.

### *Localization Challenges Addressed by XLIFF*

XLIFF has already provided solutions to difficult localization challenges, including the following.

**Challenge: Many different file formats to localize** – The typical localization project is composed of data stored within many unique resource formats. Building tools to support the localization of these formats is costly and inefficient.

**Solution:** Transform or extract all localizable resources, regardless of native format, into XLIFF containers. For example, before XLIFF existed, one major database and enterprise software vendor localized 32 unique resource formats. Some of the formats were proprietary, others were legacy, but most were industry standards (i.e., Java List Resource and Property Resource Bundles, Windows RC Data, .html, .jsp, and various XML). Additionally, each new release introduced additional resource formats, usually XML-based, but each required extensive retooling of the localization tools, which lead to quality problems and scheduling delays. To reduce the retooling work and its consequences, the core development teams were given the responsibility of extracting or transforming the new resource formats to XLIFF before handing off for translation. Three years and three major releases later, the number of unique resource formats supported by the localization tools was reduced from 32 to a much more manageable 17. Additionally, by adopting XLIFF, the tools development team was able to shift resources that had been dedicated to retooling onto building new cost-saving enhancements such as XLIFF's built-in suggested translations features. Listing 1 is an example of a Java properties file and its XLIFF representation.

**Challenge: Lack of version management metadata in native resource containers** – Localization projects often run concurrently with core development projects, which means that resource files will be made of up of multiple versions or milestones. Version management at the segment level (a segment is the smallest discrete unit of translatable text) is a very useful feature if your goal is to maximize the reuse of previous translations. Web content is often dynamic, and multilingual content must be kept in sync in order to maintain quality. Few if any native resource containers provide a means of tracking the version of the content.

**Solution:** XLIFF provides metadata structures for tracking versions of source and translated content.

**Challenge: Lack of workflow metadata in native resources** – During the localization process, data passes through many different hands. Data to be localized is typically externalized by software publishers and handed off to a localization service provider, who in turn may hand it over to translation subcontractors. At each stage of the process, the types of data required are unique to the particular phase (source/target text, Translation Memory, Machine Translation, Termbase, etc.). Introducing automation into a development process saves money and resources and improves quality by ensuring the reproducibility of the process. Native resource files don't usually contain mechanisms for tracking the stage of the process at which changes were introduced to the localization process.

**Solution:** XLIFF provides structures to track workflow phases.

**Challenge: Lack of standard resource or content container** – Developing tools to improve the localization process involves commitments in resources, testing, and deployment. Frequent retooling to keep up with modifications to proprietary resource formats will adversely impact schedules and quality levels of both the localization tool and the localized content. Developing a custom localization tool or acquiring an off-the-shelf tool that is compliant with a standard, localizable resources container such as XLIFF mitigates project risks and reduces potential vendor lock-in. The existing localization standards supported very specific types of localization related data. For example the Localization Industry Standards Association (LISA) OSCAR standards group defined TMX for Translation Memory Exchange, and TBX for Term Base Exchange, but these XML vocabularies aren't used to store resources to be translated. Another de facto standard, Opentag, an XML-based container for localizable content, provided inspiration to the founders of XLIFF, but it failed to provide the hierarchical structure required for describing groups of related resources such as dialog boxes and menus, and lacked project and workflow management metadata.

**Solution:** XLIFF provides a standard XML-based hierarchical container for localizable data and related metadata that is compatible with existing localization industry standards TMX and TBX.

The XLIFF Technical Committee (TC) succinctly summarizes the approach to addressing these localization challenges in its charter: "The purpose of the OASIS XLIFF TC is to define, through XML vocabularies, an extensible specification for the interchange of localization information. The specification will provide the ability to mark up and capture localizable data and interoperate with different processes or phases without loss of information. The vocabularies will be tool-neutral, support the localization-related aspects of internationalization, and the entire localization process. The vocabularies will support common software and content data formats. The specification will provide an extensibility mechanism to allow the development of tools compatible with an implementer's own proprietary data formats and workflow requirements."

## XLIFF Architecture

XLIFF is an XML vocabulary for exchanging localization related data. This standard was designed to take full advantage of XML's technical features, yet fit comfortably into the typical localization process. The architecture of XLIFF is designed so that it can be used in a number of ways:

- Extract only the localizable content within the original files to XLIFF, leaving untranslatable content in a skeleton file
- Transform the entire resource file to an XLIFF file

### *The Extract-Localize-Merge Model*

The original files to be localized are read by a filter that extracts localizable data into the XLIFF file, and generates a skeleton file that contains only nonlocalizable data and placeholders for where the localized data will be written. The skeleton file can be either externally referenced or embedded directly into the related XLIFF file. Upon completion of the translation process, XLIFF is merged with the nonlocalization data in the skeleton file to create the localized file.

### XSL Transformation

Today XML is more widely adopted, so XSL can be used to transform an XML file directly into an XLIFF container. This process is illustrated in Figure 2.

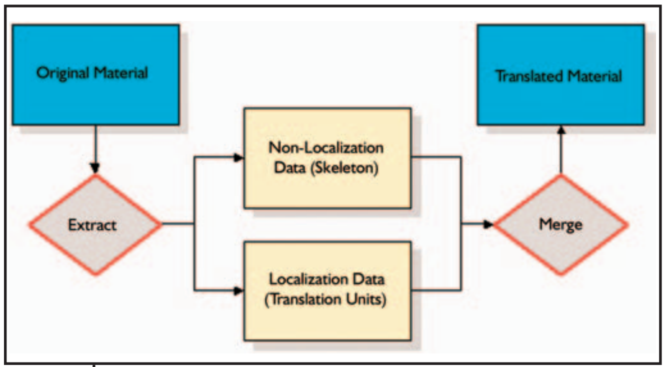

Original Material
Translated Material
Non-Localization Data (Skeleton)
Extract
Merge
Localization Data (Translation Units)

FIGURE 1 The extract-localize-merge process

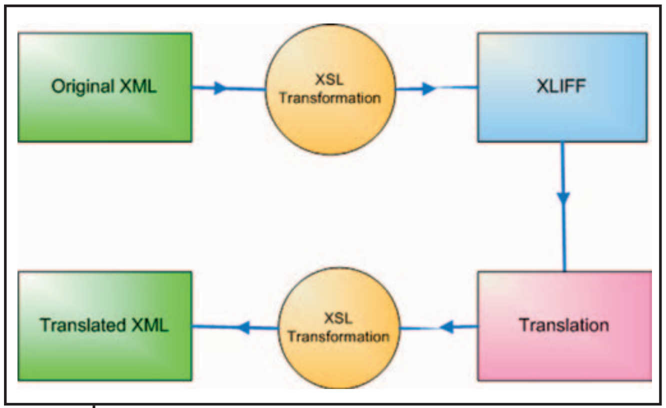


FIGURE 2 XSL transformation

In this model, an entire XML document is transformed into XLIFF, translated, and transformed back to its original structure. An open source tool is available for transforming XML into XLIFF and back again, and may be downloaded from https://sourceforge.net/projects/xliffroundtrip/.

### Structure of XLIFF

Listing 2 is an example of a simple XLIFF file. This document does not intend to describe in detail all of the components that make up an XLIFF file. For further information on this, refer to the XLIFF specification from OASIS. The current specification is available at www.oasis-open.org/committees/xliff/documents/cs-xliff-core-1.1-20031031.htm.

XLIFF files are bilingual. There is only one source language and one target language. This makes the overall model much simpler when you consider all of the additional features the format provides, such as pretranslation, version tracking, etc. It also avoids the problem that a multilingual file would have where different versions are localized by different translators at the same time.

*File Element*

Each *file* element corresponds to a real or virtual file from the source material. By virtual file, I mean that it could correspond to part of a database. An XLIFF file must contain at least one but can contain any number of file elements. It would be possible for a single XLIFF document to contain all of the resource files for an entire software application, each corresponding to its individual file element. This approach should take into consideration the effect that having many file elements will have on the size of the XLIFF file and management of the project.

The file element has a number of attributes that contain data that describes the actual file to which they relate.

Each file element contains both *header* and *body* child elements. The header element contains metadata about the file being localized. The body element contains the data to be localized.

*trans-unit*

The *trans-unit* contains the data to be localized. Each *trans-unit* has a *source* element that contains the original data to be localized. After translation, the *target* element contains the data that has been localized. The status of the target can be tracked with the *state* attribute.

```
<trans-unit>
 <source xml:lang="en-US">We are testing XLIFF.</source>
 <target state="translated" xml:lang="fr">Nous testons XLIFF.</
target>
</trans-unit>
```

The *source* and *target* elements can include specific in-line elements for specific purposes. The *g* element, for example is useful for tagging text mark-up such font face changes or bold or italic.

```
<source xml:lang='EN-US'>We are testing <g id='b1'
ctype='bold'>XLIFF</g></source>
```

The *x* element is useful for empty in-line elements, like images.

```
<source xml:lang='EN-US'>We detected spaceships on the planet <x
id='i1' ctype='image' img:src='enterprize.jpg' /></source>
```

The attributes of the *trans-unit* element can be used to store data about the resource being translated. Note that the *id* tag in a trans-unit identifies the trans-unit relative to the XLIFF file, and does not represent the native resource id. The *resname* tag stores the native resource id.

The XLIFF sample code below represents a cancel button. Using this information, a localization resource editor can render the button or windows resource, which can then be visually represented in a WYSIWYG editor.

```
<trans-unit id='1' resname='IDCANCEL' restype='button'
coord='8;8;50;14'>
     <source xml:lang='en'>Cancel</source>
</trans-unit>
```

Logically or structurally related *trans-unit* elements can be grouped together using the *group* element. An example of where this is useful is representing a dialog box as a group of trans-units that contain its individual components.

*bin-unit*

The *bin-unit* is used to store nontextual data such as icons, cursors, or images. The objects can be stored as a reference to an external file or embedded internally within the XLIFF document itself. When stored as an external file reference, the bin-unit contains the address where the file is found. The binary file can also be embedded within the bin-unit as base-64 encoded data.

If there is localizable text associated with the binary object, it can be stored in one or more *trans-unit* elements contained within the bin-unit.

*context-group/context*

The *context-group* element and its child *context* provide a method for storing resource metadata related to the trans-unit. This structure keeps track of vital information that may be used to rebuild the content into its native format, or to identify resource-related content such as custom control class names or related properties.

Listing 3 shows how to represent PO (portable object) files when the translation unit occurs in multiple positions within a file. In the example, we use context-group elements to represent each individual position.

## Extensibility

One of the major initiatives within XLIFF 1.1 is the use of XML schema namespaces to allow XLIFF to be embedded within another XML document or for XLIFF to be extended with another namespace. A company might have an XML application that meets their needs in all areas except localization. A small section of XLIFF could be embedded within this file to deal with that. It is also possible to embed a section of another XML vocabulary at particular extension points within an XML document.

Listing 4 shows an example of how an XLIFF document could be extended. The "mysc" elements are embedded within the XLIFF document and are identified by its namespace.

## Alternate Translations

The *alt-trans* element is used to store alternative translations. This could be Translation Memory, Machine Translation, a previous translation, or a previous translation in a different dialect than the current target language. An example of where you would store a different dialect would be where you have translated an English document to French Canadian and you previously have a French translation. The XLIFF file would have English as the source language and French Canadian as the target language, but you can provide the French translation as the alternative translation in the *alt-trans* element.

The example below illustrates a previous translation of similar text is available and the alt-trans element, with this translation, is placed within the trans-unit.

```
<trans-unit>
        <source xml:lang="en-US">We are testing XLIFF.</source>
        <target state=”needs-review-translation” xml:lang="fr-
CA>Nous testons XLIFF.</target>
        <alt-trans>
              <target xml:lang="fr-FR">Nous testons.</target>
        <alt-trans>
</trans-unit>
```

## Version Control

Within an XLIFF document, it is possible to keep track of what is happening with the use of the *phase* element. This can be used for version control where different versions of the translation are stored in the *alt-trans* element, or it can used to provide other control data.

The *phase* element is used inside the header element and this provides the details of a particular phase. Information such as what was done to a segment with a particular tool can be tracked here.

```
<header>
 <phase-group>
  <phase phase-name='trans' process-name='translation' tool='Trados'
date='2005-04-01T20:01:23Z'/>
  <phase phase-name='edit' process-name='editing' tool='Trados'
date='2005-04-01T20:01:23Z'/>
 </phase-group>
</header>
```

A target element can be associated with a particular phase. In the example below, we have used the phase element to show that the target element introduced during the phase labelled “trans.” Details of this phase would be specified in the header section.

```
<trans-unit>
        <source xml:lang="en-US">We are testing XLIFF.</source>
        <target xml:lang="fr">Nous testons XLIFF.</target>
        <alt-trans>
              <source xml:lang="fr">We are testing.</source>
              <target xml:lang="fr" phase-name='trans'>Nous tes-
tons.</target>
        <alt-trans>
</trans-unit>
```

## XLIFF in Action with Web Services

This solution was implemented for a customer of Bowne Global Solutions (BGS) that manufactures automobiles. They had a significant Web presence but wanted to expand their Web site to be more global by providing the content in a range of languages. There was a complicated process involved in the localization of their existing Web site. They were concerned that this process was adding extra cost without the corresponding value.

Together with this customer, BGS designed a solution that made

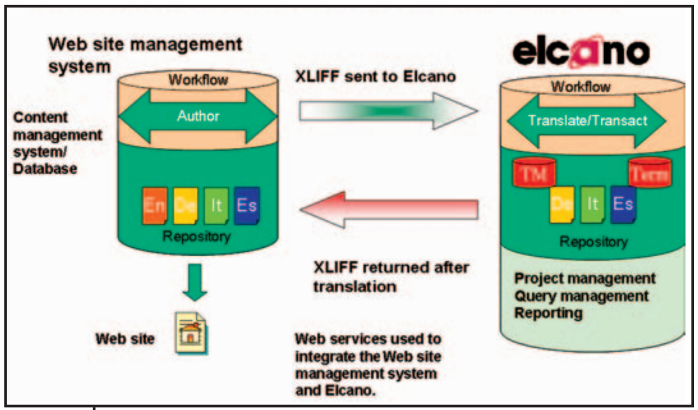

Web site management system
Workflow
Author
Content management system/ Database
En De It Es
Repository
Web site
XLIFF sent to Elcano
XLIFF returned after translation
Web services used to integrate the Web site management system and Elcano.
elcano
Workflow
Translate/Transact
TM
Term
De It Es
Repository
Project management
Query management
Reporting

FIGURE 3 **Elcano Web services overview**

innovative use of XLIFF, Web services, and BGS's online translation service, Elcano. Figure 3 shows the system that was built in order to achieve this. The repository on the customer's side was a Content Management System (CMS).

Within the database for the CMS, the content was stored as segments of HTML. The Web pages were generated by assigning the HTML segments to placeholders on the page. Listing 5 shows an example of an HTML segment stored within the CMS.

We made use of in-line tags to preserve the formatting information from the HTML segment and to ensure that the file created was a valid, well-formed XLIFF file (see Listing 6).

The XLIFF files that were transported to and from the CMS were translated using a software translation tool. These tools use Translation Memory technology in order to reuse previous translations. The editor

"Today global markets are not content to wait months or even weeks for locally adapted software to be made available to them, so localization must now be done in parallel with the core product development"

is designed so a translator can only edit those parts of the XLIFF that relate to translations. What was striking during this project was how stable XLIFF was as a localization format. At that time, we needed to localize problematic HTML files. Using XLIFF assured the technical quality of the end delivery.

## A Brief History of XLIFF

XLIFF got its start as an informal group of like-minded localization and globalization professionals based in Dublin, Ireland who recognized that the lack of a standard localizable resource container added significant tools, development, and project management costs to the localization process. Present at the kickoff meeting were Novell, Oracle, Sun, and Berlitz (now Bowne Global Solutions). The group shortly thereafter welcomed Lotus/IBM, Moravia IT, RWS Group, and Lionbridge. Initial discussions did not define the core technology for the format, and the group was open to any ideas, including flat or binary data files. Once preliminary research was completed, the group concurred that XML solved our technology requirements and chose it for the core technology.

Our schedule was aggressive, and the 1.0 draft specification was completed within six months. Once the core deliverables were completed, the group began preparing the specification for public distribution throughout the industry for comment. We recognized that we lacked the organizational and intellectual property framework and procedural expertise that was needed to support the advancement of our work. So in the autumn of 2001, the XLIFF group commenced an investigation of consortia as potential homes for our work. Specifically, we researched OASIS, W3C, and LISA, and selected OASIS because of its XML focus, and because it provided us with a turnkey infrastructure within which to operate. A secondary but important factor was that many of our member companies were already members of OASIS. Over the life of the TC, corporate participation has included: Bowne Global Solutions, Convey Software, HP, IBM, Lionbridge, LRC, Microsoft, Moravia IT, Novell, Oracle, RWS Group/Enlaso, SAP, SDL International, Sun Microsystems, TRADOS, and Xerox. The educational institution Localisation Research Centre of the University of Limerick is also a member of the TC. Individual members have included Doug Domeny, Paul Gampe, Enda McDonnell, Florian Sachse, Bryan Schnabel, Shigemichi Yazawa, and Andrzej Zydron.

The XLIFF 1.0 Specification was submitted to the Technical Committee as the starting point for its work, and was reviewed and approved by the TC on April 15, 2002 as an OASIS Committee Specification. The TC decided not to submit the 1.0 specification as an official OASIS standard, choosing instead to focus on completing revision 1.1 to deliver key improvements, including customization and extensibility using private namespaces, embedded XLIFF, and improved data validation via XSD rather than 1.0's DTD. The XLIFF 1.1 Specification was approved as an OASIS committee specification on October 31, 2003.

The TC is currently authoring a number of profile specifications to help tools developers and implementers implement XLIFF 1.1 consistently. The HTML Profile has been completed and will be made available publicly, and the final draft of the PO/POT Files Profile shall be made public shortly. Profiles are being authored for Java Resource Bundles and other proprietary resources formats, and public drafts will be published within the next quarter. The TC is also finalizing guidelines for XLIFF's support of segmentation – the process of optimizing the reuse of previously translated discrete logical or linguistic units of translatable content. After the profiles and segmentation guidelines are published and publicly reviewed, the TC plans to submit the XLIFF 1.1 spec and related profiles to the OASIS standards review process for consideration as an official OASIS standard.

## Conclusion

XLIFF is an emerging standard that provides a structured interchange format for localization data. It is increasingly used as a mechanism by many large companies to optimize enterprise localization workflow processes. The open source community is also developing tools and processes that use XLIFF. The XLIFF TC at OASIS is continuing to work on improving the standard. We welcome those with an interest and expertise in this area to either join and actively participate in the TC or to comment on our work.

## References

- *XLIFF TC Homepage:* www.oasis-open.org/committees/tc_home.php?wg_abbrev=xliff
- *Comments Submission Form:* www.oasis-open.org/committees/comments/form.php?wg_abbrev=xliff
- *XLIFF 1.1 Committee Specification:* www.oasis-open.org/committees/xliff/documents/cs-xliff-core-1.1-20031031.htm
- *XLIFF 1.1 Schema:* www.oasis-open.org/committees/xliff/documents/xliff-core-1.1.xsd
- *XLIFF TC Whitepaper:* www.oasis-open.org/apps/group_public/download.php/3110/XLIFF-core-whitepaper_1.1-cs.pdf
- *XLIFF TC Frequently Asked Questions (FAQ):* www.oasis-open.org/

committees/xliff/faq.php
- *XLIFF – The XML-based Open Standard for Localisable Content (PowerPoint):* www.localisation.ie/publications/presentations/2004/Conference/Presentations/TonyJewtushenko.ppt
- *IBM Developer Works: "XML in localisation: Use XLIFF to translate documents":* www-128.ibm.com/developerworks/library-combined/x-localis2/
- *Sun's Java.Net open-language-tools Project:* https://open-language-tools.dev.java.net/
- *Bowne Global Solutions Whitepaper "An introduction to XLIFF":* www.bowneglobal.com/english/media/WP_XLIFF_US_0505.pdf
- *XLIFF Tools Open Source Project:* http://xliff-tools.freedesktop.org/wiki/
- *XLIFF Support in KBabel:* http://developer.kde.org/summerofcode/xliff.html
- *OASIS:* www.oasis-open.org
- *Translation Web Services Technical Committee:* www.oasis-open.org/committees/tc_home.php?wg_abbrev=trans-ws
- *Localisation Research Centre, University of Limerick:* www.localisation.ie/
- *Opentag:* www.opentag.com/xliff.htm
- *Localisation Industry Standards Association:* www.lisa.org/
- *W3C i18n Activity:* www.w3.org/International/

## Author Bio

Tony Jewtushenko is the founding and present chair of the XLIFF TC as well as the director of R&D for Product Innovator (www.productinnovator.com), a Dublin, Ireland-based consultancy that provides product management, process improvement, localization, and internationalization services to software companies. During his 23-year career, Tony was a key contributor to dozens of releases of successful commercial software products, including Lotus 1-2-3 and Notes, Oracle JDeveloper, and iDS. His multinational work and life experience spans USA, Europe, Middle East, and Asia.

Peter Reynolds is manager of the software development team at the Dublin, Ireland office of Bowne Global Solutions (BGS), the leading provider of localization and translation solution. Peter and his team are responsible for developing some of the software that BGS use to run their business, including Elcano, the online translation service, and myInfoShare, the online project collaboration workspace. Peter has been working on XLIFF since its inception and is a founder member and secretary of the OASIS XLIFF Technical Committee. He also chairs the OASIS Translation Web Services Technical Committee.

**tony.jewtushenko**@productinnovator.com

**peter.reynolds**@bowneglobal.com

**Listing 1**

```
MessageBundle_en_US.properties file:
template = At {2,time,short} on {2,date,long}, we detected
\
{1,number,integer} spaceships on the planet {0}.
planet = Mars

MessageBundle_en_US.properties.xlf

<?xml version="1.0"?>
<xliff version='1.1'
xmlns='urn:oasis:names:tc:xliff:document:1.1'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:schemaLocation='urn:oasis:names:tc:xliff:document:1.1
xliff-core-1.1.xsd'>
  <file source-language="en_US" datatype="javapropertyreso
urcebundle"
   original="MessageBundle_en_US.properties">
    <body>
      <trans-unit id="1" restype="string"
resname="template" >
      <source>At <ph id="1">{2,time,short}</ph> on <ph
id="2">{2,date,long}</ph>, we detected <ph id="3">{1,numbe
r,integer}</ph> spaceships on the planet <ph id="4">{0}</
ph>.</source>
      </trans-unit>
      <trans-unit id="1" restype="string" resname="planet"
>
        <source>Mars</source>
      </trans-unit>
    </body>
  </file>
</xliff>
```

**Listing 2**

```
<?xml version="1.0" encoding="utf-8"?>
<xliff version="1.1">
<file source-language='en' target-language='fr'
datatype="database" original="LR GWS">
<header>
      <count-group>
        <count count-type="word">11</count>
      </count-group>
</header>
<body>
<trans-unit>
        <source xml:lang="en-US">We are testing XLIFF.</
source>
        <target xml:lang="en">Nous testons XLIFF.</target>
</trans-unit>
<trans-unit>
        <source xml:lang="en-US">The translation portal.</
source>
        <target xml:lang="fr">Le portal de traduction.</
target>
</trans-unit>
</body>
</file>
</xliff>
```

**Listing 3**

```
A sample PO file:
#: example.cpp:134 example.cpp:343 example2.cpp:23
msgid untranslated-string
msgstr translated-string

is represented by an XLIFF as follows:
<trans-unit id="##1">
   <source xml:space="preserve">untranslated-string</
source>
   <target xml:space="preserve">translated-string</target>
   <context-group name="x-po-reference#1"
purpose="location">
      <context context-type="sourcefile">example.cpp</con-
text>
      <context context-type="linenumber">134</context>
   </context-group>
   <context-group name="x-po-reference#2"
purpose="location">
      <context context-type="sourcefile">example.cpp</con-
text>
      <context context-type="linenumber">343</context>
   </context-group>
   <context-group name="x-po-reference#3"
purpose="location">
      <context context-type="sourcefile">example2.cpp</con-
text>
      <context context-type=»linenumber»>23</context>
   </context-group>
</trans-unit>
```

To view Listings 4-6, please view this article online at http://webservices.sys-con.com/read/issue/archives/ (Vol: 5 Iss: 8).

WRITTEN BY **TOM YOHE**

# Caching for XML Performance

## Delegating optimization throughout your Web services infrastructure

**Highly scalable implementations of service-oriented architectures (SOAs) always include heavy doses of caching. A guided tour through the SOA tiers, describing the caching and XML acceleration techniques employed along the way, provides the SOA enterprise architect with an awareness of optimization possibilities applicable to a Web service infrastructure. Consolidating the acceleration functions in an integrated appliance and controlling them via policies specified by WSDL annotations simplifies the implementation.**

**AUTHOR BIO**

*Tom Yohe is the VP of Engineering and co-founder of Stampede Technologies, and currently leads one of the world's most elite enterprise optimization engineering teams, developing Web Acceleration Appliances for a broad range of Fortune 500 companies. Tom has been delivering award-winning enterprise products for over 25 years and has been granted numerous patents for unique data communications optimization techniques. Tom has a computer science degree from Penn State University.*

Caching is a fundamental optimization technique found in all computer systems and networks. Modern CPU chips maintain instruction and data caches along with clever methods of ensuring coherency with the RAM that may be simultaneously accessed by other CPUs and I/O processors. In many respects, RAM can be thought of as a cache of your virtual memory swap file. Our networked world is full of caching subsystems, each designed to minimize the latency required to access correct content. As the computing paradigm has evolved to multi-tiered SOAs, caching continues to play a crucial role.

A typical SOA solution is implemented in a multitiered fashion for many reasons, not the least of which is performance. Figure 1 illustrates the "nearest tier" through the "farthest" tier – the nearest tier is closest to the Web service consumer that issued the request; the farthest tier is the central database engine.

The computing devices and software that drive the farther tiers are generally more expensive compared with the subsystems in the nearer tiers. The processing that takes place in the farther tier subsystems tends to be "business-logic," is often executed in higher-level interpretive languages, and generally takes more time and resources to execute. The logic of subsystems at the nearer tiers is usually very system software–oriented, is executed in the native machine language if not in ASICs, and tends to be highly efficient. The subsystems in all tiers are usually designed for high scalability, where simply adding more like subsystems can expand capacity. In front of each subsystem, tier load balancers are employed to intelligently deliver object requests and route responses between the tiers. The message communication between each tier requires TCP/IP routing, load balancing, parsing, and other processing, all of which adds further latency to the transaction response time. Therefore, the more work that can be completed in the nearer tiers, the more efficient and responsive the overall SOA implementation will be.

### A Caching Tour of the Tiers

Caching begins at the farthest Enterprise Information System (EIS) tier and proceeds through the nearest tier. XML caching takes place at the nearer tiers.

### Caching at the Core

In the farthest EIS tier, most database engines such as DB2, MS SQL Server, and Oracle have a myriad of caching options designed to minimize disk I/O operations as much as possible while ensuring that transactions are always physically recorded. The guidelines for these products usually suggest that the database files should be stored on a file system where the caching services supplied by the native OS's file system are disabled. The rationale for this is that the caching services of the native OS can't possibly be as intelligent as what is performed by the database, so therefore any memory resources devoted to the native OS caching engine would be wasted. Furthermore, the database engines want to guarantee that updates are physically written to storage. Database engines also make extensive use of "SQL Statement Caching," which reduces the costly parse processing on the database server. While these database-caching techniques may be very exciting for system and database administrators, they are likely only of passing interest to the SOA enterprise architect.

### Moving Nearer Outward

As we move outward to the application servers or business tier, the application designer often carries the burden of integrating caching optimization. In this tier we find two basic flavors of server platforms, J2EE and Microsoft ASP.NET. Each of the two environments offers the application developer a means for injecting caching to optimize performance and increase scalability. Some of the caching techniques employed in the business tier directly relate to Web services, while others optimize the business logic implementation of the Web service. The SOA enterprise architect should have a full understanding of all caching techniques utilized in this tier.

In the business tier, there are two important caching objectives: storing business data in memory to be readily accessed by application servers, and controlling caching strategies by setting related HTTP headers through platform APIs.

## J2EE Caching Techniques

In the J2EE environment, the caching of business data is achieved by various Object-Relational mapping technologies. The dominant choice offered by J2EE vendors is the Enterprise Java Bean (EJB). The Entity Bean developer often implements transactional caching of dynamic data by adhering to the Container Managed Persistence (CMP) model that provides caching as an inherent attribute. Some J2EE platforms such as IBM WebSphere offer extensions to CMP that allow data caching of static items across transactions. This extended functionality can be easily integrated into an application by setting attributes in property sheets. EJB can be a bit heavy-handed targeting large, complex systems. Some "lighter" technologies include both standards-based model abstraction such as Java Data Objects (JDO) and open-source offerings such as Hibernate.

The "JCache" specification defines a powerful set of Java-caching APIs that support implementations of caching frameworks. Both open-source and commercial offerings of caching frameworks allow cached objects to be shared among multiple applications. These caching frameworks are often an important component in enterprises where "single sign-on" to multiple applications is required. JCache provides methods to control temporary storage and expiration of Java objects in common memory in a manner that ensures that the cached objects are coherent with the data stored in the farther tiers.

For controlling caching-employed HTTP servers and proxies, the J2EE specification defines the *HttpServletResponse* class, a component of the servlet container, as a fundamental J2EE building block. This class provides control of HTTP-specific functionality by furnishing methods to access and generate HTTP headers and cookies. Specifically, the *addHeader* method allows the HTTP 1.1 "cache-control" and "expiration" directives to be emitted to facilitate caching in the nearer tiers.

## The Microsoft Way

Instead of mapping business data into a constellation of objects, Microsoft employs a different strategy of caching transactional data. Data elements are presented as relational tables in memory, which is very similar to how they are persisted on disk. Those in-memory tables can be accessed via the *rowset* API, and maintained via either *DataReader* or *DataSet* APIs by ADO.NET.

ASP.NET developers have at least three types of caching capabilities available to them. "Output caching" informs ASP.NET that responses built for a page can be returned on subsequent requests for the same page without re-executing the ASPX script. Either an entire page can be added to the output cache or to a specified ASP.NET user control. Although the ASP.NET designer can embed an *HttpCachePolicy* class into their objects so that standard HTTP caching proxies can interpret HTTP headers, this feature is of limited value because most standard HTTP proxy servers are not designed to deal with responses associated with HTTP "POST" commands. Another powerful caching method of ASP.NET is "application caching," which allows the application to use the cache to store any data by leveraging the *System.Web.Caching.Cache* class.

## Web Servers and HTTP Caching Proxies

Much of the Web services infrastructure is dependent on a highly scalable Web server tier. Microsoft's .NET framework for Web services depends largely upon the Web services functionality that is built into "Internet Information Server" (IIS) and its accompanying Microsoft Internet Security and Acceleration Server (ISA). J2EE-oriented infrastructures are more likely to leverage the J2EE-based Web/application servers, such as Apache Tomcat, embedded with the Axis SOAP handlers, as a common critical component. In an SOA implementation, caching proxy servers perform front-end processing offload and can sometimes provide layer-7 load balancing to the Web servers.

Traditional Web-caching proxy servers were designed to optimize the viewing of HTML Web pages by a browser. With HTML, even individualized page views share a similar layout, thus a large number of common components and many page views will be

"Although the ASP.NET designer can embed an HttpCachePolicy class into their objects so that standard HTTP caching proxies can interpret HTTP headers, this feature is of limited value because most standard HTTP proxy servers are not designed to deal with responses associated with HTTP 'POST' commands" absolutely identical. This led to the establishment of a well-defined set of caching rules spelled out in the HTTP 1.1 specification for caching frequently requested pages and page components.

In an SOA, though the need for caching might be even greater, the means for caching in the Web tier is less defined. Unlike HTML, XML supports many different types of messages determined by the set of applications supported by the enterprise network. XML messages are most often transactional in nature – reflecting the dynamic nature of a conversation, negotiation, or exchange between parties. With XML traffic, there is less use of the common components that can be effectively cached. One of the caching rules set forth in the aforementioned HTTP 1.1 specification is that responses to HTTP POST messages are not cacheable, unless the response includes appropriate "Cache-Control" or "Expires" header fields. Unfortunately, most SOAP toolkits are designed to embed SOAP requests inside an HTTP POST message. Therefore, since traditional HTTP proxy servers make the assumption that responses to an HTTP POST are not cacheable, they provide no benefit for an SOA implementation.

An intelligent, "SOAP-aware" Web-caching tier can provide many important optimizations to an SOA implementation. Advanced caching engines *can* observe the "Cache-Control" and "Expiration" directives that *can* be set in the farther tiers. They also provide the ability to observe caching policy directives defined by the SOA enterprise architect.

"WSDL annotations provide a powerful and extensible means for an SOA enterprise architect to centrally specify the internal policies needed to build a highly scalable, secure SOA infrastructure"

## Near Tier XML Caching and Optimization

Traditionally, XML processing was entirely handled at the business tier. The now mature XML-related standards and the desire for higher performance have resulted in the capability to perform many of these functions in the nearer tiers.

The nearest tier is what I refer to as the XML "front gate" – the subsystem that first receives and parses the SOAP message. Usually, the first task to be performed in a highly scalable SOA implementation is content-based routing of incoming SOAP messages. This technique uses knowledge of traffic and caching for message classification in which a series of rules or tests are cached by a server component that provides useful information about each message as it is processed. These tests are used to determine whether the message type is accepted by the receiving system and has an associated cached XML schema. Dropping unsupported message types saves processing time. This XML content-aware classification is the basis for routing and load balancing SOAP messages to different servers based on various characteristics of the messages. Classification is an important ingredient in efficient protection against malicious content, including denial-of-service attacks. XML rules or tests are most frequently expressed as a series of XPath statements, a W3C-standardized language for finding any XML content in a message and making assertions against that content. Similar in many ways to the SQL statement caching performed at the EIS tier, a sophisticated SOA near-tier component will precompile and cache these XPath statements for the most rapid possible evaluation against incoming XML messages.

A properly classified SOAP message should then have its schema validated. While Web browsers can deal with malformed HTML content by simply ignoring the unrecognizable content, the same is seldom true of XML applications. It is critical for network secu-

| SOA Function | Standard Library Throughput | RAX-CP Throughput | Improvement Factor | Comment |
|---|---|---|---|---|
| **Schema Validation – Large Document Set** | 52 Megabits per second | 1,340 Megabits per second | 25x | Xerces was used as the standard library. The test schema validation was run with 45 different large SOAP messages, with an average size of about 280K. |
| **Schema Validation – Small Document Set** | 3,760 validations per second | 34,892 validations per second | near 10x | Xerces was used as the standard library. The test schema validation was run on 131 different small SOAP messages, with an average size of about 2K. |
| **Content-Based Routing** | 132 message routes per second | 10,012 message routes per second | 76x | Saxon was used as the standard library. Router observed 63 rules and 42 destinations. |

Table 1 • **Summary of the throughput improvement that can be achieved by effectively applying the RAX-CP to SOA functions**

| Attribute | Default Value | Description |
|---|---|---|
| Tcs.cacheable | No | Response is cacheable by a trusted caching server |
| Tcs.lifetime | 30 | Specifies how long in seconds the trusted caching server should be considered fresh after storing the object in its cache |
| Tcs.private | No | Although the response is cacheable the trusted caching server must be stored in a consumer context cache |
| Tcs.invalidationOperations | None | A list of SOAP operations that would immediately cause the trusted caching server to invalidate the cached response, regardless of lifetime |
| Tcs.loadSchema | Yes | The trusted caching server should load and cache this schema |
| Tcs.allowSchemaImport | No | The trusted caching server should load and cache schemas from this location |
| Tcs.validateRequest | Yes | The trusted caching server should validate requests against a cached schema |
| Tcs.validateResonse | Yes | The trusted caching server should validate responses against a cached schema |

Table 2 • **Examples of caching and XML acceleration attributes**

rity and performance to ensure that each message is correctly composed and can be safely handled by the receiving applications in the farther tiers. XML message types are described in a formal grammar known as an XML schema. This grammar, processed against the individual message, is used by an XML validator to ensure that the message can be correctly and safely handled. XML schema validation is a CPU-intensive operation, consuming as much as 30 percent of the overall time spent processing each XML message. When XML Schema validation is performed on a server in the nearer tier, the computing resources of the farther tiers are significantly off-loaded. XML schema validation should also be performed on outbound responses. This ensures that dynamically composed XML messages do not compromise unprotected Web service intermediaries and that sent messages conform to content-level agreements between partners.

Caching is important to smooth schema validation processing. While XML may describe an infinite number of message types, within any given application environment there is a limited number of types of messages that can be handled. Therefore, an intelligent SOA-processing component will cache the XML schemas for those messages that an SOA implementation will handle. The cached schemas are precompiled into a binary data structure to increase the execution speed of validation. In most validation systems, the XML message must be fully parsed before it is validated, but a more sophisticated validator may be able to scan the message to determine whether the message type is supported before invoking full validation. Messages that are not supported by the SOA implementation can be discarded at the least possible cost.

In order to meet the heavy computational requirements of schema validation and content-based routing, using a device that can perform these computations in ASIC hardware is a recommended practice. One such device is the Tarari RAX Content Processor (RAX-CP). Table 1 summarizes the tremendous throughput improvement that can be achieved by effectively applying the RAX-CP to these important SOA functions.

## WSDL Annotations – The New Sheriff in Town

In an SOA, caching optimizations can be provided by many SOAP nodes, including service providers, clients, and intermediaries. Caching is not even limited to HTTP traffic since SOAP is not bound to any specific transport. A generic, transport-independent caching control mechanism is needed.

The common denominator for setting policies that can be observed throughout an SOA implementation is the Web Service Descriptor Language (WSDL). The primary objective of WSDL is to specify the operations and interfaces so that a consumer can employ a Web service without any knowledge of its underlying implementation. The WSDL is like the contract between the Web services consumer and provider. The designers of the WSDL message format had the foresight to allow seemingly innocuous annotations to be added to a SOAP interface. To external consumers, these annotations are meaningless and therefore ignored, but inside a Web services–based infrastructure, these annotations can provide instructions to trusted components running at different tiers. These annotations can advertise reliability capabilities and transactional properties, as well as provide caching and other optimization directives.

WSDL annotations provide a powerful and extensible means for an SOA enterprise architect to centrally specify the internal policies

“An intelligent, ‘SOAP-aware’ Web-caching tier can provide many important optimizations to an SOA implementation”

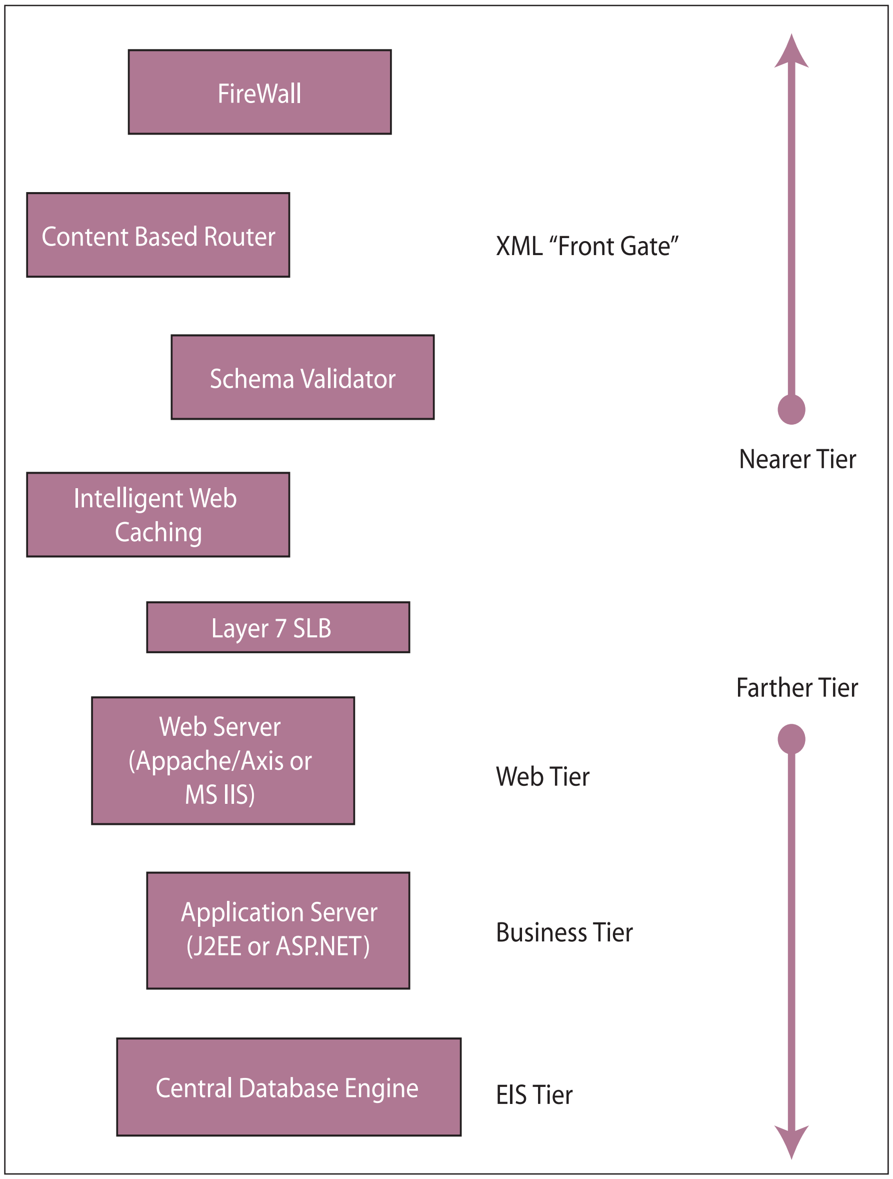


Figure 1 • **Multitiered SOA**

needed to build a highly scalable, secure SOA infrastructure.

## Consolidating-Near Tier Caching Functionality

Deployment of Web service–caching techniques is greatly simplified when multiple functions can be performed in a single appliance. An appliance that consolidates the XML "front gate" functionality of the XML content-based router and schema validation with the advanced HTTP Web-caching engine that is necessary for caching SOAP responses works best. An intuitive browser-based management interface to control the powerful XML schema validation and content-based routing features of an integrated RAX/CP covers all bases. XML caching and optimization should be easy to configure by annotating enterprise WSDLs to define policies that are consistent with your SOA design objectives. Table 2 presents some examples of caching and XML acceleration attributes that could be associated with the operations exported by a Web service from a good XML caching appliance.

## Conclusion

A senior business manager once taught me that the key to running a successful organization is to always push work down to the lowest level possible. Of course for this to work the manager must be able to trust that the delegated tasks will be carried out competently. This philosophy also holds true in an SOA implementation. If possible, the SOA enterprise architect should delegate work down to the cache processing of the nearer tiers. However the SOA enterprise architect must ensure that the processing that takes place in the nearer tiers is securely performed without compromising the integrity of the databases of the farthest tier. By specifying XML content processing and SOAP-aware caching using WSDL annotations, the SOA enterprise architect can delegate this work to the nearer tiers in a trusted fashion.

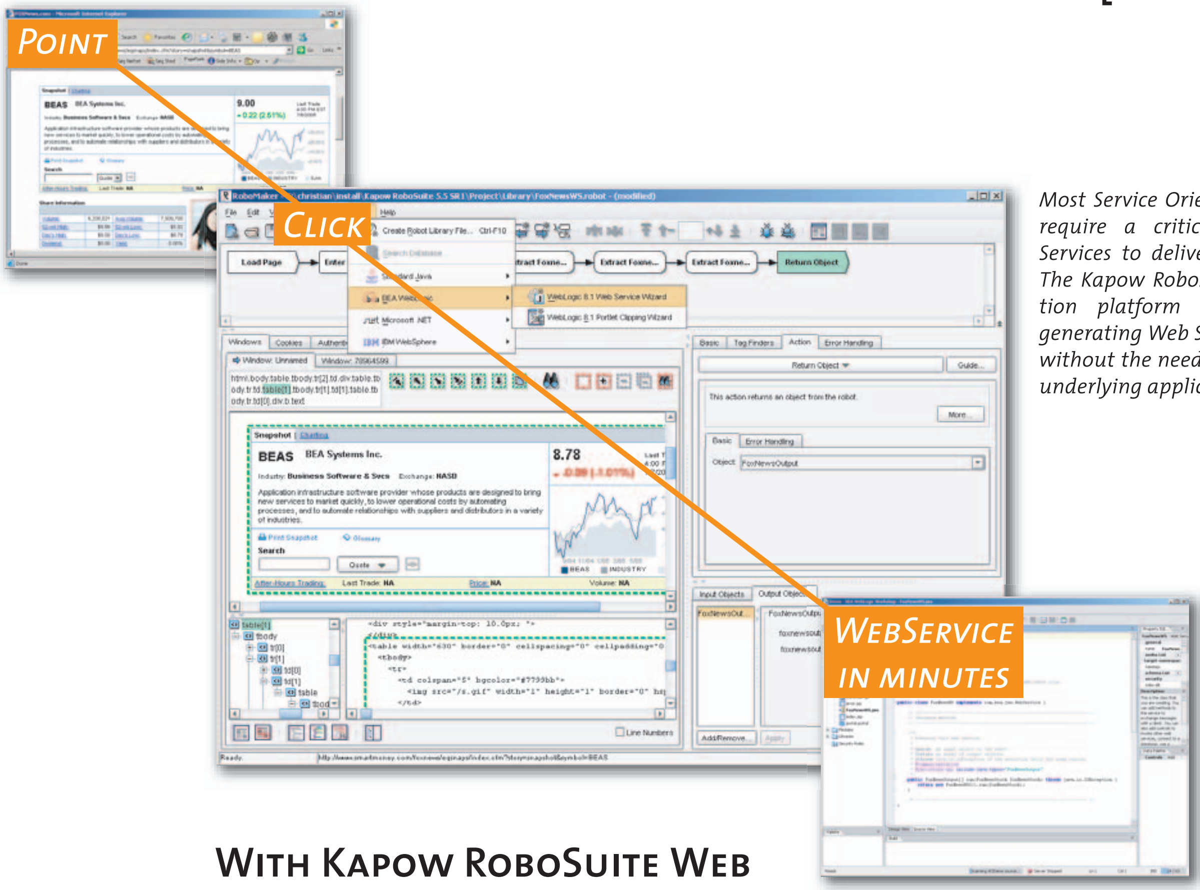
## References

- "Dynamic Statements Cache: Getting the Most Out of It" by Namik Hrle. *The IDUG Solutions Journal*, Volume 8, Number 2 (September 2001): www.idug.org/idug/member/journal/Sept01/articl08.cfm
- Using JCache to Save Money – Nigel Thomas: http://java.sys-con.com/read/37551.htm
- XML Web Service Caching Strategies – Matt Powell, Microsoft: http://msdn.microsoft.com/library/en-us/dnservice/html/service04172002.asp
- Container-managed persistence features: http://publib.boulder.ibm.com/infocenter/wsdoc400/index.jsp?topic=/com.ibm.websphere.iseries.doc/info/ae/ae/rdat_cmppers.html
- Package javax.util.jcache documentation: www.jdocs.com/jcache/1.0/api/javax/util/jcache/package-summary.html
- J2EE 1.4 API Specification: http://java.sun.com/j2ee/1.4/docs/api/index.html
- Microsoft Research: Caching of XML Web Services for Disconnected Operation, Venugopalan Ramasubramanian and Douglas B. Terry (December 2004): http://research.microsoft.com/research/pubs/view.aspx?type=Technical%20Report&id=844
- Tarari Random Access XML Content Processor: www.tarari.com/rax/
- *Random Access XML Programming Assisted with XML Hardware* by Michael Leventhal, XML 2004, November 2004, Washington, D.C.: www.idealliance.org/proceedings/xml04/papers/299/RandomAccessXML

**tyohe**@stampede.com

# Leverage existing IT to generate Web Services [ in minutes]

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



*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

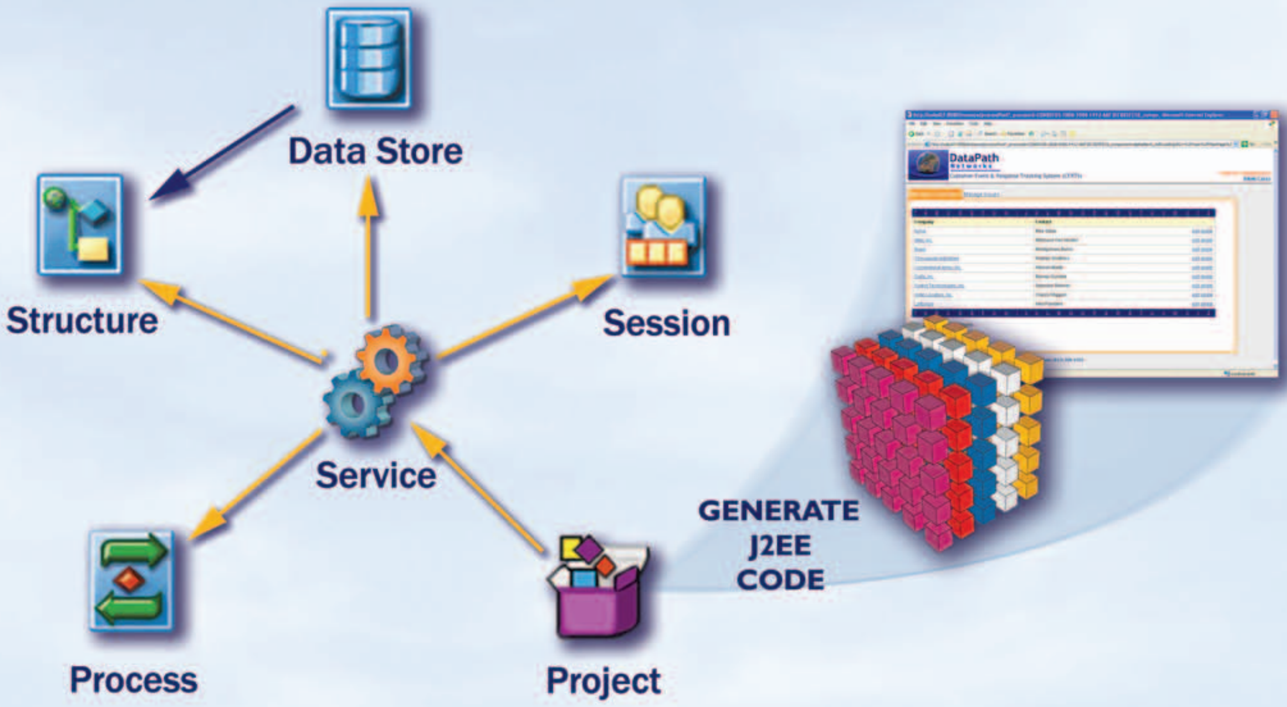
kapow TECHNOLOGIES

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.

# 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**

Skyway software

(813) 288-9355