SYS-CON

THE WORLD'S LEADING MAGAZINE DEDICATED TO WEB SERVICES TECHNOLOGIES

FOR ALL THE LATEST NEWS AND INFORMATION VISIT:

SOA WORLD MAGAZINE

OCTOBER 2007 / VOLUME: 7 ISSUE 10

Web 2.0-based Rich Service Consumer Ecosystems

PLEASE DISPLAY UNTIL DEC 31, 2007

$6.99US $7.99CAN

10

0 71486 03420 9

_INFRASTRUCTURE LOG

_DAY 84: Feeling really disconnected. We're not getting the most out of our existing assets. Service and application integration is a nightmare. Our connections are restrictive. We've got to stop working on these islands.

_Please rescue me from this lack of connectivity.

_DAY 87: I've taken back control with IBM WebSphere solutions. Now we can service-enable and connect our existing assets for mission-critical goals. We can reuse existing applications and save money by eliminating redundant systems. Now we're ready for any SOA integration project.

_Plus, no more jellyfish stings.

Download the enterprise service bus white paper at:
IBM.COM/**TAKEBACKCONTROL**/CONNECT

Uncle Sean

WRITTEN BY **SEAN RHODY**

I have no children myself, but I've watched my nieces and nephew grow from newborns to walking, talking, independent individuals over the years. To me, one of the most fascinating parts of watching a child grow is seeing them go from their first tentative, hesitant steps to toddling around, grabbing the furniture at every opportunity but gaining mobility to finally running around and crashing into my legs more often than I care to think about.

We've been speaking about service-oriented architecture here in the magazine for over four years now. In many ways, I've felt like an uncle to SOA as well. In the very beginning, when almost no one had even considered a service-oriented enterprise, we were at the newborn stage – people wondered about XML, WSDL, and UDDI, and how best to create a service.

As an industry we're well past that infant stage and have moved on from crawling to the first hesitant steps. We now have answered many of the initial questions (XML –good, UDDI – nice-to-have but not critical) and have faced the next series of questions in our journey toward SOA adulthood. These questions include things like how to create a transaction, what to do about security, and even to a certain extent how to manage our new world of loosely coupled services.

We've even begun to tackle some more meaty questions as we begin pulling ourselves along the furniture. We know enough to realize that moving to a service-oriented enterprise means not just Web services, but business processes and their management and governance. More important, we have now begun to gain the awareness of the business community that service orientation is as much about changing how we do business as it is about how we do information technology. This is a powerful and sometimes frightening realization.

Some years ago there was a fairly sensational article that posed the question "Does IT matter?" with the somewhat implicit position that in fact it did not – that IT is now commoditized and unworthy of any further investment; there is no competitive advantage to be gained from IT.

I've come to believe that SOA and SOE is an opportunity for IT to once again add value and allow an organization to differentiate itself amidst its competitors. For SOA, while founded on information technology, is truly about business transformation. It is about going from those first baby steps where IT created services just to prove they could to an organization that truly knows its mission and values, and applies them in all of its operations. Without fail, every organization I have worked with that has moved beyond the first simple Web services has discovered what should really come as no surprise – the way we do business, supported by automation, is not fully aligned with the goals of the organization. There are countless reasons why this happens – mergers, software packages that were built as silos – but the bottom line is that a move to SOA is about transformation of the business and fully aligning the services within the organization at a business level.

Of course it's easier to acknowledge this than it is to accomplish it. An organization is not just made up of software; it's made up of people. People have a natural need to organize, as well as a certain resistance to change. Constant change is difficult for many people to accept, regardless of the reality that nothing remains the same for very long. Embarking on a journey that says we're going to change how we do everything –where people sit, who they work for, what they do – is very difficult. It's a constant balance between change and security, providing just enough stability to the organization to ensure that business can be transacted. It's not easy.

But it is necessary. Take it from Uncle Sean, it's time to let go of the furniture and start to walk on our own. Next time we'll talk about the diapers.

About the Author

Sean Rhody is the editor-in-chief of **SOA World Magazine**. *He is a respected industry expert and a consultant with a leading consulting services company.* *sean@sys-con.com*

SOA WORLD MAGAZINE
www.SOA.SYS-CON.com

EDITORIAL OFFICES
SYS-CON MEDIA
577 CHESTNUT RIDGE ROAD, WOODCLIFF LAKE, NJ 07677
TELEPHONE: 201 802-3000 FAX: 201 782-9637
SOA World Magazine (ISSN# 1535-6906)
Is published monthly (12 times a year)
By SYS-CON Publications, Inc.
Periodicals postage pending
Woodcliff Lake, NJ 07677 and additional mailing offices
POSTMASTER: Send address changes to:
SOA World Magazine, SYS-CON Publications, Inc.
577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677

Altova® MapForce® 2008 – The premier data conversion, transformation, and integration tool.

ALTOVA® mapforce® 2008

Give your data direction

Link up with MapForce®, and exchange data with ease.

Spied in MapForce 2008:

- ValueMap function for mapping numeric values to names, etc.
- Support for aggregate functions: SUM, MIN, MAX, AVG, STRING-JOIN
- Enhanced options for XML output

Altova® MapForce 2008, the award-winning data conversion, transformation, and integration tool, makes it easy to exchange data between XML, databases, flat files, EDI, and/or Web services. Simply drag connecting lines from data sources to targets and drop in data-processing functions. MapForce instantly converts the data or auto-generates a data integration application for royalty-free use. Languages for code generation include: XSLT 1.0, XSLT 2.0, XQuery, Java, C++, and C#. Get connected!

Download MapForce® 2008 today: www.altova.com

MapForce is also available as part of the value-packed Altova MissionKit™ software bundle.

ALTOVA®

Strategies for Versioning Web Services

Environment and shifts in roles or responsibilities

WRITTEN BY **SAMEER TYAGI**

As architects and developers continue to design and implement Web Services in distributed environments, they are faced with a versioning issue – namely, how do you deprecate, evolve, and continue to use different versions of the same service with multiple service consumers. This article explores the implementation issues and identifies versioning patterns and strategies that can be employed to resolve them. Practical code examples using Java API for XML-based Web Services (JAX-WS) can be downloaded from the online version of this article at http:soa.sys-con.com.

The versioning issue isn't unique to SOA and Web Services and is an integral part of any software development. API, object, and class versioning have been around for a long time. Unique to the problem space is the environment and shift in roles or responsibilities. In distributed SOA environments control is decentralized – the service producer doesn't necessarily control the implementation technology or design of the service consumer (and vice versa). Moreover, the service producer may not be the same as the deployer and may have no control over the eventual deployment environments. The glue holding the two together is the service contract and the service level agreement. The former is usually the interface (i.e., the sum of WSDL, schemas, policies, etc.) and the latter is usually an out-of-band negotiated agreement defining the quality of service requirements (i.e., the sum of the "ilities" – scalability, reliability, availability, etc.) that the implementation provides. Architecturally the contracts can be producer- or consumer-driven.

Given this framework the architectural problem space is something like this. Consider that a service S_1 is developed and consumed by clients C_1. Business or technical requirements changes to the original service implementation result in service S_2. The consumer space for this service is C_2 as shown in Table 1.

	Service 1	Service 2	Service n
Consumers 1	✓	?	?
Consumers 2	?	✓	?
Consumers n	?	?	✓

Table 1 Service evolution from S1 to S3

If the consumer space for Service Sn were limited to C_n there would be no problem. Only new clients use new services. However, if this space is a defined by set (C_1U C_n) or (C_1 U C_{n+1}) – then versioning needs to be addressed.

The WC3 Technical Architecture Group (TAG) categorizes versioning schemes that mitigate coupling issues into four categories that we can adapt to our problem space:

- **None** – Service Sn doesn't distinguish between different consumers and must therefore tolerate all changes – Service S_1 handles request from C_1 U C_n
- **Backwards-compatible** – Service S_n handles requests from C_1 U C_n
- **Forwards-compatible** – Service S_1 handles requests from C_1 U C_n
- **Big Bang** – Service S_1 aborts processing for requests from C_2 U C_n

– CONTINUED ON PAGE 31

About the Author

Sameer Tyagi, an enterprise architect, focuses on strategy, design, and implementation of large-scale enterprise applications. His publications include several books and periodicals on technology. He can be reached at http://sameertyagi.blogspot.com. sameer_tyagi@yahoo.com

SOA WORLD MAGAZINE
www.SOA.SYS-CON.com

CORPORATE

President and CEO
Fuat Kircaali fuat@sys-con.com

Senior VP, Editorial & Events
Jeremy Geelan jeremy@sys-con.com

ADVERTISING

Senior VP, Sales & Marketing
Carmen Gonzalez carmen@sys-con.com

Advertising Sales Director
Megan Mussa megan@sys-con.com

Associate Sales Manager
Corinna Melcon corinna@sys-con.com

SYS-CON EVENTS

Event Manager
Lauren Orsi lauren@sys-con.com

Event Associate
Sharmonique Shade sharmonique@sys-con.com

CUSTOMER RELATIONS

Circulation Service Coordinators
Edna Earle Russell edna@sys-con.com
Alicia Nolan alicia@sys-con.com

SYS-CON.COM

VP information systems
Bruno Y. Decaudin bruno@sys-con.com

Consultant information systems
Robert Diamond robert@sys-con.com

Web Designers
Stephen Kilmurray stephen@sys-con.com
Richard Walter richard@sys-con.com

ACCOUNTING

Financial Analyst
Joan LaRose joan@sys-con.com

Accounts Payable
Betty White betty@sys-con.com

SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM
1-201-802-3012 or 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

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

Altova® XMLSpy® 2008 – The industry standard XML development environment.

ALTOVA® xmlspy® 2008

Immerse yourself in XML intelligence

Dive into XMLSpy®, and take XML development to greater depths.

Spied in Version 2008:

- Support for Microsoft® Word, Excel, and now PowerPoint Open XML formats
- Direct editing of database content via database query window
- XPath evaluation across multiple files at once
- Support for XInclude and XPointer

Altova® XMLSpy, the industry standard XML development environment, is vital for modeling, editing, transforming, and debugging XML applications. Breathe life into your plans with the world's leading XML editor, the original graphical schema designer, a code generator, file converters, debuggers, profilers, full database integration, support for XSLT, XQuery, WSDL, SOAP, and a trove of treasured XML utilities and usability aides. Become a markup mastermind!
Download XMLSpy® 2008 today:
www.altova.com

XMLSpy is also available as part of the value-packed Altova MissionKit™ software bundle.

ALTOVA®

Web 2.0-based Rich Service Consumer Ecosystems: Usability and Expectations

The usability perspective in RIA-based service consumer design

WRITTEN BY **MAYANK MATHUR**

The Web is evolving as an open platform with rich user interface capabilities of desktop clients. This has triggered user-driven management of service consumer ecosystems, expanding the reach of SOA with rich interactive controls and Web 2.0 tools to access the Web content and services. However the usability dimension of these Web 2.0-based service consumer ecosystems is often ignored, leaving doubt about whether present usability testing techniques in Web-based systems are capable enough to guarantee a usable experience in RIA-based service consumer systems.

While the focus of Web 1.0 was content delivery and communication, it has evolved to Web 1.5 showcasing content personalization and multi-level communication. In progressing towards Web 2.0, which manifests "authoring and collaboration," it has emerged as a key facilitator for creating rich SOA consumer ecosystems to augment the benefits of SOA including interoperability, reuse, and standardization. Web 2.0 is about leveraging standards (CSS, JavaScript, etc.) to deliver engaging, interactive, and integrated content and services in ways that allow users to focus on the task and the customer rather than the interface.

Rich Internet Applications (RIA)-based SOA front-ends offer a better user experience than static HTML Web pages or portals leveraging services because of an:

- Interactive and engaging interface, taking advantage of desktop richness and responsiveness, bringing content to users through controls with a low learning curve
- Direct manipulation of data with real-time input validation and

data responsiveness, all in a one-page smooth flow rather than page-by-page browsing.
- No more waiting for undo actions, page refresh and downloads.
- No more limitation of sequenced Web control manipulation for a task and the possibility of moving back and forth between steps.
- Richer page elements like grids, pop-ups, dialog, tabs, contextual dropdowns and right-mouse menus, with drag-and-drop and resize capabilities.
- Easy RIA installation and upgrades with accessibility regardless of operating system.
- Decentralized social networks with rating reviews, and feedback: Full Collaboration.

RIA Scenario	Consequence in Usability Perspective
Business influences design decisions	Flawed business model
Incorporating revenue-generating features	Basic functionality and objective overlooked
Conversion of application from desktop-based to browser-based for minimum deployment cost	Traditional desktop systems become slower; inefficient and inhibited browser-based apps.
Customized design approach	Complicated interface with low user yield

Table 1 Scenarios for degraded user experience

RIAs are powerful and attractive alternatives to HTML for enabling complex interactions. However, they might have user experience problems as depicted in Table 1.

RIA Usability Challenges

- Complex, unconventional way of presenting content; No defined standards as of now
- Unpredictive behavior of RIA controls; Users need to be careful and attentive while using them. They appear and disappear fast; variety of possible behavior-disengaging experience.
- Inconsistent and confusing navigation and nomenclature with concurrent activities.
- No calls can be made to the Web server when JavaScript is turned of and with no data transfer pages will be inaccessible.
- AJAX quick updates are vexing for people with partial sight and screen readers can easily miss them. Appropriate alerts (with checkboxes or pop-ups) along with sound will make it noticeable and improve the screen reader experience.
- Optimum richness may save "degraded user experiences" for the vast majority of people comfortable with less interactive applications to accommodate user learnability. Moreover, people should be able to identify the usage of any interactive page element or control, i.e., perceived affordance as defined by Don Norman in "The Design of Everyday Things." Uniform use of these controls would facilitate user learning based on past experience. The challenge is to develop an interactive application with user-centric design while providing alternative keyboard means for mouse-driven events.
- RIAs have the concept of in-page refresh, as such the back button would bring them to the same page, thereby confusing them. A different navigation system is required for static Web page refresh and in-page refresh.

RIA-based Web and service consumer ecosystems need to look at usability to enhance the user experience and excite users to visit again. These systems have to be highly accessible, visually consistent, accurate and engaging, oriented around a natural user task flow. Some basic questions while designing such a system include:

- What is the goal of people entering the system and why did they exit?
- Is the page layout and form design intuitive and interactive enough to retain the user?
- Is the site navigation and nomenclature consistent enough to ease task flow? Do we have a sitemap?
- Is the content usable to facilitate online reading for all classes of people?
- Is the site on a par with competitive trends in services and user experience?
- Does in-page refresh with appropriate dialogs and opening functions in new browser windows optimally balance to facilitate a beginner's task?
- Do we have proper feedback on the controls we are implementing? Is it maneuverable with appropriate perceived affordance?
- Does information searching require too much user attention and mouse events like clicking, dragging, and scrolling. Aren't we cluttering one page too much?
- Are we providing alternatives for content search as is common with static Web pages?
- Do the customers understand the way information products and services are presented?
- Do advertising, illustrations, and graphics on the home page convey the right message?
- How efficient and forgiving is the mechanism to collect information from the user?
- Finally, are users expected to behave the way a site is designed? After all we develop features to be used by users in their own natural way. Usability testing directs developers to design systems so people don't have to modify their approach to achieve their goal.

The Usability Testing of RIA Applications

Usability testing evaluates how easy an RIA application is to learn and use. There should be optimum balance between business needs, technology potential, and user expectations. Traditionally, usability refers to the "extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use." We restructure this definition for RIA usability to the "ability of a user to utilize rich controls for an intended use interactively, independently, and consistently resulting in reduced task time and enhanced user

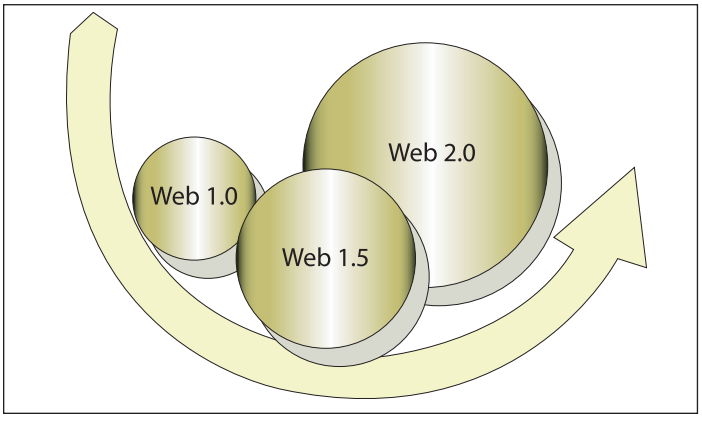


Figure 1: Web Evolution

experience." Human Computer Interaction (HCI) in this context would be "the discipline concerned with the design, evaluation, and implementation of usable RIAs and studying the optimum balance between traditional user expectations and rich technology innovations."

Usability testing for RIAs would involve analyzing UI design on established usability guidelines, and evaluating user task time and job satisfaction while using rich Web 2.0 controls through tools or first-hand observation and recommending fixes for the flaws. A commonly suggested approach is to have an optimal combination of restraint while implementing needed rich widgets with significant business usage and using traditional means where easier for a usable experience. After all use of traditional controls will phase out with time and it will take time for rich interactive widgets and RIA prowess to become commonplace. The different perspectives of usability testing in RIA context is explained below:

Planning & Prioritizing for Usability

Planning such a system requires the inculcation of a usability perspective while identifying site objectives and goals. During development we should focus on the form under development and have a clear idea of how it will function for the end user. This system should be delivered on a mental model of natural user task flow. The usability perspective in RIAs requires decisions about what functionality should execute on a new page and what is best suited to a partial page update. Incorporating user-generated content along with ratings and reviews enhances the usability of RIAs. Their design should comprehend visual perception and user behavior. Key tasks to be planned and techniques to be used in an RIA context include:

A Holistic View of RIA Controls from a Usability Perspective

Any usable RIA control can be thought of as having four characteristics:

Ability: The developer's ability to render rich and interactive control while using rapidly evolving and intuitive Web technologies.
Functionality: This characteristic is used to deliver functionality benefiting the user and business as a whole improving task time, easy input, and real-time solutions with a rich experience.
Capacity: This characteristic hints at infrastructure for rendering RIA controls with customizable and interactive solutions to the end user with enhanced experience and significant business gains.
Learnability: This hints at the designing widgets for its higher retention and recall while using it with natural ease.

This fourfold usability delivers a successful and sustainable application with customer retention meeting the user's expectations

Perspective	Advantage of Usability Testing
For Users	Minimal learning curve, decreased user task time and errors, increased job satisfaction, and improved user performance
Developers	Improved user acceptance, less technical support and training of users, reduced software development and rework time and effort.
Designers	One can identify which designs are effective and which to avoid
Companies	Reduced redesigning, development, and technical support costs, increased sales, increased traffic, and improved user productivity and features usage.
Overall	Inexpensive. Can be applied to anything from prototypes to finished products, findings are immediate without any special equipment needed to test.

Table 2 Advantages of usability testing

Usability Inspection	Quick, cost-effective ways to identify usability flaws for RIA design. Suggests what features need richness and which should be left as is.
Prototype	Measure design effectiveness while watching users interact with RIA controls.
Web Standards Design	Evaluate against W3C guidelines and analyze functionalities against business needs and technical constraints for improved page load times, improved bandwidth usage, maintainability, accessibility and compatibility, and support for non-traditional devices.
Card Sorting	Explore how users structure and find information time and again. Helps us put content for maximum effect.
Navigation Design	Focus on making user navigation consistent, quicker, easy, and effective for intelligent and smooth task flow for the user.
Keystroke-level Model Analysis	Analyzing specific aspects of a task and the controls that are slowing users down, identifying controls that can be retained rather than redesign the whole thing.
Instructive Design	Optimal use of the design elements for RIA control showcasing its uniqueness and why users should use it and how to use it while providing requisite directions for its use thereby accelerating users learning new interfaces.
Competitive Analysis	Evaluate the usability, not only of a company's product, but also competitive products.
Accessibility Inspection	Instruct that the minimum features and alternatives be implemented to achieve basic, intermediate, and advanced levels of accessibility while checking through text-only browsers and screen readers.
Findability	Structure content for effective find. Track the extent to which a feature or product is understood and used by the end user.
User Testing	Identify key usability issues as experienced by target users in accomplishing tasks through Web 2.0 innovative features to ascertain any problems. The design team can have its design evaluated heuristically after the fixes are done. Evaluate the appropriate balance between richness and efficiency of the application.
Testing Tools	Check recorder, remote viewer, and playback functions to log, observe, analyze, and share any site to uncover usability flaws. It makes usability testing easy, quick, and accurate and can remotely observe and record every mouse movement, keystroke, and click for site navigation.
Eye Tracking	Session images show the path that users' eyes take when looking at a page, while heatmaps highlight the areas users pay the most attention to. Analyze what attracts customers and what they ignore.
Physiological Measurements	FMRI (track processing in brain); pupal diameter and GSR (track emotional changes); a skin-conductivity sensor can detect changes in sweat, temperature, and heart rate. It provides a fair picture of an interface's user appeal.
Color Blindness Check	Simulate a number of color blindness conditions for how site features may appear to color-blind users and plan corrective actions accordingly.

Table 3 Usability techniques

while enhancing his learnability and user experience. This can help in articulating important design decisions such as deciding how much richness to add – keeping in mind how existing functionality would be enhanced – while catering to existing infrastructure needs and reducing the user's task time.

Best Practices for RIA Usability

Stick to the Basics: Before implementing RIA innovations, involve a trained usability tester to find usability flaws. Designers should redesign per usability conformance.

Moreover, focus should also be on privacy and security policies and guidelines for error-handling Provide Alternatives. RIA accessibility depends on JavaScripting being enabled in browsers. Browsers should auto-detect this and redirect the user to an alternative form.

- **Use AJAX to Enhance the User Experience:** An RIA control should be provided if it makes a user task easier and increases learnability and efficiency. One should refrain from delivering a rich solution if the user doesn't benefit.
- **Develop for Users:** RIA can confuse the user. We need to develop

SOA for SMB's?

small budget
BIG QUESTIONS
mission-critical services
MAJOR PRESSURE
short timeline
reduced staff

JaxView

A flexible Web Services management solution for small and medium-sized businesses that need the following capabilities:

- Monitoring of Web services
- XML gateway and authentication
- Runtime SOA policy enforcement
- Automated Web service discovery
- Runtime message modification and routing

Download a free trial at www.managedmethods.com

JaxView
versatile, scalable, affordable

Managed Methods

to the target audience and objective use so the user can achieve his intended goals.

- **Usability Inspection:** Have a usability specialist attempt to judge RIA effectiveness on a predefined set of usability criteria to identify potential problems early.
- **Usability Test:** Participants should test effectively, evaluating design effectiveness for the task and disseminate the results for testing iterative solutions.
- **Report:** Report the essential design recommendations and fixes divided into short-term, long-term, and must-do while the features are retained as they are.

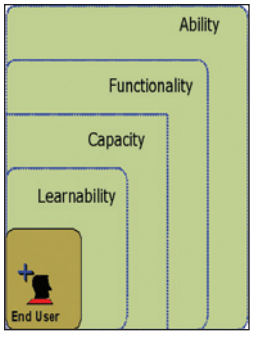


Figure 2: Holistic view

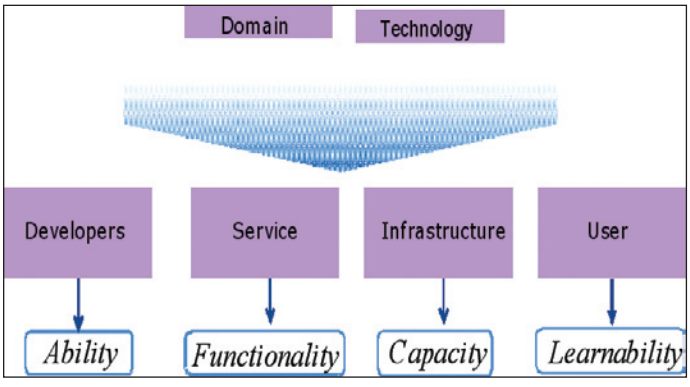


Figure 3: Involved parties

Conclusion

Understanding customers is critical in delivering on the business promise of SOA. Portals and Web 2.0 have emerged crucial SOA consumer ecosystem enablers. The Web 2.0 approach to consumer RIAs has primarily focused on the richness of the controls rather than the end user. To enhance user focus, there is a need to circumvent common design flaws in RIA with universal usability guidelines following right methodologies. This vision should use the right usability tools and techniques. In this article we have discussed the diverse usability perspectives of Web 2.0 RIA systems and have discussed appropriate techniques that can help RIA developers and end users by reducing task time, user errors, user disruption, training time, maintenance, and redesign costs.

References

- Megan Burns with Ron Rogowski and Steven Geller. Measuring Rich Internet Applications. Forester Press. www.forrester.com/go?docid=41476.
- Ron Rogowski with Kerry Bodine and Caroline L. Carney. Rich Internet Application Usability 101. Forester Press. www.forrester.com/go?docid=39615.
- Frank Spillers. "Demystifying Usability: Eye-Tracking Studies - Usability Holy Grail?" http://experiencedynamics.blogs.com/site_search_usability/2004/12/eyetracking_stu.html.
- Robert Hoekman, Jr. "Communicating Web 2.0 Through Design." http://www.thinkvitamin.com/features/design/communicating-Web-20-through-design.
- Deborah Abratt, Wayne Mallinson, and Antonet Bekker. "Usability Testing: Recipe for Success." www.stickyminds.com/getfile.asp?ot=XML&id=7072&fn=XDD7072filelistfilename1.pdf.

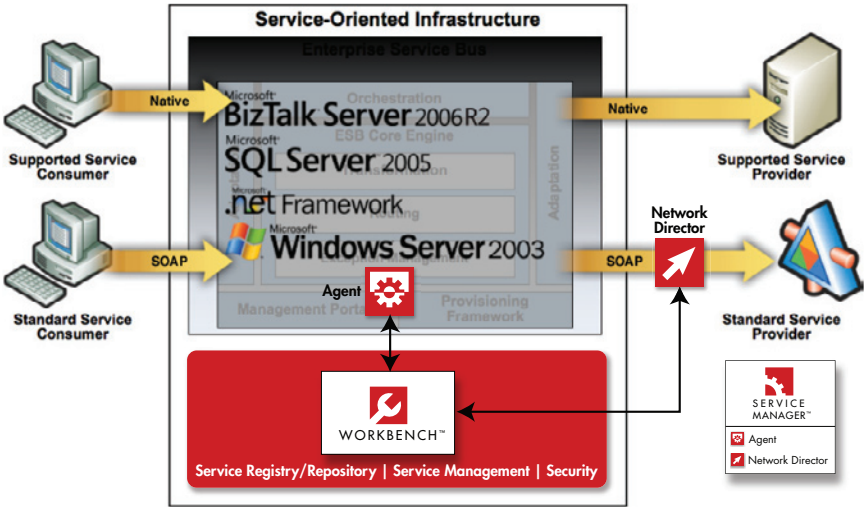
About the Author

Mayank Mathur works in a Web 2.0 research group in Software Engineering and Technology Labs, the R&D arm of Infosys Technologies, Ltd. He is currently working on a methodology for the usability testing of RIA applications. His other research interests include Semantic Web, AJAX, and RIA frameworks.

Mayank_Mathur@infosys.com.

SOA™ software

Integrated SOA Governance for BizTalk



- **Uniform Policy Management** -- Ensure consistent behavior across all enterprise applications
- **Heterogeneous Governance Automation** -- Automate processes for IT and SOA Governance
- **Dynamic Policy Enforcement and Implementation** -- Abstract applications from complex policies
- **Trust Mediation and Bridging** -- Seamless interoperability between security domains

SOA Software products provide a comprehensive closed-loop SOA governance solution. SOA Software's Service Manager and Workbench products enhance Microsoft's governance capabilities on both IIS and BizTalk platforms. The Workbench provides an integrated Service Management, Registry/Repository and policy management solution. The solution integrates with BizTalk as a BizTalk pipeline and provides full service lifecycle management along with features such as design-time compliance testing, service discovery via a UDDI v2/v3 interface, service provisioning, WS-S security and service monitoring regardless of protocol. Learn more at www.soa.com/microsoft

"SOA Software's Workbench for Microsoft BizTalk allows BizTalk to be a first class citizen in delivering a governed enterprise SOA platform. Workbench provides our customers with a comprehensive integrated SOA governance solution complementing BizTalk server, WCF, .NET and other Microsoft SOA platform components."

-- Marty Wasznicky
BizTalk Server Program Manager
Microsoft

www.soa.com/microsoft

Index XML Documents with VTD-XML

How to turn the indexing capability on in your application

WRITTEN BY **JIMMY ZHANG**

Traditionally DOM or SAX-based enterprise applications have to repeat CPU-intensive XML parsing when accessing the same documents multiple times. VTD-XML 2.0 introduces a simple general-purpose XML index called VTD+XML (http://vtd-xml.sourceforge.net/persistence.html) that eliminates the need for repetitive parsing of those applications.

This article combines various examples and the latest benchmark reports to show you how to get started with this indexing. This article also discusses various scenarios and use cases where you may find VTD+XML useful.

Avoid Repetitive XML Parsing with VTD-XML

As discussed in "Simplify XML processing with VTD-XML," to date one of underlying assumptions in XML application development is that an XML document must be parsed before anything else can be done with it. In other words, the processing logic of XML applications can't start without parsing. Frequently considered a threat to database performance, XML parsing is usually many times slower than other XML operations such as XPath evaluation. When those applications perform multiple read-only access to XML data that don't change very often, wouldn't it be nice to able to eliminate the overhead of associated repetitive parsing?

With the native XML indexing feature introduced in version 2.0 of VTD-XML, you can do precisely that. VTDGen, the class encapsulating various parsing routines, now adds "readIndex(...)" and "writeIndex(...)." VTD-XML 2.0 also introduces two new exceptions: indexWriteException and indexReadException.

Let me put those new methods into action and show you how to turn on the indexing capability in your application. Consider the following XML document:

```
<purchaseOrder orderDate="1999-10-21">
      <item partNum="872-AA">
        <productName>Lawnmower</productName>
        <quantity>1</quantity>
        <USPrice>148.95</USPrice>
      </item>
</purchaseOrder>
```

Below is a simple pre-2.0 VTD-XML code named "printPrice.java"

that prints out the content of the element "USPrice." Notice that it parses the XML file and then uses XPath to filter out the target nodes.

```
import com.ximpleware.*;
import com.ximpleware.xpath.*;
public class printPrice{
     public static void main(String args[]){
          VTDGen vg = new VTDGen();
          try{
               if (vg.parseFile("po.xml",true)){
                     VTDNav vn = vg.getNav();
                     AutoPilot ap = new AutoPilot(vn);
                     ap.selectXPath("/purchaseOrder/item/USPrice/text()");
                     int i=-1;
                     while((i=ap.evalXPath())!=-1){
                           System.out.println(" USPrice ==> "+vn.toString(i));
                     }
               }
           }catch(Exception e){

           }
      }
}
```

A few changes are needed to add VTD-XML's new indexing capability to the Java code above. First, you need to read in the XML document, parse it, and then write out the indexed version of the same XML document. From that point onward, your application can run XPath query or processing logic directly on top of the index, saving the CPU cycles of parsing the XML document again. The following code snippets (named "genIndex.java" and "accessIndex.java" respectively) show you how to generate and access the index. Notice that, when executed sequentially, both applications produce the identical output as "printPrice.java."
The first application (genIndex.java) reads in "po.xml" and produces "po.vxl."

```
import com.ximpleware.*;
import com.ximpleware.xpath.*;
public class genIndex{
 public static void main(String args[]){
  VTDGen vg = new VTDGen();
  try{
   if (vg.parseFile("po.xml",true)){
    vg.writeIndex("po.vxl");
   }
  }catch(Exception e){
  }
 }
}
```

The second application (accessIndex.java) loads "po.vxl" and filters the document using XPath expression "/purchaseOrder/item/USPrice/text()."

```
import com.ximpleware.*;
import com.ximpleware.xpath.*;
public class accessIndex{
     public static void main(String args[]){
          VTDGen vg = new VTDGen();
          try{
               VTDNav vn = vg.loadIndex("po.vxl");
               AutoPilot ap = new AutoPilot(vn);
               ap.selectXPath("/purchaseOrder/item/USPrice/text()");
               int i=-1;
               while((i=ap.evalXPath())!=-1){
                   System.out.println(" USPrice ==> "+vn.toString(i));
               }
          }catch(Exception e){
          }
     }
}
```

VTD+XML in 30 Seconds

Allowing XML parsing to be decoupled from application logic, the key in the example above is the index file "po.vxl," which conforms to the VTD+XML spec. What is VTD+XML? Since VTD-XML's internal representation of XML infoset is inherently persistent, VTD+XML, as the name suggests, is simply the binary packaging format that combines VTD records, LCs entries, and XML into a single file. The detailed technical spec can be found at http://vtd-xml.sourceforge.net/persistence.html.

A Simple Example

This section gets down to the nitty-gritty of the specification by manually composing, byte-by-byte, a VTD+XML index. For the sake of simplicity, this example chooses to index a simple XML document containing a single child-less root element whose parsed representation doesn't have location cache entries. This example also assumes a big-endian byte order (as in Java) and UTF-8 document encoding (the default character set). The name space awareness is set to false.

```
<root/>
```

The first four-byte word of the corresponding index file is 0x0102A000 containing:

- The VTD+XML version number (0x01) in the first byte
- The character encoding format (0x02) in the second byte
- The name space awareness, word length of LC entries in the last level, byte endian-ness of the platform, and VTD version as encoded in various bit fields in the third byte (0xA0)
- The document depth (0x0 as the root element has no child)

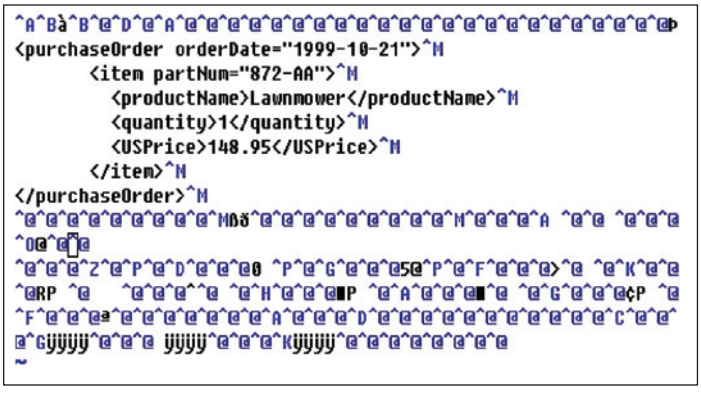
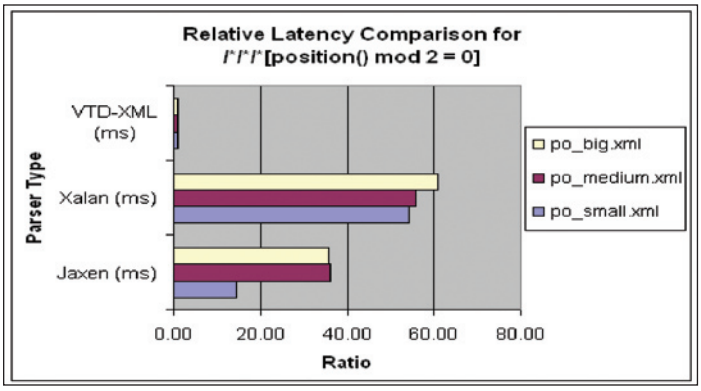
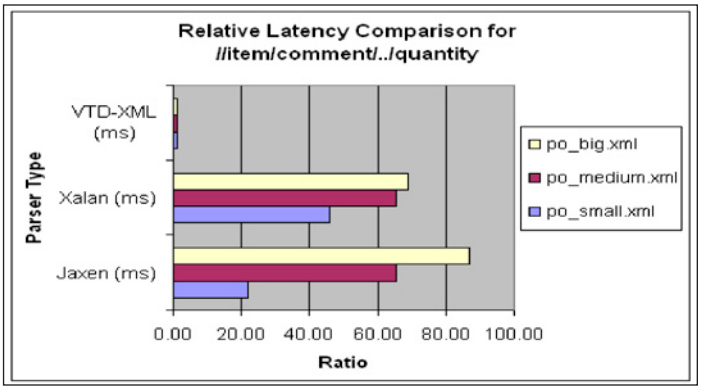
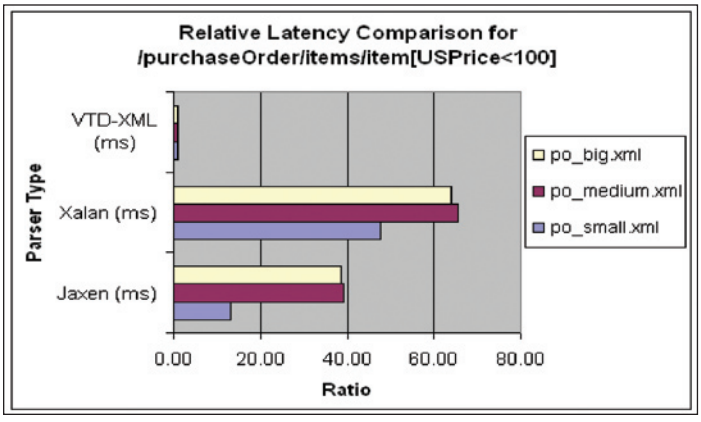
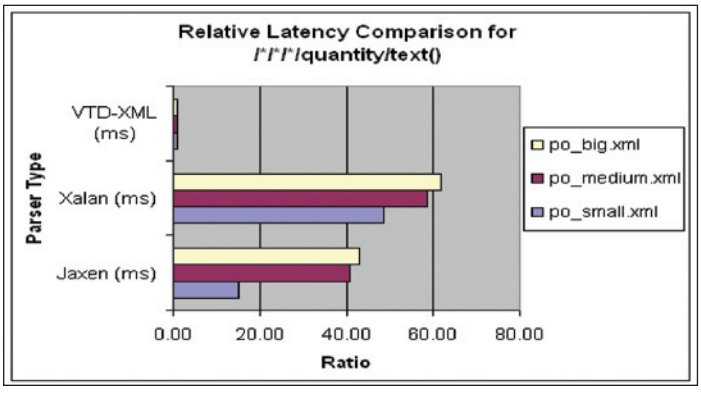


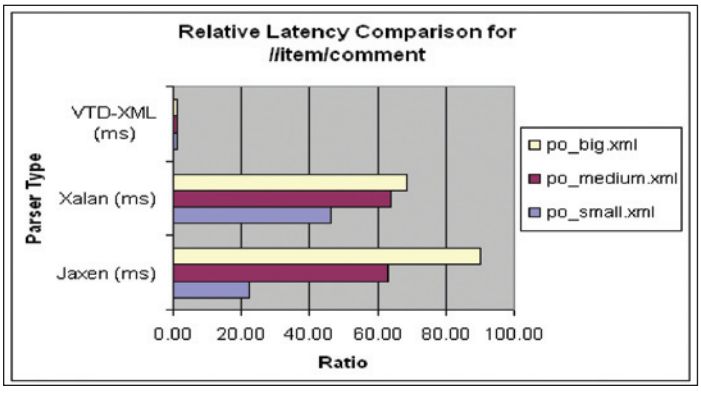
Figure 1 Read "po.vxl" in VI











The second four-byte word has the value of 0x00040001 containing:

- The number of LC levels supported by the VTD-XML implementation in the upper 16 bits (0x0004 in big endian)
- The root element index value in the lower 16 bits (0x0001 in big endian)

The next four four-byte words are reserved and set to zero. The byte order of all the ensuing 32-bit or 64-bit words is platform-dependent and specified in the third byte of the VTD+XML spec. The next eight-byte words indicate the size (in bytes) of the XML document, which equals seven in this example. Immediately following (0x3C726F6F742F3E00) is the byte content of the XML rounded up to an integer multiple of eight bytes by padding zero to the end.

The remaining part of VTD+XML index consists of multiple adjacent segments each containing an eight-byte word (0x0000000000000002 indicating the VTD record or LC entry count) followed by the actual content of the VTD records or LC entries. The first eight-byte word (0x00000000000000002) indicates that there are two VTD records that are 0xDFF0000000000000 and 0x0000000400000001.

The remaining three eight-byte words all have the value of zero indicating that the location caches in level one, two, and three have zero entry in the VTD+XML index.

As the final output, the VTD+XML index for "<root/>" is 88-bytes long and looks like the following hex:

0x0102A00000040001 0x0000000000000000
0x0000000000000000 0x0000000000000007
0x3C726F6F742F3E00 0x0000000000000002
0xDFF0000000000000 0x0000000400000001
0x0000000000000000 0x0000000000000000
0x0000000000000000

Benefits and Limitations

Because VTD+XML straightforwardly combines VTD and XML, it inherits all the benefits of VTD-XML parsing. When compared with existing XML indices (e.g., various pure-binary XML indices modeling labeled, ordered tree etc.), VTD+XML possesses many unique technical benefits:

- **General Purpose** – Before VTD+XML, most native XML indices only optimize specific types (e.g., the axis) of Xpath lookups. If an input query differs slightly from the index type, the query execution still has to resort to expensive parsing. Due to this limitation,

many native XML databases today require users to create multiple indices, one for each input query type so users can benefit from those indices. The problem is that XML database applications usually serve many types of queries that are unpredictable and complex in nature, often rendering the benefits of indexing insignificant. In comparison, VTD+XML is the first index that completely eliminates the cost of XML parsing and predictably speeds up any type of XPath query. It also works with namespaces exceptionally well.

- **Human Readable** – VTD+XML is also the first human-readable XML index. You can actually open it in a text editor to examine the XML text. Figure 1 is what "po.vxl" looks like in "vim." More than just a nice property, VTD+XML's human-readability offers distinct advantages over pure binary indexing schemes. Everything else being equal, keeping XML in its original format avoids the processing cost of converting to and from any binary formats. Moreover, what if your applications just wants to modify the XML payload, such as inserting into it a chunk of XML text extracted out of another SOAP message? What's the point of converting XML to binary formats? In a service-oriented heterogeneous environment, maintaining XML in its original format automatically retains the openness and interoperability. It just seems to me that the only loss-less equivalent of XML is XML itself, no less.
- **Superior Indexing Performance** – The performance of generating an VTD+XML index is identical to the parsing performance of VTD-XML, since both are essentially the same operation viewed from two different angles. On a 1.7GHz Pentium machine, it's reasonable to expect a sustained indexing performance of 50 MB/s-70 MB/s.
- **Easy To Use** – Usually adding a couple of lines (loadIndex(...) and writeIndex(...) as seen in the previous example) to your existing VTD-XML code is all that's needed to enable VTD+XML in your applications.
- **Compact** – The size of VTD+XML is usually about 30%-50% bigger than the size of the corresponding XML document. This is again consistent with the memory use of the VTD-XML processing model.
- **Platform Neutral** – Just like XML, VTD+XML is designed to be platform-neutral in that it explicitly includes information about the byte endian-ness of the platform on which the index is generated. Users of the C or C# version VTD-XML code can automatically recognize and make use of the index generated by the Java version.

At the same time, users of VTD+XML need to be aware of the following limitations:

- **Upper Limit on Document Size** – The maximum XML document size supported by VTD+XML is 2GB without name space support. With name space, VTD+XML supports a maximum of 1GB.
- **Lack of Support for External Entities** – VTD-XML currently supports five built-in entity references (&lt, &gt, &amp, &apos, and ") as defined in XML 1.0.

The Case Involving XML Content Update

Some of you may wonder: What if the subsequent XML operations involve content updates that shift the offset value? In general, those use cases often require the updated XML document to be re-indexed. And for large XML documents, you may argue that the cost of re-indexing can be quite significant. However, there are actually several workarounds, all aimed at reducing, even eliminating, the cost of re-indexing.

The first workaround: Instead of creating the VTD+XML index for a single big XML document, split the XML document into multiple smaller ones, each of which is then indexed using VTD+XML. From this point on, you only need to regenerate a VTD+XML index for those "updated" XML fragments that are usually a lot smaller and therefore cheaper to re-index.

VTD-XML 2.0 also introduced the "overwrite" feature that lets you modify XML content without needing to regenerate the index. The code below makes use of the VTDNav class's new "overWrite(...)" to change the text node of "<root>good</root>" from "good" or "bad." If the new content is shorter or equal in length to that of the old content, the method "overWrite(...)" fills up the non-overlapping portion of the text with white spaces and returns true. Otherwise, no change to the original content and "overWrite(...)" returns false.

```
import com.ximpleware.*;
class Overwrite{
   public static void main(String s[]) throws Exception{
         VTDGen vg = new VTDGen();
         vg.setDoc("<root>good</root>".getBytes());
         vg.parse(true);
         VTDNav vn = vg.getNav();
         int i=vn.getText();
         //print "good"
         System.out.println("text ---> "+vn.toString(i));
         if (vn.overWrite(i,"bad".getBytes())){
               //overwrite, if successful, returns true
               //print "bad" here
              System.out.println("text ---> "+vn.toString(i));
         }
    }
}
```

The "overWrite" feature may look simple, but it actually has unexpected implications for the performance of XML. Consider the database table design in which you specify the column width. You can now borrow the same technique for XML composition: By pre-serializing some extra spaces into text nodes or attribute values, you can make "in situ" updates to those nodes and do so without regenerating the index. You can even pre-serialize, in an XML document, dummy elements containing text nodes or attribute values whose initial values are entirely white spaces. Those dummy elements serve as templates in anticipation of a future content update, as shown in the example below.

The template

```
<purchaseOrder orderDate="          ">
      <item partNum="       " >
        <productName>          </productName>
        <quantity>          </quantity>
        <USPrice>          </USPrice>
      </item>
</purchaseOrder>
```

After "stamping" in the data

```
    <purchaseOrder orderDate="1999-10-21">
       <item partNum="872-AA" >
         <productName>Lawnmower  </productName>
         <quantity>1  </quantity>
         <USPrice> 100 </USPrice>
       </item>
</purchaseOrder>
```

And, by the same token, the concept of XML content deletion deserves a bit of rethinking as well. Instead of physically deleting an XML element, you can disable the XML elements by making them "invisible" to your applications to achieve the same goal. The benefit: you again avoid the need to re-index. Notice that this plays favorably to XML's strength as a loose encoding data format. Below is an example of setting the value of the attribute "enable" of an element to make it "invisible."

Before

```
 <purchaseOrder orderDate="1999-10-21">
       <item partNum="872-AA" enable="1">
         <productName>Lawnmower</productName>
         <quantity>1</quantity>
         <USPrice>148.95</USPrice>
       </item>
</purchaseOrder>
```

After

```
    <purchaseOrder orderDate="1999-10-21">
       <item partNum="872-AA" enable='0'>
         <productName>Lawnmower</productName>
         <quantity>1</quantity>
         <USPrice>148.95</USPrice>
       </item>
</purchaseOrder>
```

Applications Scenarios

There are at least two different views to make sense of VTD+XML as a practical solution to real problems. The first is a traditional view of native XML indexing. Alternatively, you can think of VTD+XML as a binary data format backwards-compatible with XML.

Native XML Indexing

In this view, you simply use VTD+XML as the basis for native XML data stores that serve the backend data needs of XML/SOA applications. By saving it as a BLOB (Binary Large OBject) in a more traditional database table, you obtain the additional capabilities such as concurrency and data integrity and replication. Being vastly superior to the awkward shredding-based XML to relational data mapping, VTD+XML fits exceptionally well in a pure XML/SOA environment. Have a lot of XBRL (Extensible Business Reporting Language) documents, or those big GML (Geography Markup Language) files? VTD+XML should equip you with horsepower never before available.

Binary Enhanced XML

VTD+XML also naturally extends the core capabilities of XML by boosting its processing efficiency to a whole new level. In other words, as a wire format, XML now has it all: not only is it easy to learn, human-readable, interoperable, and loosely encoded by design, performance-wise it also leads CORBA, DCOM, and RMI by a mile. When applied to XML pipelining, VTD+XML can potentially eliminate the repetitive parsing at each stage of the pipeline – an issue none of the existing XML pipeline specs (e.g., XProc and the XML pipeline definition language) address.

If it takes too long for you to push large documents over your DOM-based ESB (Enterprise Services Bus), how does 100MB around a single second sound?

Benchmark

This section shows you quantitatively the performance gain achievable using VTD+XML. The benchmark code measures the combined latency of VTD+XML index-loading (as in VTD-XML 2.0) and XPath evaluation of a specified number of nodes (the first five nodes in the set) in the result nodeset. The same code is also rewritten using the Xerces DOM parser and Xalan or Jaxen, both of which are popular XPath engines. The benchmark code used for the test can be downloaded here.

Setup

The environment for the benchmark has the following setup:

- **Hardware:** A Sony VAIO notebook featuring a 1.7GHz Pentium M processor with 2MB of integrated cache memory, 512MB of DDR2 RAM, and a 400MHz front-side bus.
- **OS/JVM setting**: The notebook runs Windows XP, and the test applications are obtained from version 1.6.0.6-b105 of JDK/JVM.
- **XML parsers and XPath engines:** The DOM code uses both Xalan (bundled in the JDK) and Jaxen over Xerces DOM (full node expansion). VTD-XML, on the other hand, uses the built-in XPath engine.

To reduce timing variations due to I/O, the benchmark programs first read XML files into the memory buffer prior to the test runs and output XML files into an in-memory byte array output stream. The server JVM is used to get peak performance. All input/output streams are reused whenever possible.

Three XML files of similar structure, but different sizes, are used for the test.

```
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-20">
    <shipTo country="US">
        <name>Alice Smith</name>
        <street>123 Maple Street</street>
        <city>Mill Valley</city>
        <state>CA</state>
        <zip>90952</zip>
    </shipTo>
    <billTo country="US">
        <name> Robert Smith </name>
        <street>8 Oak Avenue</street>
        <city>Old Town</city>
        <state>PA</state>
        <zip>95819</zip>
    </billTo>
    <comment>Hurry, my lawn is going wild!</comment>
    <items>
        <item partNum="872-AA">
            <productName>Lawnmower</productName>
            <quantity></quantity>
            <USPrice>148.95</USPrice>
            <comment>Confirm this is electric</comment>
        </item>
        <item partNum="926-AA">
```

```
            <productName>Baby Monitor</productName>
            <quantity>1</quantity>
            <USPrice>39.98</USPrice>
            <shipDate>1999-05-21</shipDate>
        </item>
        ...
    </items>
</purchaseOrder>
```

The respective file sizes are:
- "po_small.xml" ---- 6780 bytes
- "po_medium.xml" ---- 112,238 bytes
- "po_big.xml" ----- 1,219,388 bytes

The following XPath expressions are used for the test
- /*/*/*[position() mod 2 = 0]
- /purchaseOrder/items/item[USPrice<100]
- /*/*/*/quantity/text()
- //item/comment
- //item/comment/../quantity

Results

Absolute Latency

/*/*/*[position() mod 2 = 0]

file name	Jaxen (ms)	Xalan (ms)	VTD-XML (ms)
po_small.xml	0.401	1.521	0.028
po_medium.xml	16.255	25.131	0.449
po_big.xml	159.329	270.188	4.44

/purchaseOrder/items/item[USPrice<100]

file name	Jaxen (ms)	Xalan (ms)	VTD-XML (ms)
po_small.xml	0.441	1.612	0.0338
po_medium.xml	16.954	28.21	0.431
po_big.xml	174.201	288.18	4.499

/*/*/*/quantity/text()

file name	Jaxen (ms)	Xalan (ms)	VTD-XML (ms)
po_small.xml	0.47	1.534	0.0315
po_medium.xml	17.57	25.278	0.431
po_big.xml	190	272.958	4.412

//item/comment

file name	Jaxen (ms)	Xalan (ms)	VTD-XML (ms)
po_small.xml	0.805	1.689	0.0364
po_medium.xml	27.27	27.687	0.434
po_big.xml	398.57	304.103	4.43

//item/comment/../quantity

file name	Jaxen (ms)	Xalan (ms)	VTD-XML (ms)
po_small.xml	0.816	1.706	0.0372
po_medium.xml	28.367	28.338	0.435
po_big.xml	384.05	306.056	4.431

Observation

The benchmark results show that, after removing the parsing cost (by resorting to the index), VTD-XML now consistently outperforms DOM by two orders of magnitude, regardless of the message sizes. Interpreting the above results as the upper limit of how fast an XML content switch makes routing decisions based on the XPath output, VTD-XML's processing throughput, calculated by dividing the XML message size (not including VTD) by the latency, is around 250 MB/sec, roughly doubling the maximum throughput of a gigabit Ethernet connection. This means that switching/routing VTD+XML payloads based on simple XPath expressions is I/O-bound.

Conclusion

This article has introduced the latest indexing feature of VTD-XML along with the latest benchmark numbers showcasing the efficiency level it achieves. Prior to VTD-XML, an XML/SOA application written in DOM or SAX incurs the overhead of XML parsing, XPath evaluation and, optionally, content update. It's not uncommon that those overheads account for 80%-90% or more of the total CPU cycles of running the application. VTD-XML obliterates those overheads since there's not much overhead left to optimize. Using VTD-XML as a parser reduces XML parsing overhead by 5x-10x. Next VTD-XML's incremental update uniquely eliminates the roundtrip overhead of updating XML. Moreover, this article shows VTD-XML's innovative non-blocking, stateless XPath engine significantly outperforming Jaxen and Xalan. With the addition of the indexing capability, XML parsing has now become "optional."

"Benchmark results show that, after removing the parsing cost (by resorting to the index), VTD-XML consistently outperforms DOM by two orders of magnitude, regardless of the message sizes"

In other words, obstacles standing on the path to successful SOA have quietly disappeared. But this is just another starting point. It probably won't be difficult to see that none of its benefits would exist if VTD-XML stuck with excessive object allocation like DOM. In the context of XML processing, pure OO modeling of an XML infoset (e.g., string and node objects) just doesn't appear the right thing to do in the first place. Like anything else, OO has its weaknesses. The problems (e.g., DOM and SAX's problems) arise when one chooses OO for the sake of choosing it, and stops questioning its sensibility. To me, knowing when not to use objects is equally, if not more, important. Derived from the weaknesses, constraints, and limitations, VTD-XML strives to be the simple, sensible answer to the problems.

And, in the context of SOA, there are more questions on OO programming worth reflecting on. Among them, is OOP's API-based public contract suitable for building loosely coupled, document-centric Web Services applications? The answers, again, are likely to be surprisingly simple. ■

About the Author

Jimmy Zhang is a cofounder of XimpleWare, a provider of high performance XML processing solutions. He has working experience in the fields of electronic design automation and Voice over IP for a number of Silicon Valley high-tech companies. He holds both a BS and MS from the department of EECS from U.C. Berkeley.

crackeur@comcast.net

Multi-Enterprise SOA

The most powerful tool we have for accelerating growth & delivering acquisition value

WRITTEN **BY JOEL REED**

Merger and acquisition expenditures exceeded $2.7 trillion worldwide in 2005 and are expected to grow through 2009. However, according to McKinsey and Company, the big global strategy consultancy, "Half or more of the big mergers fail to create significant shareholder value....

"The sad conclusion is that an average corporate control transaction...delivers little or no value in the return." The fact that such astronomical levels of cash and equity are sunk into M&A activity when it can put shareholders at risk is far from comforting.

Service Oriented Architecture (SOA) has become a common IT approach to bridge the barriers of collaboration between business partners, customers, suppliers, and different organizations in one company. The adoption of SOA to connect dynamic business processes and disparate IT systems is initially a daunting task beset on all sides by potential complications and setbacks.

Adding M&A activity to the mix compounds the obstacles and complexity of committing to an SOA. When public companies leverage mergers and acquisitions as business strategies to achieve growth and increase shareholder value, another set of complications arise.

M&A Doesn't Always Pay – The Acquisition Death Spiral

Business acquisitions are on the rise, showing no sign of slowing. A recent Morgan Stanley survey found that the chief financial officers at Fortune 1000 companies consider M&A their top priority. The survey says that CFOs ranked M&A as the seventh most important use of capital. Because M&A is top of mind for CFOs, IT management must be prepared to accommodate – and in some cases steer – M&A strategies.

Although M&A is wildly popular, that doesn't necessarily mean that it's in the best interest of shareholders. Mergers and acquisitions can hinder service responsiveness and quality due to reduced, fractured, or even eliminated service and support. Post acquisitions, companies can struggle to grow revenue and become too cost-focused. As a result, the focus on shareholder value, derived from top-line growth at reasonable margins, can be lost.

As discussed in the Harvard Business Review in September 2006, "Companies typically create most of their value through day-to-day operations, but a major acquisition can create or destroy value faster than any other corporate activity." An unhealthy death spiral can start when cost-cutting reduces growth and customer loyalty, leading to lower revenue. Lower revenue leads to increased pressure to cut costs beyond the acquisition business case. And, the downward spiral begins...

Acquisition Value = Growth at Acceptable Margins

Executives need to think differently. According to McKinsey and Company, "Revenue deserves more attention in mergers, indeed, a failure to focus on [revenue] may explain why so many mergers don't pay off." In its study, only 12% of companies analyzed succeeded in accelerating revenue.

Instead of focusing on cost reductions, executives of merging companies can pounce on short-term opportunities and accelerate long-term growth by pressing three core levers:

1. Cross-selling and up-selling in a multi-channel world

2. Channel acceleration
3. Maintaining or enhancing service levels in a post-acquisition environment

Cross-Selling and Up-Selling in a Multi-Channel World

Acquisitions add new products to sell through existing channels that must gain accelerated sales momentum in order to maintain revenue growth. One way to accelerate sales is through cross-selling and up-selling. But there are barriers to cross-selling and up-selling newly acquired products – not the least of which is gaining visibility into existing customers through the incompatible or unconnected IT systems of an acquired company. Post-acquisition, executives must focus on gaining intelligence about existing customer bases by connecting disparate customer information systems and applications so sales teams can mine for cross-sell and up-sell opportunities. At the same time, the channels by which customers purchase products and services must be visible and manageable from anywhere at any time. Whether the channel is online, by phone, in-store, or through a reseller, the sales force must be given access to customer data that lets them effectively up-sell and cross-sell into the base of the acquired and acquiring companies.

Channel Acceleration

Acquisitions frequently bring new channels that may grow, not rationalize, over time. Partners of one organization may be competitors to another. Economies of scale achieved through a relationship with an acquired company might hamper efficiencies with the acquiring company. Mutually beneficial relationships must be quickly identified and cultivated to accelerate channel sales and lead generation. Again, the interoperability of IT systems and the sharing of information must happen quickly before potentially growth-driving channel partners go elsewhere.

Maintaining or Enhancing Service Levels in a Post-Acquisition Environment

Acquisitions can pollute the brand and hamper maintenance and service revenue if promises aren't kept and service level agreements (SLAs) aren't met. If companies turn inward to tediously rewire systems and processes to match perfectly, the organization can easily become misaligned with customer needs. Post-acquisition, there's rarely anything more important to long-term growth than first maintaining then improving customer service levels.

Interoperability versus Standardization

When seeking the optimal combination of these growth levers post-acquisition, fractured processes, data, and systems must be connected. For example, catalog, pricing, and configuration systems must be updated to reflect new offerings and new combinations. Order processing and billing systems need to present a single interface to the customer.

The natural inclination is to standardize everything. That inclination unfortunately delays revenue acceleration and adds to the death spiral. *CIO Magazine*, in its March 2007 issue states that the average TCO of ERP solutions is $15 million with the range going as high as $300 million. No wonder nearly 40% of the companies that McKinsey surveyed were able to hit cost-reduction targets after acquisition. Standardizing ERP is expensive...and risky.

Standardization vs. Interoperability

	Standardization	Interoperability
Short-term costs	Higher	Lower
Deployment time	Years	Months
Sales visibility	Product-centric	Customer-centric
Channel interface	Inflexible	Flexible
Long-term costs	Lower	Higher
Growth optimized	No	Yes

Figure 1

It's usually better to allow newly joined companies to operate in tandem with a focus on growth at acceptable margins than it is to attempt cost-cutting through standardization of core transaction platforms.

Implementing SOA Strategy Post-Acquisition

How can newlywed organizations get focused on growth, when all the complexity of different systems is getting in the way? One way is to loosely couple internal services and systems in both companies to cultivate growth with external parties without uprooting operations.

SOA lets the systems of acquiring and acquired companies interoperate without restructuring the enterprises or replacing underlying systems. By allowing systems to interoperate, the company can immediately tie together existing systems to cross-sell and up-sell offerings through all of its channels; to accelerate the channel; and to enhance service levels.

Growth-driving systems to be considered when developing a post-acquisition SOA strategy include:

- E-cataloging across channels
- Complex product and bundled pricing across channels
- Complex order orchestration
- Integrated supply chain visibility

Case in Point: Sterling Commerce

It's important to consider whether a company is equipped for SOA-based integration when seeking to acquire them. Starting in 2005, Sterling Commerce set out to acquire supply chain capabilities and multi-channel selling capabilities to sell on top of its own SOA-based offerings. But unlike other competitors, Sterling only pursued acquisitions that met its SOA strategy. When acquired, each of these companies was SOA-based and so interoperated easily with Sterling's existing system to deliver the improved products. With the addition of Comergent, Sterling was able to deliver a comprehensive end-to-end order management solution that will help companies increase revenue and reduce costs and order fulfillment lead times. Adding Nistevo enabled Sterling to provide customers with a complete state-of-the-art supply chain management system from a single vendor that improves visibility and control over the entire supply chain – from order to shipment. The acquisition of Yantra let Sterling Commerce deliver supply chain products that already had high market acceptance.

–CONTINUED ON PAGE 23

Solving Real-World Business Problems with SOA

A case study: The wealth management industry

WRITTEN BY **MARK EATON**

The financial services industry is well suited to application solutions based on Service Oriented Architecture (SOA) due to the heterogeneity across front-, mid-, and back-office systems and different business lines.

Many financial services firms, particularly those that have undergone mergers or acquisitions, can benefit from SOA as they are burdened with legacy data silos, redundant applications, overlapping functionality, brittle proprietary systems, and steep integration costs.

As financial services firms investigate the merits of SOA adoption, wealth management stands out as an example where SOA is having a positive effect. The wealth management business, including banks, broker-dealers, and insurance companies that service high net worth (HNW) clients, has demonstrated how SOA can improve firm efficiencies through improved advisory productivity, decreased IT costs, and increased revenues.

The Wealth Management Challenge

For a wealth services firm to differentiate itself from its competitors, it needs to improve client service while at the same time growing new business. On both counts, wealth advisors face several information challenges. In the last two years, the time advisors spend on administrative tasks has increased by 31%, according to the BusinessEdge research group, reducing their client-facing time for acquisition and retention.

Administrative time has increased for several reasons. First, the number and complexity of investment products available to HNW investors has expanded dramatically in the past few years, complicating the advisor's task of getting clients invested. Second, compliance burdens have grown, requiring advisors to document and maintain additional client information. And third, many wealth management firms haven't been able to offer their advisors streamlined and automated systems to simplify and reduce their administrative tasks.

To do their jobs, advisors require continuous access to a wide variety of information and tools, including up-to-date client data, multiple disparate legacy applications, decentralized or non-existent investment product catalogs, indicative data, and compliance documentation. To conform to client-directed investment policies and compliance regulations, client data needs to be housed in a centralized client profile to eliminate potential errors due to re-keying data at various points in the advisory workflow process.

Wealth managers, many of whom are licensed to sell only certain investment vehicles, also need an integrated desktop environment with embedded best practices to ensure that the advisory team operates within the client investment constraints and compliance environment. Multiple disparate applications can't provide this holistic and conformant approach to the wealth management discipline.

An SOA Approach

SOA dramatically simplifies the wealth management challenge. The wealth management discipline is primarily a front-office endeavor, so the challenge is consolidating information across multiple, disparate mid- and back-office applications and data silos. Financial advisors demand a single consolidated view of all client and product information as well as application workflows that use this consolidated data view.

There are two options to provide this consolidated view. The data can be replicated into a single database, an expensive and time-consuming endeavor, or firms can build their data model around SOA, which allows mapping individual portions of the data model directly onto back-office systems. SOA leverages data from existing systems of record to avoid data replication and discrepancies. For example, CRM-stored information such as client information, client service events, entitlements, and security policies can be seamlessly passed into advisory workflows.

Another benefit of SOA in wealth management is distribution. Distribution in SOA takes a portion of the data model and exposes it to more loosely coupled systems in the enterprise. SOA provides an enabling technology for consolidation of data, essential for the wealth management discipline, and provides a mechanism for distribution, making for a more open enterprise.

SOA also allows firms to purchase and deploy only the modules they need, rather than buying a complete system that may replicate data and applications they already have. The isolation of specific application functionality into services provides the ability to deploy not only the portion of the desktop appropriate for the client, but also the portion of the underlying application architecture needed for the desktop.

The multi-service integration framework of SOA can integrate all types of services into the advisory workflow – from granular services like calculators to full-blown investment planning applications. This lets companies leverage their own front-office portal with applications and services already purchased or built in-house.

For a wealth management firm, SOA can drastically reduce integration costs and have a direct impact on revenue generation. For instance, an XML import tool can integrate third-party wealth management products and schemas, creating a company-wide product catalog that expands the breadth of product information available to advisors, enhancing their ability to generate new business.

SOA for Client Services

Before service-oriented wealth management platforms existed, firms had to invest heavily in middleware to link back-office data and applications such as portfolio accounting systems, security masters, and ledger systems. Implementing middleware was complicated, very expensive, and challenging to maintain. Moreover, middleware didn't adequately address the growing number of unintegrated point applications that were springing up in the front office to serve financial advisors' needs.

The SOA-enabled enterprise eliminates redundant point applications and unnecessary middleware by leveraging existing systems and administrative staff. Moreover, the flexibility of the SOA-enabled enterprise allows for a flexible IT roadmap where legacy systems are systematically decommissioned, and data and application silos are opened up to other uses. As industry regulations continue to evolve, SOA is a flexible architecture more adaptive than a conventional architecture to new and changing compliance requirements.

SOA technology offers the promise of transforming how the front-, middle-, and back-office systems are connected. Advisors no longer need to access multiple separate applications. SOA facilitates a unified advisory desktop that reduces advisors' administrative time so they can spend more of it acquiring and servicing clients. In this way, SOA technology is a powerful and essential tool for transforming the wealth management business overall.

For the wealth advisor and client-service professionals in all industries, SOA means better customer service through increased transparency and better consolidation of data and functionality.

About the Author

Mark Eaton is vice-president of engineering at NorthStar, a provider of wealth management software for financial services institutions

Multi-Enterprise SOA

–CONTINUED FROM PAGE 21

And we're drinking our own champagne because to enable cross-channel selling of all of our built and bought products, we're in the process of implementing our own SOA-based multi-channel selling solution right on top of our existing systems. By deploying this we can offer all of our acquired products alongside our existing product lines through one customer and partner interface. We can quickly launch new prices and bundles in the marketplace to accelerate sales of existing and new products through all of our channels.

SOA: A Framework for Post-Acquisition Renovation

SOA-based business-to-business integration lets companies create, manage, and deploy any business service across any channel – inside or outside enterprise boundaries – and assist with all the growth strategies mentioned above. By first applying an SOA for interfaces outside companies, two merging businesses can provide a unified front to external parties and begin harvesting opportunities for growth long before welding together internal systems and processes.

When companies are considering SOA strategies, they should start with revenue-producing projects that connect to partners, suppliers, and customers. Focusing on growth maximizes shareholder value. Companies should apply integration to one business problem at a time – but they can shorten the time it takes to do this by using an SOA strategy. Applied thoughtfully and incrementally, SOA helps companies successfully complete post-merger integrations by allowing systems to interoperate without restructuring the enterprises or replacing the underlying infrastructure.

Enterprises have complex, interwoven underlying IT systems similar to the utility, transportation, and plumbing systems of a city. SOA is a framework for renovating enterprise systems so that, when they merge, they can be quickly and cost-effectively rerouted, combined, and reallocated without having to weld code together haphazardly or dig up hardware, rip it out, and replace it. SOA is the key to achieving rapid interoperability, which keeps critical systems such as sales and customer service running and optimizes the deployment of existing assets toward priority growth opportunities.

Custom Integration No More

With SOA, internal systems and processes may never need to be manually integrated again. At least, not in the way they are today – with an army of developers connecting disparate legacy systems with custom code that breaks with each application or server upgrade. At the same time, with SOA, old business processes need not be maintained when they have outlived their usefulness. Implementing SOA can eliminate the need to change old systems or processes, while making the implementation of new systems and processes faster and cheaper. This is especially important during an acquisition, when systems and processes are scrutinized for overlap and compatibility between the two organizations.

In this way, SOA can be the best framework for renovating enterprise systems. By using SOA to decrease the time it takes to develop new systems or integrate existing systems, executives can focus on streamlining the communications between companies to ensure that critical information is being shared quickly and accurately among their communities of partners, suppliers, and customers.

Do's and Don'ts of Post-Acquisition SOA

Here are some do's and don'ts for putting SOA to work to optimize the acquisition integration outcome and keep the business focused on growth:

1. DO focus SOA projects on revenue acceleration
2. DON'T use SOA merely to integrate systems to reduce IT costs
3. DO interoperate and integrate internal processes through SOA
4. DON'T just focus inside the enterprise. Find a process that links to customers, distributors, partners, and suppliers.

Ultimately, companies can leverage SOA to take an incremental approach to M&A integration that puts energy and resources behind growth – not cost containment or back-end wiring – while laying a foundation that will let the enterprises adapt to ever-present M&A activity in the global economy. The key is to use SOA first to facilitate external processes that enable growth at acceptable margins.

About the Author

Joel Reed is vice president, global product marketing for Sterling Commerce, a subsidiary of AT&T. Sterling Commerce provides enterprise integration, multi-enterprise integration, multi-channel selling, and multi-channel fulfillment to more than 30,000 customers worldwide.

Testing SOA Solutions

What's different and how to handle it

WRITTEN BY CHRIS HARDING, CHUCK SHRIVER, R. B. HUANG, AND TONY CARRATO

Service Oriented Architecture (SOA) has been discussed as an important architectural style for the last few years. Organizations have started to develop service-oriented solutions and many are now leveraging services in their production environments.

SOA introduces new technical complexities and challenges and makes testing a critical component of the development lifecycle. Teams need to think about:

- How do you know if the solution is ready?
- How do you know that it will scale to accommodate future needs?
- How do you know that you can actually get the business flexibility that SOA promises?

These are all questions testing should answer, but many test teams aren't experienced yet in testing service-oriented solutions.

This article contains a set of recommendations, with a rationale, that will help you to mitigate the issues that arise in testing an SOA solution. The recommendations are based on experience gained over the last three years through involvement in a number of SOA projects.

The recommendations form the main part of the article, but first we'll provide an overview of what SOA is and what challenges the test team must consider.

SOA Solutions

To think about testing SOA solutions, it's important to start with a clear idea of how a SOA solution is constructed. This is illustrated in the Figure 1, which is taken from the SOA reference architecture developed by IBM and used by The Open Group as the basis for an open standard SOA reference architecture.

Typically, a SOA solution consists of a set of services that are orchestrated into business processes using capabilities provided by middleware. Each service may be composed of other services or may be atomic.

The services are invoked by service consumers such as portlets in a portal or external applications in a business to business (B2B) solution. The individual services are likely to be called in some ordered sequence, probably involving decisions (branching) and going back to previous steps (looping), with a good chance that there will be times when handing a task to a person and then waiting for a result is required.

Services are actually facades for underlying IT assets, which may

be part of existing (pre-SOA) operational systems, or may be specially created service components. These components provide the implementation or "realization" for the services, and may be newly created or may provide access to capabilities of existing operational systems. (It's often better to insert these service components between the services and the existing systems than to access the existing systems from the services directly.)

Besides the services and their implementations, an SOA solution also contains functionality relating to service and component integration, quality of service, information management, and governance.

Testing SOA Solutions

The goals of testing remain the same as for non-SOA solutions, but SOA testing requires that testers add to their skills. It also requires changes to how testers approach testing and what tools are used. There are a number of characteristics of SOA solutions that affect the nature of testing, at all levels. Besides the traditional levels of unit test, integration test, etc., SOA testing requires testing at two new levels we've not previously had to think about:

- Testing of services, the facades in front of provider components.
- Testing of orchestrated processes, invoking many services.

What else is special about testing service-oriented solutions? Here are the main characteristics that make testing different for SOA:

- Services can be independently developed, changed frequently, intended to be reusable but must still work together to provide the total solution. A service may have been unit-tested and work on its own but, when it's added to the larger system, its interaction with other components poses additional integration testing challenges.
- Services can be "GUI-less" and testers require a mechanism to invoke and test the service without a traditional user interface.
- Services can be implemented using different technologies, which provide performance challenges.
- Packaged services can provide unique challenges since they've been created by a third party, and the team doesn't have total control over the design or implementation.
- Due to the way a SOA solution is composed from various services, meeting quality of service requirements and problem determination require special consideration.
- Because multiple services on heterogeneous platforms must work together, standards become critical.
- SOA testers must have knowledge of multiple standards and technologies.
- Many more combinations of use of services will be possible than the team can test. Assuming finite resources, resource/test coverage decisions have to be made. Risk-managed testing should be used.
- Since services are also intended to be discoverable, they are likely to be used in processes and by service consumers not contemplated when the service was originally designed. Hence the service must be fully encapsulated and do no more or less than advertised.
- Performance characteristics must be carefully considered. If a service was originally designed for 100 invocations a second, with an initial load of 10 invocations a second, what happens when other consumers discover and try to use the service? In particular, when the load exceeds the design goal, is there a mechanism for managing the overload gracefully?
- Service providers are likely to have their own security models, with different rules for access and use. The test manager must decide how to confirm that only appropriate access to the services underlying a given business process is handled.

> "Services are actually facades for underlying IT assets"

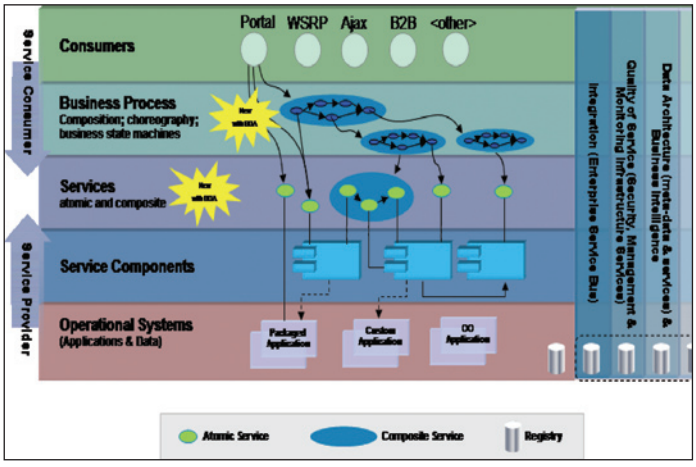


Figure 1 A SOA solution stack

Each of these aspects exists in software systems today posing a significant challenge to test teams. What makes SOA unique is both the degree to which they exist and the challenge in dealing with all of these features in most, if not all, SOA projects. The recommendations are on the following page.

The Recommendations

Leveraging experience in a number of SOA projects done over the last three years, we have developed a number of suggestions for the SOA test team. These suggestions aren't intended to replace good test design, development, and execution. Instead, they're intended to supplement what the test team would be doing in any other large complex project. The recommendations are on the following page.

Conclusion

We hope that we've demonstrated some important issues in SOA testing, highlighting areas where testing SOA solutions bring new challenges. We also hope that our recommendations provide useful guidance for you if you are testing a service-oriented solution for the first time. We would welcome your feedback on this article and your suggestions for improving the recommendations for SOA testing.

Recommendation	Rationale
Establish the test environment(s) early	Merely testing the delivery of the service's capability (function testing) isn't sufficient. SOA solutions add additional software layers. These typically include: • An Enterprise Service Bus (ESB) • A process orchestration engine They may also include an external security subsystem, a portal subsystem, a Web Services gateway (to outside service providers), and a master data subsystem. It is vital to baseline the test environment and ensure that testing the solution doesn't turn into testing the test environment. Setting up and validating the test environment early will minimize this problem.
Plan for incremental testing	SOA solutions are based on services and service orchestrations. A waterfall approach to development invites disaster, with problems discovered very late in the project timeline. It is imperative to test new services and updated orchestrations as they become available to identify problems early, so that they can be rectified and the lessons passed back to the development teams, to improve their processes and approaches.
Test at least one or two end-to-end services early in the project	Once the test environment is up and available, it needs to be validated. The best way to do this is to test one or two (more is usually not necessary) services that touch all the major elements of the environment. This will let the tester confirm that all parts of the test environment and, by extension, the planned production environment are working together correctly and that no false assumptions have been made about the environment.
Plan for mock services as stubs to allow testing other parts of the solution	Just as various elements of a classical solution will be stubbed out to allow early testing, service-oriented solutions should make use of stubs to enable early testing. Be sure to track these stubs and their removal as you move toward production.
Have an enablement plan for the test team	SOA solutions are likely to require new skills and knowledge not just on the part of the architects and developers, but on the part of the test team as well. Plan for learning; don't just hope it will happen.
Use tools to support the testing process	Testing SOA solutions is complex. As more services are added to the solution under test or as business processes orchestrations are built, many test cases are likely to be needed. Using tools to support the testing process improves efficiency. It also reduces testing errors, which improves the quality of the testing results.
Validate the service model early	The service model, which is developed by business analysts and solution architects, is the heart of the SOA solution. Validating the model early is critical to ensuring that the right solution is built.
Develop a performance model for the solution early, including performance budgets for the services	Because they involve many software layers and orchestration of services into processes, it's easy for the performance of a service-oriented solution to suffer. Modeling performance against requirements early in the solution design process will highlight areas where special attention is required. In addition, performance budgets (how long a given activity is allowed to take) for the elements of the solution will enable developers and testers to focus attention on problem areas, the ones exceeding their budgets, rather than spreading attention of performance engineers across the entire solution.
Test for Quality of Service targets	Quality of Service targets for the SOA solution can be tested by using system management software that will be part of the SOA solution.
Take a risk-based approach to testing	Like all other activities in a large project, the testing activity probably won't have unlimited time or resources. This means that testing attention must be focused on the most important parts of the solution first. The best way to do that is to evaluate risks in the solution in terms of complexity, the impact of failures, the frequency of invocation, etc. Based on a risk evaluation, the highest priority areas can be tested first.
Design service tests without worrying about the underlying technical implementation details	Services are facades over the provider IT components. A tenet of a SOA solution is that these components may change but the service consumers are insulated from those changes. Testing services without assuming a particular implementation will help validate that this is true.
Test services for reusability	Services are intended to be discovered and able to used by multiple consumers. Consequently, the tester should be concerned about validating that a service can be reused.
Don't forget testing external services	Many service-oriented solutions use services from one or more outside organizations. Testing these services early, to confirm that they behave as described, will reduce uncertainty and errors in the project development process.
Think process testing, not just service testing	Service-oriented solutions bring the questions of process design and development into IT projects in a bigger way than more traditional solutions have. This means that the processes, not just the underlying services and IT components, will have to be tested. Testing should be done using process models, describing the planned processes as guidance for the behavior of the system. Many process modeling tools provide simulation capabilities that can be used to inform and support the testing processes
Think mediation testing plus service/process testing	The capability of mediation inside an ESB plays an important role in SOA solutions. This means the testing has to verify that the mediation function works properly. Testers have to verify that functions such as dynamic selection and transformation work well in all kinds of situations.

References

- Bryson, Brian et al. "Spotlight on IBM software solutions: Quality management for Web services-based applications." http://www.ibm.com/developerworks/podcast/spotlight/st-071707txt.html.
- IBM SOA Testing Forum. http://www.ibm.com/developerworks/forums/dw_forum.jsp?forum=1091&start=0&thRange=15&cat=24.
- Linthicum, David S. "Adjusting Testing for SOA." http://www.sdtimes.com/article/special-20070815-01.html.
- Rose, Laura et al. "Software Testing in SOA." Internally available within IBM.
- SOA Testing Blog. http://soa-testing.blogspot.com/
- Joines, Stacy. "SOA Performance Best Practices." https://www-926.ibm.com/events/IBMImpact/ii07cms.nsf/download/kcfb95cea53f8fa8f112fd2129c4/$FILE/4246%20SOA%20Performance%20Best%20Practices.dist.pdf

Acknowledgements

This article owes a debt to important work by multiple IBM teams:

- SOA Design Center in Beijing
- Quality Software Engineering, within the IBM Software Group
- IBM Research's testing efforts
- IBM SOA Advanced Technology Quality & Risk Management Team.

About the Authors

Dr. Chris Harding is the Forum Director for SOA and Semantic Interoperability at The Open Group - an open forum of customers and suppliers of IT products and services. He has been with The Open Group for over ten years.
c.harding@opengroup.org

Chuck Shriver is a test architect for the SOA Advanced Technology team in IBM's Software Group, focusing on SOA test strategy. In this role, he works with a variety of organizations within IBM who help IBM clients implement SOA projects.
shriverc@us.ibm.com

Huang Ruobo is advisory software development manager for the China SOA Design Center of China Software Development Lab in IBM's Software Group, focusing on SOA tooling development and delivery.
huangrb@cn.ibm.com

Tony Carrato is the worldwide chief operations architect for the SOA Advanced Technology team in IBM's Software Group, focusing on SOA delivery. In this role, he is responsible for a team of IT architects who help IBM clients define and implement SOA projects.
acarrato@au1.ibm.com

STATEMENT OF OWNERSHIP MANAGEMENT AND CIRCULATION (2004)
1. Publication Title: WebServices Journal; 2. Publication no.: 1535-6906
3. Filing date: 10-01-2007; 4. Frequency of issue: monthly: 12; Annual subscription price: $69.99;
7. Complete mailing address of the headquarters of general business offices of the publisher: 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677; 9. Full names and complete mailing address of publisher: Fuat Kircaali, 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677; 10: Owner: Fuat Kircaali, 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677. 11. Known bondholders, mortgages, and other security holders owning or holding 1 percent or more of total amount of bonds, mortgages, or other securities: none.
13. Publication name: WebServices Journal
14. Issue date for circulation data: September 2004 – Volume 14 #9
15. Extent and nature of circulation: (a) Average no. copies each issue during preceding 12 months: 3179; Total number of copies (Net press run): 9146; (b) Paid and/or requested circulation: (1) Individual paid/requested mail subscribers: 5583; (3) Sales through dealers and carriers, street vendors, counter sales and other paid or requested distribution outside USPS: 2646; (c) Total paid and/or requested circulation: 8229; (e) Total nonrequested distribution: 617; (f) Total distribution: 8846; (g) Copies not distributed: 300; (h) Total: 9146; Percent paid and/or requested: 93.
16. This Statement of Ownership will be printed in the October issue of this publication.
17. I certify that the statements made by me above are correct and complete.
Fuat Kircaali
Publisher

STATEMENT OF OWNERSHIP MANAGEMENT AND CIRCULATION (2005)
1. Publication Title: WebServices Journal; 2. Publication no.: 1535-6906
3. Filing date: 10-01-2007; 4. Frequency of issue: monthly: 12; Annual subscription price: $69.99;
7. Complete mailing address of the headquarters of general business offices of the publisher: 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677; 9. Full names and complete mailing address of publisher: Fuat Kircaali, 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677; 10: Owner: Fuat Kircaali, 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677. 11. Known bondholders, mortgages, and other security holders owning or holding 1 percent or more of total amount of bonds, mortgages, or other securities: none.
13. Publication name: WebServices Journal – Volume 5 #9
14. Issue date for circulation data: September 2005
15. Extent and nature of circulation: (a) Average no. copies each issue during preceding 12 months: 4217; Total no. of copies (Net press run) 4200. (b) Paid and/or requested circulation; (1) Individual paid/requested mail subscribers: 2087; (3) Sales through dealers and carriers, street vendors, counter sales and other paid or requested distribution outside USPS: 1511; (c) Total paid and/or requested circulation: 3598; (e) Total nonrequested distribution: 204; (f) Total distribution: 3802; (g) Copies not distributed: 398; (h) Total: 4200; Percent paid and/or requested: 95.
16. This Statement of Ownership will be printed in the October issue of this publication.
17. I certify that the statements made by me above are correct and complete.
Fuat Kircaali
Publisher

STATEMENT OF OWNERSHIP MANAGEMENT AND CIRCULATION (2006)
1. Publication Title: SOA WebServices Journal; 2. Publication no.: 1535-6906
3. Filing date: 10-01-2007; 4. Frequency of issue: monthly: 12; Annual subscription price: $69.99;
7. Complete mailing address of the headquarters of general business offices of the publisher: 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677; 9. Full names and complete mailing address of publisher: Fuat Kircaali, 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677; 10: Owner: Fuat Kircaali, 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677. 11. Known bondholders, mortgages, and other security holders owning or holding 1 percent or more of total amount of bonds, mortgages, or other securities: none.
13. Publication name: SOA WebServices Journal
14. Issue date for circulation data: September 2006 – Volume 6 #9
15. Extent and nature of circulation: (a) Average no. copies each issue during preceding 12 months: 3505; Total no. of copies (Net press run): 2500. (b) Paid and/or requested circulation: (1) Individual paid/requested mail subscribers: 1085; (3) Sales through dealers and carriers, street vendors, counter sales and other paid or requested distribution outside USPS: 961; (c) Total paid and/or requested circulation: 2046; (e) Total nonrequested distribution: none; (f) Total distribution: 2046; (g) Copies not distributed: 454; (h) Total: 2500; Percent paid and/or requested: 100.
16. This Statement of Ownership will be printed in the October issue of this publication.
17. I certify that the statements made by me above are correct and complete.
Fuat Kircaali
Publisher

STATEMENT OF OWNERSHIP MANAGEMENT AND CIRCULATION (2007)
1. Publication Title: SOA World Magazine; 2. Publication no.: 1535-6906
3. Filing date: 10-01-2007; 4. Frequency of issue: monthly: 12; Annual subscription price: $69.99;
7. Complete mailing address of the headquarters of general business offices of the publisher: 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677; 9. Full names and complete mailing address of publisher: Fuat Kircaali, 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677; 10: Owner: Fuat Kircaali, 577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677. 11. Known bondholders, mortgages, and other security holders owning or holding 1 percent or more of total amount of bonds, mortgages, or other securities: none.
13. Publication name: SOA World Magazine
14. Issue date for circulation data: September 2007 – Volume 7 #9
15. Extent and nature of circulation: (a) Average no. of copies each issue during preceding 12 months: 2066; Total no. of copies (Net press run): 2000; (b) Paid and/or requested circulation: (1) Individual paid/requested mail subscribers: 697; (3) Sales through dealers and carriers, street vendors, counter sales and other paid or requested distribution outside USPS: 675; (c) Total paid and/or requested circulation: 1372; (e) Total nonrequested distribution: 500; (f) Total distribution: 1872; (g) Copies not distributed: 128; (h) Total: 2000; Percent paid and/or requested: 73.
16. This Statement of Ownership will be printed in the October issue of this publication.
17. I certify that the statements made by me above are correct and complete.
Fuat Kircaali
Publisher

Leveraging SOA in Communications-enabled Business Processes

The additional enabler, policy-based, real-time session management

WRITTEN BY **ROD HODGMAN**

Covergence is the creator of Eclipse, an award-winning series of next-generation session border control solutions. Recently, the company unveiled the Covergence Policy Manager, a policy enforcement solution for real-time applications.

Using Policy Manager, organizations can now build session policy enforcement into applications that include VoIP, video, IM, presence, conferencing, shared workspaces, and other forms of real-time communications. This case study details how one major financial institution leveraged the Covergence Policy Manager to improve communications between customers and the sales force by integrating call control and call policy into its CRM system.

Enterprises and service providers alike are entering a new era of communications-enabled applications (CEA). By integrating business applications with real-time communications, CEA streamlines many business activities and helps companies serve their customers more responsively. The movement to CEA is enabled by two major trends: The growing popularity of Service Oriented Architecture (SOA) for business applications and the widespread adoption of SIP-based real-time communications, including voice, video, and presence. But an additional enabler is needed to make CEA a secure, cost-effective reality: SOA-compliant, policy-based, real-time session management.

Like any IP application, VoIP and other real-time services must be secured, controlled, and managed. However, real-time applications pose a unique challenge in that voice, video, and other real-time media are highly sensitive to the latency, jitter, and packet loss that are common in IP networks. Unlike e-mail or Web browsing, where a few seconds' delay or an occasional resend go unnoticed, a small delay or the loss of a few packets can make an audio or video session unusable. Therefore, real-time session management solutions are designed specifically to execute policy functions without degrading performance or compromising quality.

While CEA and dynamic real-time session management may sound "visionary," enterprises and service are implementing these solutions today. This case study examines how one major financial institution recognized that the efficiency of communications between customers and the sales force could be improved significantly by integrating voice capability and call policy into its CRM system to improve the efficiently of the selling process. Working with Covergence, this company, which operates in a highly regulated environment, was able to reap the benefits of CEA.

The CEA Challenge

There are two challenges that must be addressed to deploy communications-enabled applications successfully. The first is to understand the need for and the role of the session management function. As shown below in Table 1, the session management function must provide comprehensive security to protect against attacks/intrusions and to ensure the confidentiality, integrity, and authenticity of communications. It also must ensure that the organization has complete control over all communications to enable compliance with internal policies or external regulations. It also

Connectivity and Interoperability	Identity-based access control
Multi-vendor protocol normalization	File transfer control
SIP-aware NAT traversal	Instant message content control
Corporate directory integration	URL access control
Protocol repair and CODEC transcoding	
	Monitoring
Security	Session detail recording
Cryptographic authentication	QoS detail recording
Signaling and media encryption	Instant message recording
Stateful signaling and media validation	Voice and video recording
Denial-of-service attack mitigation	File transfer recording
Intrusion prevention	System and administrative event logging
Virus scanning	
Malicious URL filtering	**Application-aware session routing**
	Policy-based call routing
Control	Configurable routing policies
Signaling and media control	Multiple routing constraints cost, quality, precedence, bandwidth, etc.
Quality of service (QoS) control	

Table 1: Session management features

provides the monitoring and management capability necessary to ensure "toll-quality" service all the time. Finally, the session management function must provide the intelligence so the organization can set routing polices based on cost, quality, class of service, or other business rules. In short the session management function enables the organization to secure, manage, and control all real-time sessions.

After understanding the requirements for session management, the next challenge was to understand how to make it possible for developers to build real-time policy into their applications. For instance, in a contact center environment the session manager would have the information needed to make call routing decisions based on the cost or congestion of a particular route. But the business logic determines where a call should be routed based on the customer's need and the location of the support agent with the correct skill set. In this case the correct route is selected only when session routing policy is influenced by the business logic contained at the application layer.

The answer of course is to ensure that the session manager solution provides a standards-based interface that lets developers exert fine-grained control over session logic from the business logic. Most of today's business applications, whether built for the public Internet or for enterprise intranets, rely on a Web-like model — centrally hosted applications interacting with distributed user devices via the Hypertext Transfer Protocol (HTTP). Moreover, most of these applications are based on a layered SOA.

But to extend the benefits of SOA and Web Services to real-time applications and facilitate the implementation of CEA, a session management layer is needed (see Figure 1). The session management layer combines session border control with application-level security, session routing, and session management to create single point of policy enforcement for real-time applications and services.

Building Real-time Policy into CRM with Web Services

By adding a Web Services interface to the session management layer, developers get the ability to exert dynamic, fine-grained, policy-based control over real-time communications (see Figure 2). For instance, they can specify that certain VoIP calls should follow least-cost routes while others are routed to maximize quality. Acting as a policy gateway, the session management layer also assures compliance with security, regulatory, and business policies. To satisfy Sarbanes-Oxley, for example, the session management layer could log all inter-enterprise instant messages. To control costs, it could block international phone calls by unauthorized users.

With this particular customer, the objective was to improve the sales process by embedding voice call and control into its CRM application. Working with Covergence, the enterprise created a CEA solution that functions as follows:

a, The broker could come out of a morning meeting briefing with new research or information about certain stocks that he wanted to share with his customers
b. He could log into the CRM system to pull up all of his customers who are interested in that stock
c. The broker would then click-to-call them from the CRM system

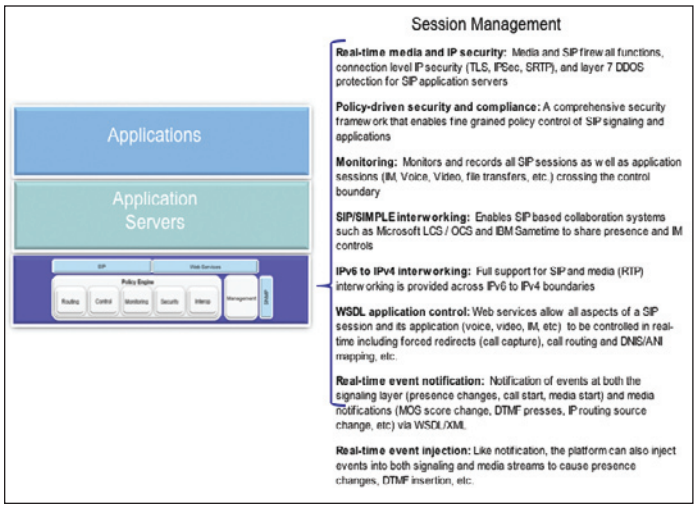


Figure 1: A session management layer must be added to today's SOA architecture to facilitate CEA.

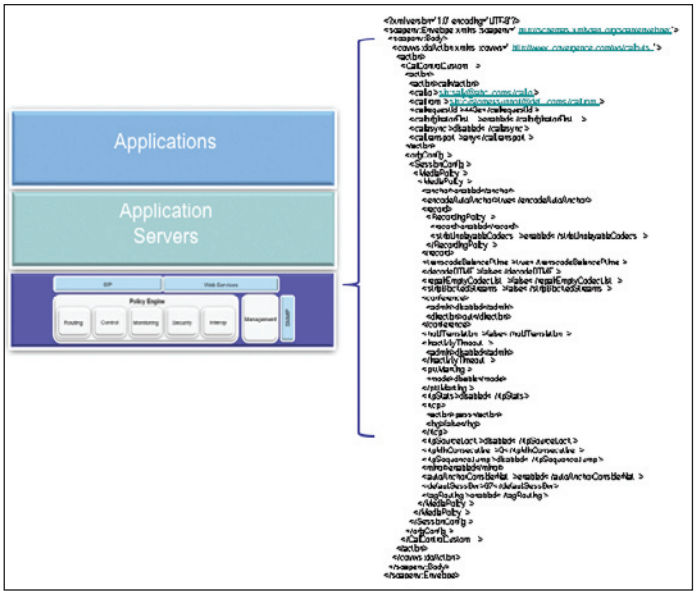


Figure 2: Web Services Code Snippet to Start a Call

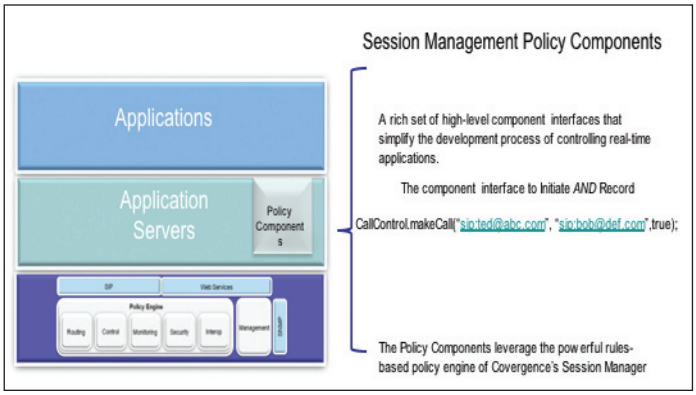


Figure 3: Session management policy components make it easier for the developer to embed real-time policy into applications

d. If his call went to the customer's voice mail the broker could click another button to have a customized voice mail delivered. The broker is then ready to call the next customer
e. If the customer takes the call the broker could speak with him while all the relevant information about the customer's holdings are displayed from the CRM system

In this use case, the process is optimized because the broker only spends time speaking with customers, not leaving voice mail. Also his conversations with the customer are contextualized because he is looking at all the information he has on both the customer and his recommendations.

With robust session management accessible via familiar SOA component interfaces, the company was able to implement CEA quickly and cost-effectively.

Evolving Web Services to Java Components

Over time it became apparent that certain combinations of Web Services could be grouped together into Java components and that this would make it even easier for developers to embed policy into applications. Components are self-contained, reusable software units that can be customized easily and assembled into composite components.

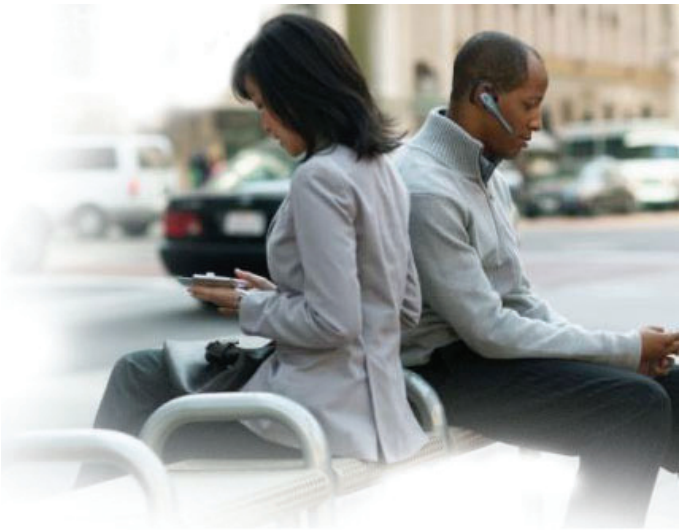
Moving to a component model provided two key benefits. The first benefit was that the component model made it even easier for developers by insulating them from the policy model expressed in the XML document shown in Figure 2. In Figure 3 we present the new component interface used both to initiate and record a voice call. You can readily see how one line of code (CallControl.makeCall ["sip:ted@abc.com", "sip:bob@def.com",true]) is much easier to understand conceptually than a hundred lines of XML. So components make session management more approachable for developers by shielding them from the internal details of session management.

Another benefit of moving to the component model is the increased extensibility and customization that the model brings to developers. Components use the properties, methods, and events model. Properties can be thought of as an object's attributes (the set of things that describe the object), methods as its actions or what the object can do for you, and events as its response or the code that's executed when a certain situation or event occurs.

"Real-time applications pose a unique challenge because voice, video, and other real-time media are highly sensitive to the latency, jitter, and packet loss that are common in IP networks"

Components are dynamic in that they can be readily changed or customized. Through the design mode of a builder tool the developer can modify the properties of the component to customize it. They can also modify its appearance and behavior, define its interaction with other components, and combine it and other components to create a new composite component.

By creating new powerful components that aggregate together some of the more frequently used Web Service calls the company's application programmers don't have to learn the details of session management or debug XML code. They can set dynamic real-time policies by writing a line of Java instead of hundreds lines of XML.



The Result

Ultimately, the institution discovered CEA implementation. Making session management an integral part of the application infrastructure gives business applications dynamic, fine-grained control over real-time communications. Basing session management on the SOA model minimizes the time and cost required to implement CEA. And building in stringent policy enforcement ensures that all real-time activities comply with security, regulatory, and business policies.

As this case study illustrates, by integrating real-time communication with business applications, CEA can boost employee productivity and reduce operating costs simultaneously. It can also make companies more responsive to ever-changing business and customer requirements. The best way to implement CEA is to make real-time session management part of a company's existing SOA/Web Services architecture. By adding a robust session management layer and making it accessible to applications via familiar Web Service protocols or through customizable components companies can leverage their developers' existing skill sets and make a quick, cost-effective transition to CEA. ■

About the Author

Rod Hodgman is vice president of marketing at Covergence. He was a member of the Covergence founding team and is responsible for positioning the company's session management solutions for maximum value and growth. Rod brings over 25 years of related business experience in technology start-ups and large companies. Prior to joining Covergence, Rod was the VP product marketing, Enterprise Products at Macromedia where he was responsible for the definition and introduction of Flex.

rod.hodgman@authors.sys-con.com

Strategies for Versioning Web Service

– CONTINUED FROM PAGE 6

Which of these approaches should to be applied to a given service architecture is purely a function of the application and the business use case. In complex enterprise environments, the extremes of the scale – i.e., handling all versions or not handing any versions implicitly – limit the value of the system and adversely affect coupling, manageability, and extensibility and affect the long-term return on investment. For example, with the Big Bang scheme, since S_1 is coupled to C_1 any change to it will ripple through all the consumers or require the development and deployment of a new S_2 service that now can only be consumed by consumers C_2. The backward- or forward-compatible schemes and the change associated with them provide the most value and flexibility for a SOA.

To decompose the problem further, let's distill the meaning of this change. Given a Web Service contract, change can be incremental, decremental, or transparent.

- ***Incremental change:*** Involves adding – new optional headers, operations, input or output messages (as long as they result in a new operation), new schema types or adding to existing types, or introducing new policy assertions.
- ***Decremental change:*** Involves removing or renaming required headers, operations, input or output messages, or their order from operations, schema types, structures from existing types, or removing policy assertions.
- ***Transparent change:*** There's no change in the interface or contract with the consumer however the service implementation or behavior is changed.

In general incremental changes are backward-compatible and decremental changes aren't. Deprecating parts of a service contract is different from decrementing it – decremental changes result in breaking backward-compatibility and have to be addressed out-of-band. Compatible changes would be the same as those applicable to a programming language. For example, the Java Serialization specification lists compatible changes as changing the implementation of a method, adding new methods, adding new instance variables and incompatible changes such as changing the type of a variable or moving a class in an inheritance hierarchy.

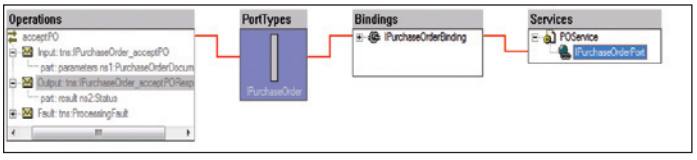


Figure 1

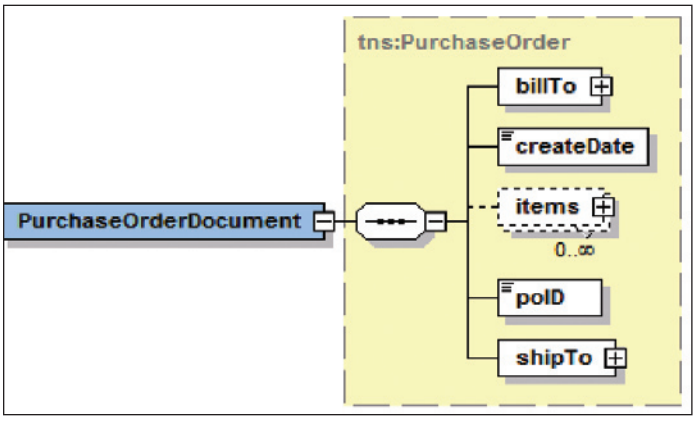


Figure 2 Sample service

From an architect's perspective, given incremental or transparent change, backward/forward-compatible versioning schemes for service implementation and deployment can be broadly be put into two main categories that can be characterized:

- Invocation context-based versioning
- Invocation content-based versioning

We'll examine these from the service producer and service consumer perspectives.

Invocation Context-based Strategies

These strategies rely on discriminating versions of Web Services based on the context under which the services are invoked. An invocation context can be construed either from a deployment context, physical endpoint location, or combination of both.

The advantages of context-based strategies include flexibility to introduce intermediaries and additional layers of value-added services like content-based routing, the flexibility to provide separate infrastructures to manage scalability, reliability, and availability, or introduce network-based SOA appliances.

Implementation Scenario

From a service consumer's perspective the different versions are:

- Version S_1 of a service is available at location L_1 (http://host:port/myservice/v1)
- Version S_2 of the same is available at location L_2 (http://host:port/myservice/v2)

Or some other variation of this scheme – for example:

- Version S1 of a service is available at location L_1 (http://host:port1/myservice)
- Version S2 of the same is available at location L_2 (http://host:port2 /myservice)

Both these strategies rely on deriving the context based on combinations of host, port, and URL pattern in the endpoint address. The latter separates version concerns by deploying on a different host or opening additional ports. This is generally suitable when a version of the code is being completely retired, maintained by different groups/teams, or has significant underlying dependency differences. It may also be suitable for certain containers where the deployment model for EAR files is port-based.

From a service developer's perspective both these service endpoints can be implemented by the same class or a class hierarchy if the operations are being overloaded (effectively introducing incremental change). However, separating the endpoints allows the implementation to determine the consumer version at runtime by deriving it either implicitly or explicitly from the invocation context.

For example, a document-based Web Service that accepts documents based on a purchase order schema and returns a document based on the status schema is shown below in Figure 2 and the relationship between the schemas is shown in Figure 3.

This service S_1 is built and deployed at http://localhost:8080/ver-

sionop1/v1. A new version of the service now needs to have additional operations and handle orders from different customer levels (gold/silver etc.). These incremental changes are introduced into S_1 in the form of a new operation and data types as shown in Figure 4, and the resulting service S_2 is deployed at http://localhost:8080/versionop2/v2. The schemas remain in the same namespace. Consumers C_1 can use S_1 or S_2, if they are aware of the new service. C_2 can use either S_1 or S_2 for existing functionality and only S_2 for new functionality

Invocation Content-based Strategies

Invocation content-based implementation strategies rely on discriminating service versions based on the message payload, message headers, or combination thereof. The service implementation retrieves this information and makes invocation decisions transparent to the service consumer.

An invocation content strategy based on payload versioning is better suited to Document Based Web Services, particularly those that rely on industry standard schemas since they already tend to contain payload- or namespace-based version information. The notion of payload-based versioning is analogous to how Java classes uses serialVersionUID during serialization to version classes. Evolving the payload though schema versioning is a well understood problem and a few different design techniques exist to ensure that schemas and their corresponding document instances are both backward- and forward-compatible.

For example, the Must Ignore extensibility strategy requires schemas to incorporate extensibility points into the design, allowing extension elements to be added to the schema types and attributes to each element. In a service consumer's perspective there is only one physical endpoint and the schemas contain provisions for extensibility of types and attributes. This usually means incorporating an xsd:any element that lets the complex types be extended with elements not specified by the initial schema. For example, the PurchaseOrder type in Listing 1 contains such an extensibility point and a subsequent version in Listing 2 adds to it using an OrderLevel complex type. The endpoints must cater to the presence or absence of extensions for backwards-compatibility (e.g., use default values).

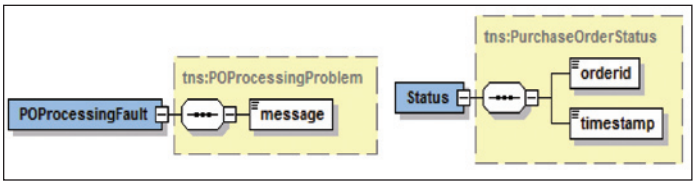


Figure 3 Schemas used in the sample service

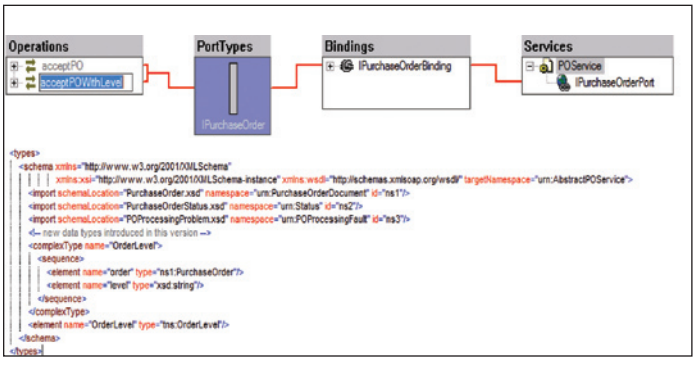


Figure 4 Incremental changes to S1

In a service developer's perspective nothing changes between versions. Existing consumers unaware of the new OrderLevel ignore this extension by default whereas new consumers may choose to use it. A sample JAX-WS service and client using this schema are provided the code.

This strategy is also known as hotfix versioning and generally tends to increase overall complexity. Other techniques for payload versioning include versioning schema with selective namespaces or adding version or identifiers attributes to individual types themselves. Deploying an updated implementation of a Web Service creates a net-new current version. However, since the change is incremental or transparent, existing service consumers aren't affected and only consumers aware of the change can leverage it. Irrespective of the underlying technique, solely relying on payload versioning tends to increase system complexity and maintainability because it depends on internalizing the versioning capabilities into the data model and system design in a predetermined manner.

The other family of invocation content-based strategies relies on the use of SOAP header blocks. Header-based versioning is commonly used for services that rely on the application of different business rules for different versions and typically don't rely on parsing the payload. The endpoint or intermediaries on the wire can promiscuously inspect headers and make routing or implementation decisions respectively. However, one use case where this may not work is the use of RESTful services that don't contain a SOAP header block. In a service consumer's perspective there's only one physical endpoint but at invocation it determines and introduces the necessary version information into the headers. In a service developer's perspective the endpoint implementation contains the logic to retrieve the version information from the header. It has to account for cases where there may be a mismatch between the header and the corresponding payload. For example, if a consumer sends an incorrect version with a correct payload or vice versa. The JAX-WS PurchaseOrder service implementation based on headers can be found in the sample code and a sample SOAP request is shown in Listing 3.

In conclusion, the realization of a SOA increases business value by increasing agility. However, change in the form of business and technical requirements is a constant in large enterprise environments and agility depends on being able to respond to this change in a timely manner. As described here, there are a number of strategies available based on the invocation context, content, or their combination and depending on the architectural requirements some may be more suited than others. The versioning problem is complex and no single strategy can satisfy all use cases but to extract maximum value from their Web Service implementations and mitigate risk early, architects need to think about the versioning issues beforehand rather than as an afterthought.

References

- Consumer-Driven Contracts: A Service Evolution Pattern - Ian Robinson http://www.martinfowler.com/articles/consumer-DrivenContracts.html
- WC3 Technical Architecture Group (TAG) http://www.w3.org/2001/tag/doc/versioning-strategies
- Java Serialization Specification http://java.sun.com/j2se/1.5.0/docs/guide/serialization/spec/serialTOC.html
- Extending and Versioning XML Languages with XML Schema http://www.pacificspirit.com/Authoring/Compatibility/ExtendingAndVersioningXMLLanguages.html
- Designing Extensible, Versionable XML Formats http://msdn2.microsoft.com/en-us/library/ms950793.aspx

11th International 2007 SOAWORLD™ CONFERENCE & EXPO

2007 VIRTUALIZATION CONFERENCE + EXPO www.virtualizationconference.com

iTVCON.COM INTERNET TV CONFERENCE & EXPO 2007

The Number One SOA Event Comes to San Francisco Colocated with the Virtualization Conference & Expo

REGISTER NOW SAVE $200 USE COUPON CODE JDJ SOA 1210

NOVEMBER 12-13, 2007

AT THE GRAND HYATT IN SAN FRANCISCO

Register Online!
www.SOAWorld2007.com

Service-oriented architectures (SOAs) have evolved over the past few years out of the original vision of loosely coupled Web services replacing constrained, stovepiped applications throughout enterprise IT. Every major enterprise technology vendor today has developed its own SOA strategy, supported by innumerable mid-size companies and start-ups offering specific SOA aspects or entire solutions. This explosive growth in SOA technology is in response to a global demand – IDC estimates that spending on SOA services alone will grow from $8.6 billion to more than $33 billion by 2010.

SOA World Conference & Expo 2007 brings together the best minds in the business to San Francisco for a two-day conference that offers comprehensive coverage of SOA and what it means to enterprise IT today. As Zapthink analyst Jason Bloomberg has noted, "SOA involves rethinking how the business leverages IT in many various ways." Attend SOA World Conference & Expo 2007 West and learn from more than 100 speakers about how SOA is transforming business – and the way IT and business managers think about their businesses, processes, and technology.

The colocation of SOA World Conference & Expo 2007 West, Virtualization Conference & Expo, and iTVCon - Intenet Video Conference & Expo 2007 delivers the most relevant content to IT and business. Conference attendees will be able to choose from five great tracks at this year's event. Mix and match all you want, or slot into your favorite topic for the duration!

TOPICS

SOA World:

- Enterprise Mash-Ups
- SOA and Web 2.0
- SOBAs
- Patterns and Anti-Patterns
- Governance Enforcement
- AJAX RIA and SOA
- Run-Time Governance
- Wikis and Lightweight Software
- Policy-Driven SOA
- Complex Event Processing
- Windows Interoperability
- EAI Without the Cost
- Mixed Portal Environments
- Reference Architecture

Virtualization:

- Service Virtualization
- The Effect on Enterprise SOA
- Managing Data Centers
- Virtualization Economics
- OS Virtualization
- Adaptive Infrastructure

iTV:

- Advertising Models for Video-on-Demand (VOD)
- Internet TV Commercials
- Mastering Adobe Flash Video
- How to Harness Open Media Formats (DVB, etc)
- Multicasting
- Extending Internet TV to Windows CE-based Devices
- Live Polling During Webcasts
- Video Press Releases
- Pay-Per-View
- Screencasting
- Video Search & Search Optimization
- Syndication of Video Assets
- V-Blogs & Videoblogging
- Choosing Your PVR
- Product Placement in Video Content
- UK Perspective: BBC's "Dirac Project"
- Case Study: SuperSun, Hong Kong

For Exhibit and Sponsorship info:
Call 201 802-3020 or email
events@sys-con.com

Avoid VDA! (Vendor-Driven Architecture)

It's time to throw your comfort blanket away

WRITTEN BY **DAVID S. LINTHICUM**

When looking at technology buying patterns in the world of SOA, there's one common thread. The Global 2000, and many government agencies, are purchasing from their existing vendors, no matter what the needs or requirements. I call these solutions purchasing "comfort technologies" since they consider the relationship with the vendor more than the value of the technology itself. It's comforting to deal with the same company, people, and platform.

Moreover, many of these same companies working with "comfort technologies" are also allowing the vendors to design and define their solution. I call these vendor-driven architectures or VDAs but they are always called a bad idea if you understand the core issues. What's most disturbing is that this seems to be an emerging buying pattern. Architects are making the "SOA deal" with a vendor in hopes some magical technology emerges from the box that will suddenly make their existing poorly defined and design architecture, agile, loosely coupled, and return quickly on the investment. Won't happen without work, and there is no magic technology that can enable SOA. It's an architecture not a technology after all.

What this means in the long run is failed SOAs where the blame is put on the concept, perhaps the product, but never the architecture or the architect, where and when the work needed to get done. While we've made similar mistakes over the years, and are paying the price right now, we risk history repeating itself, if we're not careful.

The influence of the larger SOA vendors is very much a force in the market today so learn to discriminate. It may not be what you want to hear, but ultimately it's the right thing to do. The first step is to learn to recognize VDA, and don't let it kill your SOA before it has a chance to do some good. There's too much at stake.

Moving Out of Your Comfort Zone

So, why are organizations looking to their "comfort vendors" and "comfort technologies"? It's a matter of the path of least resistance and lack of education.

Path of least resistance because the relationship is already established and you don't have to go through the hassle of getting to know new players or many new players. So it's easier to buy the latest SOA stack from good old Bob than it is to go through the detailed requirements, analysis, and design required to build an effective SOA.

And many people who purchase technology don't know the first thing about SOA, or even their own requirements and business drivers for that matter. An effective knowledge transfer project is needed to understand the basics of building a SOA, like figuring out your own requirements, including a semantic-level, service-level, and process-level understanding of the problem domain or enterprise. Then, and only then, should you begin pinging vendors, including your comfort vendors about what technology may work best for you, considering that in many cases it will be a collection of technology from many vendors. For sure, not comfortable, but necessary.

Of course beyond the issue of always leveraging the same "comfort vendors" is the issue of vendors actually creating the architecture for the enterprise. This is wrong on so many levels it's difficult to know where to start but it's part of the impact those millions of marketing dollars spent larger SOA vendors is having. There are three major areas of concern:

First, the vendors who drive "SOA certification" programs. While these programs are sold as an objective SOA education, it's a way to get into deals and lead students to the promised land of the vendor's SOA technology stack. Not that this is a trick, it's not. They are merely acting in their best interest, but in many cases it may not be yours

Second, technology vendors who actually define, design, and build your SOA. The issue here is the fact that you're typically going to end up with that vendor's SOA stack. So you're missing opportunities for efficiencies that may come from other technology that may not be considered because it's not in the best interest of the vendor who's building your SOA to consider them.

Third, SOA vendors that sell "one-stop shopping" for SOA. While the "super SOA stack" approach is getting a lot of play considering the amount of marketing dollars behind those vendors, it's typically never the optimal solution for your requirements. Indeed, architects aren't doing their job if they simply point at a vendor and say yes, it's best if they understand the requirements, tactical and strategic, before defining the proper solution, and then and only then, select the technology that's optimal. Sorry, no "one-stop shopping." ■

About the Author

David S. Linthicum (Dave) is a managing partner with Zapthink, LLC (www.zapthink.com), a consulting and advisory organization dedicated to excellence in SOA planning, implementation, training, mentoring, and strategy. He is an internationally known application integration and SOA expert. In his career Dave has assisted in the formation of many of the ideas behind modern distributed computing including Enterprise Application Integration, B2B Application Integration, and SOA, approaches and technologies in wide use today. He keynotes at most major SOA and Enterprise Architecture conferences, maintains one of the most read SOA blogs, is the host of the weekly SOA Report Podcast, and is the author of 10 books, three on integration and SOA topics.
david@linthicumgroup.com

Mindreef®

SOAPscope server™

Whether you're an Architect, Developer or Tester, rely on the Web services and SOA tools that won't leave you stranded.

Mindreef provides an integrated set of tools at the XML software layer for SOA and Web services policy governance, testing, diagnostics and support. Available in server or workstation licensing options, Mindreef SOAPscope® products help professional developers, testers and architects working alone or on teams to understand and address SOA and Web service problems quickly and cost-effectively.

Visit our website and learn why more than 3,000 customers worldwide, including 40 of the Fortune 100, use Mindreef products to successfully build, test and maintain high-quality Web services and SOAs.

SEE IT!

Register on our website to attend a webinar on Automating Web services and SOA testing

TRY IT!

Request an evaluation copy of Mindreef SOAPscope Server or SOAPscope Workstation™

BELIEVE IT!

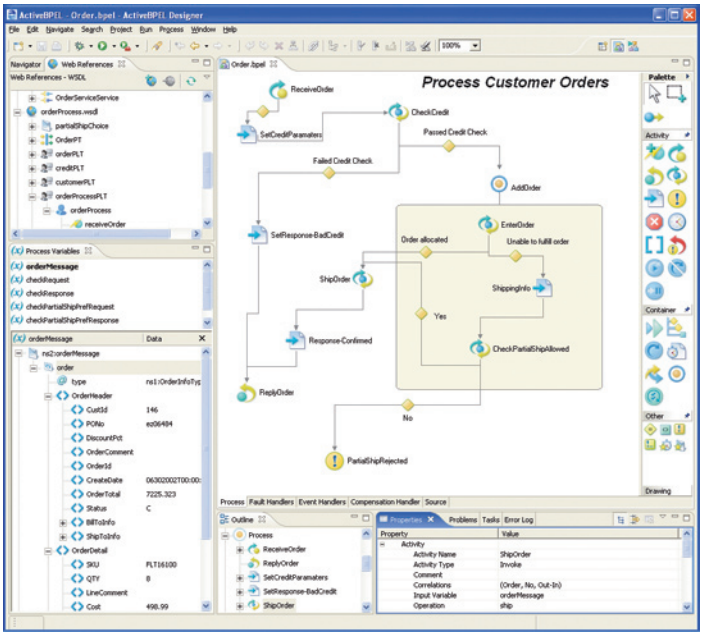
Start saving time and money, and improving the quality of your Web services and SOA projects

The SOA Quality Company™

www.mindreef.com

(603) 465-2204 ext. 581

Get Started with BPEL 2.0



Build next-generation SOA applications with the leader in BPEL technologies

Download BPEL tooling & server software today

active-endpoints.com/soa

BPEL consulting and training.
BPEL design tools, servers and source code for Eclipse, Apache Tomcat, JBoss, WebSphere, WebLogic, BizTalk and Microsoft .NET.

activeBPEL

active endpoints