SYS-CON

THE WORLD'S LEADING MAGAZINE DEDICATED TO WEB SERVICES TECHNOLOGIES

SOA Web Services JOURNAL

JULY 2006 / VOLUME: 6 ISSUE 7

Managing SOX in the Age of SOA

Policy-Based Version Control for SOA Services

Formalizing Middle-Tier Data Management

The Performance Woe of Binary XML

XML Compression and its Role in SOA Performance

PLEASE DISPLAY UNTIL SEPTEMBER 30, 2006

$6.99US $7.99CAN

07>

0 71486 03420 9

Altova® StyleVision® 2006 – The ultimate e-forms, DB report and stylesheet designer.

ALTOVA® stylevision® 2006

Transform your appearance

Outfit yourself with StyleVision® 2006, and fashion multiple outputs from a single stylesheet design.

New in Version 2006:

- Vastly enhanced user interface and usability features
- Support for multiple data sources in one output design
- Cascading stylesheets (CSS) for use in transforming XML and databases to HTML
- JavaScript editing and embedding in HTML designs

Altova StyleVision 2006, the ultimate e-forms, DB report, and stylesheet designer, lets you transform XML and database content into eye-catching HTML, PDF, and Word/RTF output all from the same design. Also use it to create intuitive electronic forms for Authentic® 2006, Altova's powerful, FREE XML and database content editor that enables business personnel to view and edit data without being exposed to the underlying technology. Get more out of your designs!

Download StyleVision® 2006 and Authentic® 2006 today:
www.altova.com

ALTOVA®

SOA Web Services JOURNAL
JULY 2006 / VOLUME: 6 ISSUE 7

TABLE OF CONTENTS

Visit us online at WebServices.SYS-CON.com

Inside This Issue

FEATURE

16

Michael Poulin

Service Versioning for SOA

Policy-based version control for SOA services

CORPORATE COMPUTING

16

Hugh Taylor

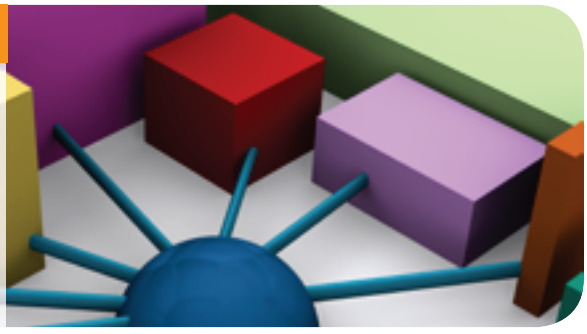
Managing SOX in the Age of SOA

Rethinking internal controls

INFRASTRUCTURE

46

Bharath Rangarajan



Enterprise Data Fabric (EDF)

Formalizing middle-tier data management

FROM THE EDITOR
A Little Help from My Friends
By Sean Rhody 5

INDUSTRY COMMENTARY
Milliseconds Matter
By Chris Martins 6

SOA
The Seven Secrets of SOA Success
How not to be misled
By Greg Coticchia 8

FRAMEWORK
SOA 2 Point Oh No!
The notion of SOA 2.0 is just plain silly
By David S. Linthicum 10

CASE STUDY
MedicAlert Embraces SOA To Drive Business Agility
How a Non-Profit Leveraged a SOA Infrastructure To Change its Business Model
By Jorge Mercado 12

BUSINESS
Business Rules Café
How IT Can Stop Worrying and Love Change
By James Taylor 26

XML
The Performance Woe of Binary XML
The scapegoat's story
By Jimmy Zhang 28

SOA
Turning Service-Oriented Events into Business Insight
Event-Stream Processing – tools for an event-driven service oriented architecture
By Mark Palmer 32

SECURITY
Creating Secure Web Service Sessions
SSL can complement your Web services security solutions to achieve optimal performance
By Kevin T. Smith 38

XML
XML Compression and Its Role in SOA Performance
Dealing with the increased size of SOA payloads
By Akash Saurav Das & Srinivas Padmanabhuni 48

SOA Web Services JOURNAL
www.WebServices.SYS-CON.com

EDITORIAL OFFICES
SYS-CON MEDIA
135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9637
WEB SERVICES JOURNAL (ISSN# 1535-6906)
Is published monthly (12 times a year)
By SYS-CON Publications, Inc.
Periodicals postage pending
Montvale, NJ 07645 and additional mailing offices
POSTMASTER: Send address changes to:
WEB SERVICES JOURNAL, SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645

SYS-CON MEDIA | www.SYS-CON.com

A Little Help from My Friends

WRITTEN BY **SEAN RHODY**

It's sometimes funny to write about service-oriented architecture. One of the things I say often and believe is that you can't buy a service-oriented architecture. SOA is not just technology, it's philosophy, organizational change, and business transformation. There's no place to buy that kind of dramatic, deeply impacting change.

The funny part, at least to me, is that you can, however, buy or acquire a good deal of infrastructure to set this up from a single source. In the industry, we call that a platform. And that's what this month's issue is about – SOA platforms.

Service-oriented architecture is a powerful concept – at its base, it replaces the concept of an application with the concept of a service. This isn't trivial. For 40 or more years, we've dealt with computers in the context of applications (I've left a few years off my count to account for the years before operating systems when we were really talking about programs, or code). We have the context of an application firmly established – people go to their computers, interact with applications, and do work.

SOA is about the next phase of computing – enabling computers to do some of the work unaided (computer-to-computer interaction), and freeing the user from having to interact with application after application to get work done. Anyone who's ever dealt with a call center and had to wait on the phone while the call center representative said, "Now give me just a minute, I have to bring up another application to do that," knows how frustrating, time-consuming, and challenging the traditional application-centric approach can be. Services (and the composite applications that no longer require the user to switch between applications to do a single business process) are the way we plan to break the tyranny of the application and take back control of the computer.

Services don't exist in a vacuum. Part of the chicken and egg problem we have in moving from applications to services is that services make only a limited amount of sense in the context of a single application, so there's not a strong impetus for application developers to use a service-based approach toward design when creating applications. Similarly, when you take a holistic view of the enterprise, you don't necessarily see the catalog of services you'd like to have to really establish a true architecture, and derive the highest value from composing processes from services.

That's where a platform becomes the middle ground and the way out of the chicken and egg problem. By establishing a platform for the enterprise, you can start to drive applications toward becoming services. A platform should provide discovery, registration, an ESB at the minimum, with most platforms also including security and management services as well. When you start to build your applications (or buy them specifically for the platform in many cases) to take advantage of the service orientation of the platform, you start to realize the advantage of the platform, and move past the challenge of service-enabling applications.

It also underscores the importance of getting a strong platform, one where the parts all work together well. You may not want to get everything from a single source or, depending upon your enterprise needs, you may not be able to do so. A soup-to-nuts infrastructure out of the box from a single vendor is not necessarily nirvana anyway, as those of you who've had the best-of-breed discussions over the years can attest. What is important is establishing an SOA infrastructure, or platform, on which an enterprise can leverage its application portfolio. The platform enables the portfolio transformation that must occur in order to move from an application orientation to a service orientation.

So it's funny, even though I don't think you can buy an SOA, I definitely think you can acquire a platform, and you're much better off with one then without. ■

About the Author

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

Milliseconds Matter

WRITTEN BY **CHRIS MARTINS**

In financial trading, if you're slow to act on an opportunity, it's gone, seized by a quicker competitor. His profit is your loss. Electronic traders can easily miss a trading opportunity because their trading algorithms failed to detect the right conditions – or didn't detect them quickly enough. So for securities trading operations dependent on automated algorithmic trading – where profit or loss is determined in less than a second – milliseconds do matter.

Event Stream Processing enables organizations to make rapid decisions in response to highly dynamic data — data that's continuously changing. Besides financial services, ESP applies to industries as diverse as telecommunications, manufacturing, distribution, retail, energy, and military. Most organizations can benefit from understanding the impact of events as soon as they occur.

Organizations must advance their computing and data architectures to address the real-time demands of today's environments. ESP, which has also been referred to as complex event processing and data stream processing, has the potential to deliver enormous benefits. Imagine the ability to monitor a city's power grid by tracking the load on generators in combination with a real-time feed of the weather forecasts, and optimizing generator output on-the-fly to changing conditions.

In these situations – including the aforementioned electronic trading desk – increasingly sophisticated systems can now be deployed to actively monitor rapidly moving event streams and identify both discreet data points, as well as sophisticated data patterns. The event streams can originate from sensors dispersed in a wireless network, satellite telemetry signals, RFID tags, or stock ticks from a market data feed. In each instance, the events coursing through these data streams can be sifted, sorted, and otherwise analyzed to derive operational value for the business operation. And in each instance, the goal of the new architecture is to understand quickly what's happening and respond accordingly.

Traditional architectures are designed around a core presumption of a stable persistent data model with a relational database as caretaker. In these traditional architectures, changes to the data (inserts, deletes, or updates) must conform to strict rules that preserve the inviolable ACID properties of a transaction – its atomicity, consistency, isolation, and durability. Ensuring data accuracy is paramount: transactional integrity is always a priority over speed. Traditional database principles necessarily build latency into their operations to assure this data integrity. As data changes, the databases must be updated, indexes maintained, and then, after that, queries can be executed. You can't query what you haven't indexed; and you can't index what you haven't written to the database.

As valuable as such constraints are for traditional computing, they impose too much latency on ESP applications. ESP applications must often monitor tens of thousands of new events that are generated every second and tens of thousands of event "conditions" at the same time. For such applications, traditional transactional models are equivalent to electronic bureaucracies whose transactional processes only impede the real work to be done. To handle the volume and speed of event processing, a new model is required.

ESP assumes that many events lack significance, seek to filter out the noise, and derive meaning from the events that do have value. Often that meaning can only be understood in the context of other events that happen at the same time or in close proximity. So ESP applications understand that time is key in determining event significance; when an event happens is often as important as what happens.

In all business areas, conventional data management technologies can't cope with real-time analysis and response. It's time for enterprises to rethink traditional data processing strategies. ESP systems must address the special attributes of event stream data: the volume of data, the speed at which it's delivered, the temporal characteristics of the data, the potentially high amount of "noise," and the need to act in milliseconds to capitalize on events as they happen. ■

About the Author

Chris Martins has product marketing responsibilities for the Event Stream Processing product line of Progress Software's Real Time Division. Chris has more than 15 years of experience in the software industry, with a background that encompasses workflow and information management infrastructure software, as well as customer relationship management applications.

SOAWebServices JOURNAL

www.WebServices.SYS-CON.com

CORPORATE

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

Group Publisher
Jeremy Geelan jeremy@sys-con.com

ADVERTISING

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

VP, Sales & Marketing
Miles Silverman miles@sys-con.com

Advertising Director
Robyn Forma robyn@sys-con.com

Advertising Manager
Megan Mussa megan@sys-con.com

Associate Sales Managers
Kerry Mealia kerry@sys-con.com
Lauren Orsi lauren@sys-con.com

SYS-CON EVENTS

Associate Event Manager
Lauren Orsi lauren@sys-con.com

CUSTOMER RELATIONS

CIrculation Service Coordinator
Edna Earle Russell edna@sys-con.com

SYS-CON.COM

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

Web Designers
Stephen Kilmurray stephen@sys-con.com
Wayne Uffleman wayne@sys-con.com

ACCOUNTING

Financial Analyst
Joan LaRose joan@sys-con.com

Accounts Payable
Betty White betty@sys-con.com

Accounts Receivable
Gail Naples gailn@sys-con.com

SUBSCRIPTIONS

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

MAKE YOUR SOA SECURITY MOVES WISELY...

XWALL
WEB SERVICES FIREWALL

XRAY
WEB SERVICES DIAGNOSTICS

VULCON
VULNERABILITY CONTAINMENT SERVICE

SENTRY
SOA SECURITY GATEWAY

PUTTING TOGETHER THE PIECES FOR THE WORLD'S MOST DEMANDING SOA SECURITY SYSTEMS

FORUM SYSTEMS ENTERPRISE SOA SECURITY SOLUTIONS:

- TRUSTED SOA MIDDLEWARE
- WEB SERVICES SECURITY
- XML ACCELERATION

WWW.FORUMSYSTEMS.COM

FORUMSYSTEMS
THE LEADER IN WEB SERVICES & SOA SECURITY

The Seven Secrets of SOA Success

How not to be misled

WRITTEN BY **GREG COTICCHIA**

There's no doubt that the computing era of Service Oriented Architecture is upon us. Everyone has caught SOA fever (is it S-O-A or SO-AH?) and most Fortune 500s are considering or have already implemented their first set of services.

Why is SOA so compelling? Zapthink, one of the industry's experts on SOA, suggests the following reasons for folks to be enamored with SOA and why it's worth all of the hype:

1. Increased Business Agility
2. Reduced Integration Expense
3. Increased Asset Reuse
4. Reduced Business Risk and Exposure

Pretty compelling and timely. In a recent BusinessWeek cover story entitled "Is Your Company Fast Enough" (http://www.businessweek.com/magazine/content/06_13/b3977001.htm) the author, Steve Hamm, makes the point that "Speed doesn't kill but is indeed the ultimate competitive advantage, especially with loosely coupled businesses connected on a global level being the basis for progressive business models." On the podcast for this article, the author cites Microsoft and Vista, the next version of Windows, as an example of not being fast. Vista has taken five years to get to market and has just been delayed again. In contrast, BusinessWeek points out that Google constantly outflanks Microsoft and releases new products, some winning, some not, but still agile, risk-reducing, and meeting customer needs. Sounds like what we expect from SOA.

Despite the number of articles and vendor claims about SOA few have spelled out the specifics of how to be successful. This article will reveals the seven secrets to being a hero with your SOA initiative. It's based on LogicLibrary's five years of experience in delivering services and solutions to major global companies. Along with our customers, we've overcome many of the challenges and now we'll share them with you.

Secret #1:
Don't Boil the Ocean

As with anything new, exciting, or potentially enormously beneficial, we can get carried away. Remember the client/server hype? Everything still isn't Internet-based, and those mainframe applications are running just fine. So be selective. Don't start with a massive project that involves a cast of thousands. Think big but start with a small project. Begin with a project that involves buy-in from three camps: an architecture representative, a development leader, and someone who represents the business (could be in the IT management food chain or actual line of business, doesn't matter). Focus on a project that can highlight the clear benefits of SOA like reworking a small set of key business processes to improve their flexibility. Clearly identifying a "before SOA" and "after SOA" time frame is a good way to get started.

Secret #2:
Beware of 'ABOS'

We've all been victims of spaghetti code. The SOA equivalent is "A Bunch Of Services" or ABOS. You can end up with ABOS if you don't align your business needs with your technical capabilities and understand where you can put your services, who will own them, who will use them, etc. In short, know what you have so you can move ahead. It's the basics of good governance. Developing 10 or 20 narrowly scoped services and putting them out there may be a gratifying short-term exercise, but it will be a long-term nightmare. Get your fundamentals correct – especially focusing on business alignment and flexibility alongside, perhaps, the more obvious technical requirements. Make sure you have a design-time repository/registry technology in place (not a spreadsheet or database, please) and make sure it can scale to your needs (you don't want to get a nice department-level product only to re-decide later). But simply putting your services (also known as software development assets and artifacts) somewhere you can find them isn't enough. You have to be able to understand, observe, and guide those who are producing services along with those who are consuming them. Make sure you establish an architecture and design-time governance process that can provide this kind of guidance and that can also produce measurable results, one that goes beyond just allowing you to locate your assets and artifacts.

Secret #3:
It's Not a Process Or a Product; It's a Process And a Product

Speaking of vendors, there are thousands (if not more) on your phone, in your e-mail and at your door telling you they can solve all your SOA problems. Don't believe them. It's important to remember one important secret: Don't wait until your process is right to add the products you need. It's logical, but a bad idea. When you decide on the small pond (versus ocean) project, you'll need process and technology together. These aspects of a successful SOA aren't se-

rial they're interactive. Getting the process "right" (which will never happen) without a set of initial products you deploy to support and enable the process is as bad as buying products without having analyzed the services and process methodology changes you'll have to make. So get in the pond, select your methodology (with a service provider to help), and get the initial set of tools needed to create the foundation.

What are these tools? Many of the same ones your development teams are using today such as IDEs, SCM systems, defect-tracking systems, requirements management systems, and test automation systems. But there's one other key product you probably don't have in your portfolio – a design-time repository/registry that both binds all these tools together and serves as the communications and governance bridge across all service production and consumption activities. Many folks don't get this right. They wait and wait and wait for the perfect process, and six to 12 months later they're still "studying" what to do, or waiting to perfect their process. By then most products could never reflect their process, and all their developers have "developed" a new set of bad habits – remember, you get one chance to set the ground rules for SOA; after that first project, behaviors become ingrained. So, it's a service and product together. That's the ticket.

Secret #4:
Big Tent Approach

Not all "services" are Web Services. Forrester SOA analyst Randy Heffner did a study of companies using SOA and found that "While most participants were either doing Web Services or planning for them, one firm — a firm that had a major implementation of services — had yet to build its first Web Service. Instead, services were accessed via IBM's WebSphere MQ (née MQSeries)." Another participant had an advanced service invocation framework and the core service access mechanism was JMS. Another firm started doing services with the immediate goal of connecting to business partners, so Web Services played a part in its first service implementation. The lesson is this: Service orientation provides great value with or without Web Services, while Web Services provide a universal accessibility layer on top of a service-based design. Pursue the two as separate but related initiatives. As you plan and build out your SOA it's crucial that you're able to manage all services.

Secret #5:
Standards Shmandards

Don't let the endless chatter about standards slow you down. Let's start with the Web Services standards bodies. There are four! And the standards? Well, the current alphabet soup of Web Service standards (what if we named a standard and no one used it) includes XML, SOAP, WSDL, UDDI, and at least nine others published and promoted. Sure you need to take security, payload structure, and all of those technical issues seriously, but if you wait for all the WS-* standards to sort themselves out you'll be waiting a long time. The real secret here is that most folks today use two: XML and SOAP – your technology stack of choice will help you out with the rest (and if they're worth their salt, will help you migrate your implementations once all those WS-* standards settle down).

And how about UDDI? This standard, which was originally specified as a runtime registry, has become commoditized (Open Source UDDI is readily available and many vendors offer UDDI for free) and seems to be moving from runtime to design time in an attempt to add value. But keep in mind, as we mentioned before, that not all services are Web Services. If you chose a UDDI "standard" (and all of its "user friendly" concepts like models, category bags, and the like) to help your design-time efforts you'll only be focused on Web Services. That's not enough — you have to be able to manage ALL your services.
So don't get wrapped around the standards axle. DCE anyone?

Secret #6:
Repeat After Me, "Design Comes Before Production"

We all know that you should design and develop something prior to running it in production. This has never been more important than in an SOA environment. But what often happens is that organizations are stymied by concerns about the performance of new loosely coupled services, even before they're written! Or, even worse, they give in to the temptation to create and deploy services without considering the management issues. Don't let this happen to you — if you get out the map and plan your service route properly you're likely to reach your SOA destination. Specifically for a development issue, quality must be designed into the product, not inspected into it.

Deciding which tools you'll need is one of the first issues to be addressed. A performance monitor, while valuable, isn't at the top of the list. The best place to start is with your current design and development tools, services vendors, and of course your own practices. Get those in shape, do a gap analysis, and see what's missing for your pond project. But start on the design/development part of the equation (and don't forget architecture on the design side of the equation). Get that right and the rest will be much easier. Just like the application itself.

Secret #7:
Be a Carpenter (Measure Twice Cut Once – Hey, How About Just Making Sure You Measure)

Key to any part of SOA success is measuring. All your tools have to provide evidence of measurable value. The most important key to your services is governance. Who's producing these services? How many are there? Where? Who's consuming them? For what? Can they understand what those services are supposed to do? As you can tell by answering these questions, you only need a handful of services before these concerns arise. Get control of them before they get control of you. Even 12 or less services could be a potential disaster without the right design-time SOA governance process (and supporting tools) in place.

Summary

As with any effort in new technology the pioneers that went first ended up with a few "learning" arrows in their backs. SOA is now past the pioneering stage, enough so that we can be clear about the secrets of a successful SOA. By using these seven secrets you can avoid the arrows and reap the rewards of a properly designed and executed SOA.

About the Author

As CEO of LogicLibrary, Greg Coticchia provides overall direction for the company. He brings to this role 20 years of experience in high-technology start-ups, including executive leadership roles in strategic planning, marketing, business development, sales, and general management. Earlier, Coticchia was with AXENT/Symantec, where he served as senior vice-president of marketing and business development. During his tenure Coticchia helped the company grow from $69 million to $150 million in revenue until its acquisition by Symantec for almost $1 billion. Coticchia holds an MBA from the University of Pittsburgh, Katz School of Business, and a bachelor's degree in industrial engineering from the University of Pittsburgh. He also completed Carnegie Mellon University's Entrepreneurial Management Program.

SOA 2 Point Oh No!

The notion of SOA 2.0 is just plain silly

WRITTEN BY **DAVID S. LINTHICUM**

Here we go again. While the paint is still wet on this new Web 2.0 stuff, many SOA vendors and large analysts firms are calling their market SOA 2.0. It's one of the silliest things I've heard in a long while, and both the analysts and vendors who use this term should be ashamed of themselves.

I get Web 2.0 because the Web is well over 10-years-old and we've been successful in using this pervasive technology and now we're moving to newer and more exciting stuff such as AJAX and RSS thus the new version number. However, we've yet to get large-scale traction with SOA so SOA 2.0 is illogical since SOA 1.0 never existed if we're realistic.

Moreover, SOA is an architectural concept, not a software product, and to put a version number on something like that shows you don't understand the notion in the first place. SOA is a journey, not a project or product, and to try to make it such is to demean the core concept and the value it can bring. My larger concern, however, is that hype like SOA 2.0 could cause many of those moving towards SOA to become disenchanted and ignore the architectural issues, and hurt their business.

I suspect the marketing guys are at it again and that that's where this thing came from. Once again the people who buy the technology have to get involved and push back against this kind of foolishness or else you'll see it again and again. As such, I urge you to tell your vendors that SOA 2.0 is silly, and if they use the term they'll lose creditability. If enough hear that, the term will die, and other new marketing words like "SOA 3.0," "SOA Next Generation," and "SOA-nator" won't show up either.

SOA (No Version Number)

A SOA is a strategic framework of technology that allows all interesting systems, inside and outside an organization, to expose and access well-defined services and the information bound to those service that may be further abstracted to orchestration layers and composite applications for solution development. This is not a product, not a piece of software; this is an architectural concept. Am I clear?

The primary benefits of a SOA include:

- ***Reusing services/behaviors*** or the ability to leverage application behavior from application to application without a significant amount of re-coding or integration. In other words, using the same application functionality (behavior) over and over again, without having to port the code, leveraging remote application behavior as if it existed locally.
- ***Agility***, or the ability to change business processes on top of existing services and information flows, quickly, and as needed to support a changing business.
- ***Monitoring***, or the ability to monitor points of information and points of service in real-time, to determine the well being of an enterprise or trading community. Moreover, the ability to change processes to adjust processes for the benefit of the organization in real-time.
- ***Extend reach***, or the ability to expose certain enterprises processes to other external entities for the purpose of inter-enterprise collaboration or shared processes. This is, in essence, next-generation supply chain integration.

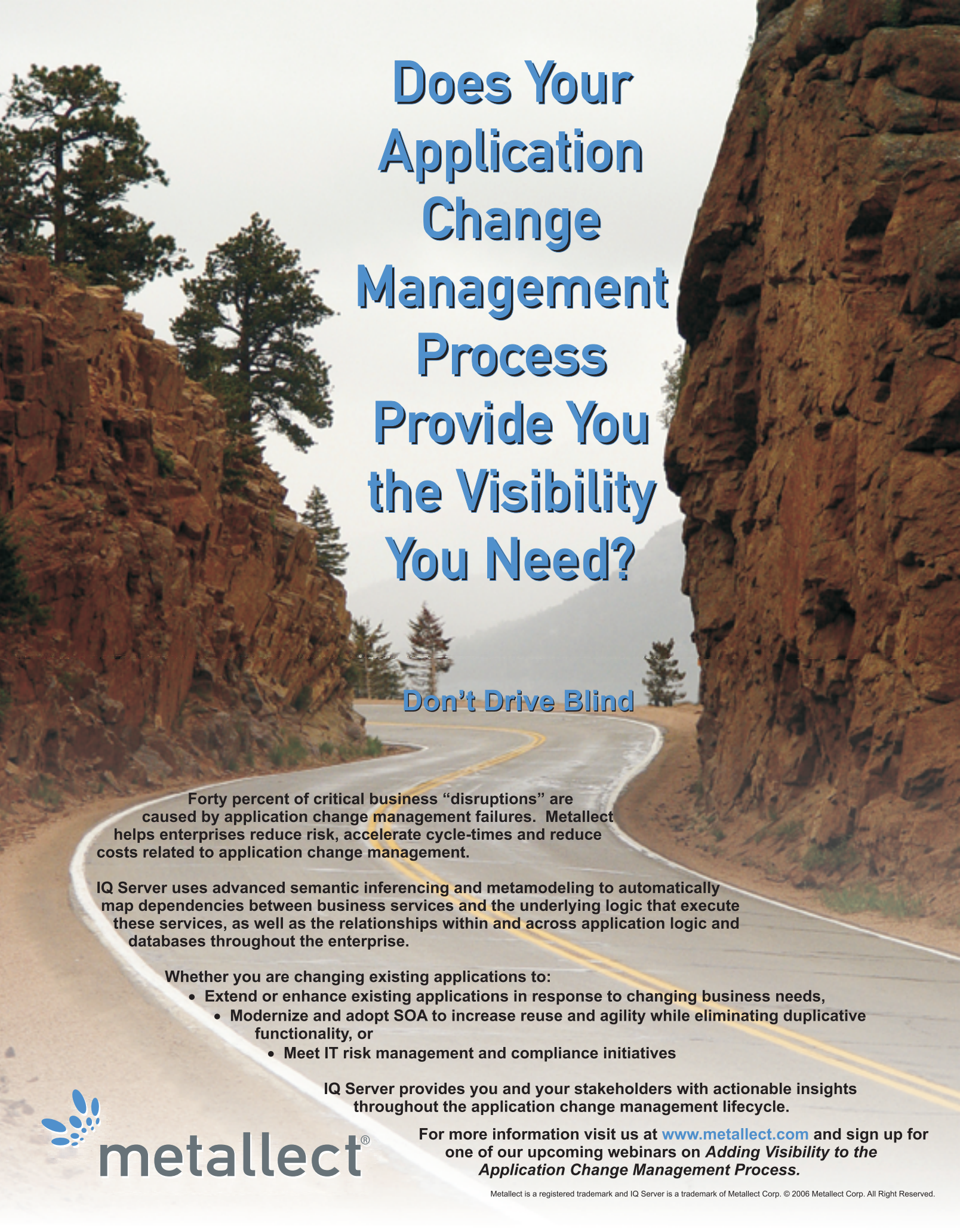
The notion of a SOA isn't new at all. Attempts to share common processes, information, and services have a long history, one that began more than 10 years ago with multi-tier client/server — a set of shared services on a common server that provided the enterprise with an infrastructure for reuse and now provides for integration — and the distributed object movement. "Reusability" is a valuable objective. In the case of a SOA it's reuse of service and information bound to those services. A common set of services among enterprise applications invites reusability and, as a result, significantly reduces the need for redundant application services.

What is unique about a SOA is that it's as much a strategy as a set of technologies, and it's really more of a journey than a destination. Moreover, it's a notion that depends on specific technologies or standards such as Web Services, but really requires many different kinds of technologies and standards for a complete SOA.

SOA as a Discipline

What's clear about SOA is that while we are now beginning to see tactical successes, the large-scale benefits of leveraging this concept have yet to be understood by most organizations. Truth be told, it's going to take time before we can brag about the benefits of SOA, and perhaps the hype will have died down by then, thus some of the confusion that's around today. This confusion includes the number of WS-* standards that are around, many of which are redundant and conflicting. But that's another column or blog.

While SOA 2.0 is a silly notion, we look to evolving our thinking to a place where SOA is more "the architecture," not "an architecture." And there's a difference. What's more, we have to understand that systemic changes such as using SOA is going to take most organizations many years to implement. Unfortunately there are no shortcuts like changing version numbers.



Does Your Application Change Management Process Provide You the Visibility You Need?

Don't Drive Blind

Forty percent of critical business "disruptions" are caused by application change management failures. Metallect helps enterprises reduce risk, accelerate cycle-times and reduce costs related to application change management.

IQ Server uses advanced semantic inferencing and metamodeling to automatically map dependencies between business services and the underlying logic that execute these services, as well as the relationships within and across application logic and databases throughout the enterprise.

Whether you are changing existing applications to:

- Extend or enhance existing applications in response to changing business needs,
- Modernize and adopt SOA to increase reuse and agility while eliminating duplicative functionality, or
- Meet IT risk management and compliance initiatives

IQ Server provides you and your stakeholders with actionable insights throughout the application change management lifecycle.

metallect®

For more information visit us at www.metallect.com and sign up for one of our upcoming webinars on *Adding Visibility to the Application Change Management Process.*

MedicAlert Embraces SOA To Drive Business Agility

How a Leading Non-Profit Leveraged a SOA Infrastructure To Change its Business Model for the Better

WRITTEN BY **JORGE MERCADO**

At MedicAlert we have vividly seen how a Service Oriented Architecture (SOA) can enable business agility and elevate the value of the IT organization's work. Since we launched our SOA initiative two years ago, we have laid the foundation for vital new forms of collaboration with partners and accelerated the introduction of new products and services that can strengthen the business.

MedicAlert, a non-profit organization with four million members worldwide, is best known for the medical bracelets worn by our members. The bracelets let doctors and other healthcare providers know if there's a particular medical condition – such as an allergy or illness – that must be recognized in an emergency situation.

While we have provided services that protect and save lives for 50 years, we have moved more deeply into healthcare information services in recent years. We now enable members to log into our systems and manage their personal health records while maintaining security, privacy, and confidentiality. The MedicAlert repository relies on Web Service interfaces to support standard Personal Health Records (PHR), including electronic drug prescriptions and for patient record interoperability. The repository of personal health information facilitates the delivery of critical medical information between patients, providers, payers, and emergency responders around the clock and across the world.

We realized two years ago that our growth, improvement, and continued success were dependent on creating greater interoperability with other health service organizations such as hospitals, doctors' offices, labs, pharmacies, and healthcare payers. We wanted to be able to accept information from these partners and populate our members' medical records on their behalf. Unfortunately, the systems we had in place were not designed to effectively populate data in this fashion.

SOA for Interoperability

Recognizing this challenge, we embarked on an initiative to implement a services-based system. We now have approximately 20 Web Services in production – and many more in development.

We began with a bottom-up approach that enabled us to start small and get results quickly. As time progressed and business needs continued to come forth, we began to engage in more of a top-down approach to our SOA delivery strategy. More specifically, we embarked on an approach whereby both strategies would converge.

We looked at how to get our systems implemented quickly using Web Services in ways that would support our business. We were most concerned with the security and management of these services, and evaluated management tools and industry-standard security mechanisms that would meet our architectural requirements. We were able to move fast; while remaining aware of the business objectives we needed to address processes we intended to model. Ultimately, we deployed Microsoft BizTalk Server 2004 as the process integration and rules engine and AmberPoint for runtime governance and Web Services management.

Our SOA system has dramatically enhanced partner interoperability. In the past, we would have to rely on S/FTP-based solutions that would involve crawling through data and writing records directly to a database. While that may be a "classic" way of conducting this sort of activity, it wasn't a scalable one.

A better approach is to have a generalized set of services that can be offered up to these partners either directly or indirectly. While one has to be prepared to address custom requirements as necessary, the objective is now to create a scalable system and be able to wrap custom policies around the services that are produced. Our new policy engine enables us to execute different policies to accomplish a vast array of tasks in a scalable and secure way.

Runtime Governance

Runtime governance is critical to the success of any services-based system. We took a long look at our options here. Our selection of AmberPoint has enabled our

architecture and infrastructure teams to concentrate on Web Services implementation and performance as opposed to the litany of low-productivity tasks that would have been necessary in its absence. Among the benefits of the "policy-based approach" that we've taken to governance are:

- ***Performance Metrics***. We are now capable of monitoring system traffic in real-time using a single console. Detailed performance metrics help us monitor traffic volume, response times, and other key performance indicators.
- ***Visualization***. This capability enables us to see and understand the impact of system changes and perform root-cause analyses if there are problems. Further, we can clearly see all the service interdependencies, which are visually mapped out by the software.
- ***Flags and Alerts.*** Having a system that can alert us to unexpected conditions enables us to detect, diagnose and address system errors rapidly – whether they're service level violations or faulted decryptions.
- ***Virtualization***. Service virtualization enables us to aggregate internal services into a single, unified, composite offering for use by external parties including partners and members.

While these capabilities represent a vital and valuable foundation for our SOA efforts, it's our team's ability to enable and support agile business moves that represents the most significant payoff to the organization overall.

Business Agility and New Product Rollouts

One of the most compelling examples of our new business agility came when we launched our E-HealthKEY service. This service enables our members to store comprehensive personal medical information to a USB memory stick, which they then can attach to a key chain and carry with them at all times. It's a Windows-based desktop application that runs on a member's PC. The member plugs it in, the application loads up the data, and the member can manage health records on the PC – synchronizing all of his or her health data off our back-end system.

As a result of our investments in SOA and Web Services, we were able to roll this service out far more rapidly and effectively than would have been possible otherwise. E-HealthKEY consumes XML documents with the member's medical information. Orchestrations then parse that data out and route it to the right systems to update the member's medical record. This underlying support infrastructure enabled the rapid rollout of the service and has positioned us to continue introducing new ones. While present efforts have focused on managing inbound information, we will increasingly be focusing on securely distributing health information – ensuring that it's available in real-time at the clinics and labs where it's needed.

Our SOA infrastructure also lays the groundwork for others to OEM some of our offerings as Web Services. Other organizations can add our services to their own to enhance the value they bring to their customers. In this way, SOA and Web Services have brought our company and our industry partners new opportunities for growth.

It should come as no surprise that these kind of capabilities strengthen ties between IT and the business. Business leaders at MedicAlert want to be able to react more quickly and make our services more valuable. They want to bring on more members at a faster rate. Being able to respond more rapidly to business opportunities addresses their objectives – and now they realize that SOA can help the organization meet those goals.

This has changed how IT systems teams collaborate with the business. The department heads from our business development group, marketing and sales group, operations, and finance all get together when new objectives emerge. They lay out opportunities and business objectives. Typically, there are one or two representatives from the IT and the engineering side at these meetings. It's very informal. We work out feasibility, necessary resources, and timeframes.

In the past, IT often wouldn't get enough information or the right information from stakeholders to develop effective systems, as the classic requirements-gathering processes didn't work for MedicAlert. We move quickly and make decisions quickly. But we've learned to ask the right questions and get the right information to build systems that will advance the business. If you build your systems in relation to your business model, then you'll have more success with your SOA.

From a Web Services management perspective, we demonstrate our agility through rapid service rollout, versioning, and upgrades. We believe it's vital to introduce new versions of services seamlessly

Our SOA initiative has significantly changed the way we work and the impact that our IT teams can have on the business

to build confidence in and commit to our approach. For internal services, and with a service policy engine, we dynamically re-route requests to appropriate service versions, and transform requests and responses to maintain backward and forward compatibility between clients and Web Services.

By taking this approach and having the capabilities to deliver, our architecture and infrastructure teams have developed greater confidence and respect among the company's business leaders. Our company is now investing significantly to enable us to build and enhance our services-based architecture, our systems, and our talent pool further.

What we recognize is that success for us comes down to two things: scalability and agility. We must be able to scale the processes necessary to support the business. We must also be able to help the business create and respond to opportunities in the marketplace.

We could probably do all (or most) of the things we're doing now without SOA. But it would be very difficult and it wouldn't scale. It wouldn't be agile or adaptive in the least. It would take months to carve out each solution. SOA is not, at heart, about the technology. It's about making your business more agile – about being able to seize new business opportunities rapidly.

Lessons Learned

Our SOA initiative has significantly changed the way we work and the impact that our IT teams can have on the business. First of all, one has to realize that is just way too much for one person or one role to do. We are a small shop. We wear many hats. But now we are trying to give everyone his own hat. We have expanded the team, hired people with more specific skills. Our developers can now concentrate on developing. Web Services infrastructure specialists can concentrate on infrastructure. We have expert programmers who really understand the nuts and bolts of Personal Health Records. We have another set of engineers who really understand Web Services – how to make them work together, how to secure, deploy, and manage them. This deepening specialization enables us to be more concerned about the bigger picture.

To me, a critical aspect of SOA delivery is about making sure a Web Service is modeled and designed properly – that it's reusable, solution-agnostic, and can be actively versioned. Part of that challenge lies in defining layers of services. The lower you go in terms of service layers, the more reusable your services should be. The opposite is true as well. The higher you go in terms of service layers, the less reusable because they start to solve very specific business problems.

Another aspect of SOA delivery is the orchestration of business processes in the integration layer. Web Services become more useful when they're combined with other Web Services to provide the substance of a particular business process. You start to address bigger business problems – not just updating a database record, for example.

In addition, make sure you've established a solid plan for management and security before your launch your endeavor. It's all too easy to lose control of services if they're merely launched in an ad hoc, bottom-up fashion. This chaos will eventually undermine your ability to act with agility and dynamism. At the same time, poor approaches to security threaten to leave the operation vulnerable.

One also must draw on the perspectives, capabilities, and best practices of experts when possible. Select vendors with a proven ability to deliver in runtime conditions – with an eye toward their design-time contributions and capabilities as well. Check their customer references and industry partners. We made our vendor selections with great care and diligence, and we've been pleased with the outcome.

However, the most important thing we've learned in the course of our SOA initiative – something I try to convey to other architects – is that complacency is a career and organization killer. If you enter the world of SOA thinking you already know it, then you're going to fail. You must go into this with a very open mind. Architectural and development approaches that have worked in the past may no longer scale or perform at the levels that are now possible. It's critical to keep learning the craft and looking for opportunities to enhance the agility of the business. ■

About the Author

Jorge Mercado is the lead architect of MedicAlert's Software Architecture Group.

About MedicAlert

The MedicAlert Foundation is a non-profit healthcare informatics organization dedicated to providing services to our members that protect and save lives.

MedicAlert services are built around a repository of health information that enables members to manage their personal health records while maintaining security, privacy and confidentiality. As the trusted third party custodian of comprehensive personal health information, the MedicAlert® repository can connect to and provide critical medical information between patients, providers, payers, and first responders 24 hours a day anywhere in the world. The premium we place on patient confidentiality has earned the trust of 4 million members and the healthcare community worldwide.

MedicAlert is committed to providing technology-based solutions and is an active member and a leader in developing interoperability standards with all the major Healthcare IT standards organizations. The MedicAlert® repository uses Web service interfaces to support standard Electronic Health Records (EHRs), including electronic drug prescriptions and for patient record interoperability. These activities will ensure the rapid development and deployment of standards to improve the quality of care, lower healthcare costs while increasing patient safety.

MedicAlert is a nonprofit membership organization founded in 1956 with a mission to protect and save lives, is headquartered in the United States and has international affiliates in nine countries.

Fiorano SOA™ 2006

The Quickest path to an SOA

- **FioranoMQ™ 2006 –** world's fastest, most scalable JMS
- **Fiorano ESB™ 2006 –** CAD/CAM for distributed applications
- **Fiorano BPEL Server –** simplifying business process orchestration
- **Fiorano Tools –** BPEL Studio, Mapper, FEPO, etc
- **Fiorano Components –** 60+ pre-built adapters

Benefits

- **Adherence to popular industry standards** - JMS, COM, .NET, JCA, JMX, BPEL, SOAP, etc.
- **Multi-language, Multi-platform, Multi-protocol**
- **Unmatched Scalability and High Performance**
- **Quick, Measurable ROI**

Download your copy of Fiorano today!

www.fiorano.com/downloadsoa

Fiorano®

Enabling Change at the speed of thought

Service Versioning for SOA

Policy-based version control for SOA services

WRITTEN BY **MICHAEL POULIN**

(Found in a blog, "Versioning is as inevitable as security.")

SOA development practice isn't much different from other software development practices except for design and maintenance. Multiple self-containing and aggregated services that interact with others have their own lifecycle and evolution. The loosely coupling model of SOA services significantly simplifies design but creates additional difficulties in maintenance, especially in the interoperability of different service versions.

To better understand the requirements of SOA service versioning, let me ask several questions and see if we can answer them easily:

1. Is SOA a structure of interfaces such as Web Services or it is a structure of services with interfaces?
2. Who is the master in SOA – the client or service (provider)?
3. Is an immutable interface more important to a client than the service behind the interface?
4. What does a version of a service interface mean in case if it's backward compatible or if it's not?
5. Should a client know if the nature of the service has been changed behind its immutable interface?
6. If multiple versions of a SOA service are available, how can a client choose which one it wants or can use?

As you can see, I make a distinction between a service provider, calling it just a service, and service interface. If the service interface is a Web Service and we consider versioning, people ask what, actually, constitutes a new Web Service (WS) versus a new version of the same Web Service. Here I'll try to get some answers looking at the questions from the perspective of the SOA service client. I believe it will help us to "see the forest behind the trees."

Why Is Version So Important?

For some people this isn't a question at all. For others, it's a burden that's usually skipped in the development phase and a "sudden" nightmare during the maintenance. Let's analyze a real-life example to validate the importance of versioning in a distributed Service Oriented Architecture.

Following best practices, we had a SOA service with a coarse granular Web Service interface at one of my previous jobs. The Web Service transferred XML documents that defined "commands" (recall the well-known Command Pattern). The service provider promised to log an audit message for every command received for our review.

The provider's team wanted to upgrade its audit database and needed to block the insertion of messages for a while. The service provider built a temporary service without a logging function and silently substituted the service component under the same interface. As it happened the upgrade took longer time than planned and a lot of commands were done without audit logging. We found the problem during our audit reconciliation process but it was too late.

Now, assume that a version control is in place. A substitute (new) service component might be released only under the new version. The version-controlling utility would have to recognize this fact and either block communication or immediately notify the client about

the change. You'd say that such version control defeats the interface concept of Web Services and directly affects the client even when an interface isn't changed. And you'd be right. However, in our example the Service Level Agreement (SLA) was violated by the service provider, i.e., the business of the client was affected, though the interface was preserved.

Thus, the OO practice of developing an immutable interface applied to a SOA can easily lead to a business problem. Versioning is one of the cheapest mechanisms to avoid such problems. It's not versioning the interface (a SOA service isn't an object and it has a nature, which is not addressed in object-oriented architecture) but versioning the contract between client and provider. That is, version control works for the client interests and SOA service has to honor it.

What Can Be Versioned?

There are two approaches to versioning: down-to-top and top-to-down. The former is more familiar to developers and the latter is usually used by architects and managers.

In the down-to-top approach, the attention is put mostly on the interface versioning, in particular, the Web Service. The OASIS WSDM-MOWS 1.0 draft specification says that Web Service's description, interface, service, and endpoint can be separately versioned and defined in their own individual namespaces. It's interesting to note that the final 1.0 release (as well as the 1.1 draft) don't have a version control section. That is, separate versions of Web Service's parts may be not a good idea after all.

As we know, a Web Service is defined by its WSDL. The WSDL 2.0 (draft) gets Web Services closer to the notion of a SOA service due to a new component-based concept, operational stile with restrictions and a Feature option. Following the spirit of WSDL 2.0, particular combinations of versions of WSDL elements can constitute an overall version of WSDL. Another combination of element versions leads to another version of WSDL. Unfortunately, even an overall version of WSDL doesn't answer the old question about a new version of the WSDL versus a new WSDL (i.e., a new Web Service).

For example, consider a Web Service in a document/literal style where XML Schema is imported for the message. Changes in the XML Schema don't reflect in the WSDL. So, if changes happen in the XML Schema, do we get a new Web Service? If changes are optional (e.g., new elements with minOccurs="0" is added), is it really a new Web Service for the client? There are too many questions to this version model; that's why I call them "trees."

In the top-down approach, two things are versioned: the service component and the service interface (e.g., the Web Service). The WSDM-MOWS 1.0 draft also recognizes the version of the service component and calls it a revision. According to the specification, "Revisions are related to each other via changes that happened between revisions. Each revision will be associated with a versioned component. Each change indicates a predecessor and successor (if any) revisions. Each change may aggregate multiple change descriptions."

Actually, a service component may have its own lifecycle even outside of the service realm. That is, component versioning is a standalone issue. So a compound SOA service version consists of a combination of an overall version of the service interface and a version of the service component. Our task is to come up with a definition and structure of a compound versioning model.

I've found that a single compound version concept is argumentative even for some people who agree with the aforementioned reasons. They usually refer to the "realistic requirements" stating "people care about particular method [signature] changes rather than changes in other methods." This results in dealing with individual method versions. They also refer to the practice in Java programming where the object version has "low importance" to the "people" in comparison to the version of the object's API. It's interesting though, I'm afraid, a misleading approach.

First of all, the "people" are mostly developers, not users of the service, i.e., service development cycles are the center of attention, not maintenance and integration with the clients. We, on the contrary, are concentrating on the service "face" to the clients. Second, if we follow that "realistic" approach, why bother with multi-method services when a single-method model is much simpler? The aggregation of versioned single-method services can be viewed as a container, like EJB or JMS containers, with no overall versions at all. Third, it's not quite clear (to me) what methods are meant in a case where the service interface is a Web Service in document/literal style. Are we going to version XML elements in the message? If, instead, the message is defined by a version of the XML Schema, the latter addresses multiple methods/elements together. So, we get back to the single version for all methods/elements, don't we?

I have a strong feeling the proponents of per-method versioning are trying to screw SOA services into the object-oriented realm. SOA is not a structure of RPC calls in OOA; Web Services and SOAP are no more than SOA-enabling elements but developers and vendors frequently overlook this fact. One of the major principles of SOA is business agility, i.e., agility with business processes and functions. So SOA service versioning has to be adequate to the SOA concept while the details of the interface versions or service component versions are taken care of by the service provider (e.g., via a hidden mapping of the compound version to the service elements' versions).

Compound Version Identifier

To simplify the observation of the compound version identifier (CVI) structure, let's discriminate between the version visible to the clients and the "assembly" of the versions of the individual service's components, interfaces, and elements. Nevertheless, considering the complexity of the SOA service internals, I'd like to propose that the following CVI structure be available to the services' clients (that include other services as well):

```
<srv>.<nbc>.<bwc>.<rel>
```

where

- ***srv*** is an element reflecting the major version of the service as a whole. Changes in this element represent significant changes in the service lifecycle that may be not backward compatible. For example, a security entitlement service adds a control at the level of the individual application function; it doesn't necessarily mean that access to the application has been changed but the final result may be changed without backward compatibility for particular clients;
- ***nbc*** is an element that represents a version state of the major version that's is not backward compatible for some or all of the service functions. For example, one previously deprecated function has finally been removed in that version while other functions were unchanged;
- ***bwc*** is an element indicating an extension or modification in service functionality. It's strictly backward compatible. For example, new functionality like a new type of message or a new method has been added that doesn't affect clients using old functions;
- ***rel*** is an element showing little backward-compatible changes like bug fixes. It may also be a release version correlated to the build and/or a source save repository such as ClearCase, CVS, and the like.

How this version structure helps clients recognize whether a backward-incompatible change affects them? Moreover, if the change is backward compatible or unrelated to the functionality used, should the client continuously modify connectivity code to assimilate a new version of the service? The answers to these and similar questions are in the version control procedure defining how the CVI can be used in conjunction with a client-provider contract.

Controlling Version Compatibility

As I mentioned, we're looking at the SOA service versioning from the client's perspective. The version control procedure may be easier understood if we assume following principles:

1) A client engages a SOA service on the basis of a Service Level Agreement. That is, a SOA service represented by, for example, its Web Service interface, is invoked not when WSDL is available but when the client is fully aware of the service behavior and conditions. The latter includes a service version model and the corresponding interpretation of currently available service functionality. I'm happy to note that Thomas Erl also supports the concept of a versioned SLA that may include separate information about the endpoint, service, and interface. I discussed SOA service SLA in another article published in *SOA Web Services Journal* (vol. 6, issue 5).
2) The compatibility between client needs and service functionality is expressed via a Version Control Policy (VCP). It's a set of rules that decide whether a particular version of the service is adequate to the client's needs and can it be used according to the SLA. The simplest case of the VCP is a single-rule policy saying that the service version has to be matched (be equal) to the version identifier represented in the client's request. You, probably, recognize the traditional API version usage in this case.
3) The service version change doesn't automatically modify the VCP. To accumulate a backward-incompatible version change, the rule(s) in the VCP have to be explicitly changed. It's assumed that the VCP is written in a way that it can automatically cover backward-compatible version changes.

For the sake of simplicity, let's assume that the version elements are numeric. Other syntaxes are also possible but require additionally defined "operations" of the version element values. So, our example of a service version is 1.3.12.123456789045678905678 and the client's VCP rules are:

- ***Rule 1:*** IF srv 'EQ' 1
- ***Rule 2:*** IF nbc 'LE' 2
- ***Rule 3:*** IF bwc 'GE' 9
- ***Rule 4:*** IF rel 'GE' 123456789045678900001 where 'EQ,' 'LE,' 'GE,' 'LS,' 'GT,' and 'NE' are Boolean expressions such as "equal," "less-and-equal," "greater-and-equal," "less," "greater," and "not-equal" respectively.

The VCP states, for example, that the service version is compatible with the client's needs when all rules are satisfied. In the example, Rule 2 is violated. This means that:

1) The client may not use the service of a particular version;
2) The client's VCP has to be reviewed and rule(s) have to be adjusted to accept the service version. It's important to notice the adjustments may be done only if the new version meets the client's business requirements.

In some cases, the client's business requirements get into conflict with new service version and the client should continue using the previous version of the service. At the same time, if the client team knows what business features are to be provided in the coming/future versions of the service, the VCP can be set in a way that permits service invocation even when the new versions aren't backward compatible. For example, Rule 2 may be redefined as nbc 'LE' 5. That is, the current and two consecutive service changes that aren't backward compatible meets and will meet the client's business requirements. With such a VCP, the client can work without interruption even when versions change.

Thus we deal with the service version and the VCP. The service version belongs to the service; it may be accessed at the point of service announcement/registration. For example, for the service with a Web Service interface, it may be a UDDI; for a RMI interface, it may be an RMI registry; for other services, it may be JNDI/LDAP, or the CORBA directory service, or Microsoft Active Directory, or others. The VCP belongs to the client. To control policy preservation, we need a policy enforcement component (PEC). The PEC belongs either to the client or to the point of service invocation/announcement/registration. Irrespective of location, a client can register its VCP for a particular Service with a PEC.

Hopefully, you'd agree that SOA service invocation from a business perspective is a little bit more complex than just WSDL discovery with this or that version of the endpoint. To simplify a SOA service control, the PEC could always be set at the announcement/registration point and might be implemented as a PEC service. If a request for service returns something like a "not available" exception, it means there's no version of the service meeting the related VCP; the client might have to evaluate/consider a new version and adjust the VCP.

An independent PEC-based approach differs from the so-called "Service Versioning Covenant" model. The latter assumes that every service has its covenant, which governs all coming requests (messages and method calls) with regard to version compatibility. The covenant controls whether the request is done for the available version and simply refuses mismatching requests. The client doesn't have information about what version of the service component is behind the interface. The consequences are: the performance of the client requests degrades due to repeated controls, the client never knows if the next request will be denied due to a change in the service version, the client never knows if the new service version is suitable for it. Moreover, since every service can have different cycles for version changes, a service composition (a structure of services working with each other according to a scenario) may become broken at any moment. Thus, such a service-centric version control appears inefficient.

Version Control Layout

Figure 1 shows two ways of controlling the version of the service. The first way the service reference is obtained from the Service Registry and invoked. The PEC intercepts every invocation call. The PEC has to acquire the service's version from the service provider, apply a related VCR, and decide if an invocation is permitted. Since all service providers are free to define their own version models and mechanisms for accessing version information, the PEC has a quite challenging task. Plus, an in-call control degrades call performances.

The second way the PEC works in the Service Registry. The PEC validates the version information of the service at discovery time and returns the first version that satisfies the VCP. It's important to note that version control doesn't substitute for a functional search. If a functionally suitable service is found and the client has a related VCP, the PEC challenges the service version. However if the client doesn't have a VCP for the service, it's not recommended to engage

Sound Architecture Requires Proper Planning

WEB AGE SOLUTIONS - YOUR TRAINING PARTNER FOR SOA

In all phases of SOA migration, Web Age Solutions provides training and customized services from awareness to implementation. We support vendor specific or generic SOA training tailored to your organization's needs.

Custom training for complementary SOA technologies

XML • WEB SERVICES • WSDL • SOAP • WEBSPHERE • WEBLOGIC • JBOSS • J2EE • SPRING/HIBERNATE/STRUTS • WID/WBI/WMB

Custom training plans for virtually every job role

BUSINESS ANALYST • ADMINISTRATOR • DEVELOPER • ARCHITECT • QA/TESTER • MANAGER • EXECUTIVE • SECURITY

Consulting services for all phases of SOA migration

IBM Premier Business Partner

Add Web Age Solutions to your plan & stay ahead of the competition

www.webagesolutions.com - 877.517.6540 - soa@webagesolutions.com

WEB AGE solutions inc.

the service based just on functional matching because of unknown business and technical risks.

Executing the PEC at the service registry level allows for the unification and formalization of the compound version model and version control mechanism. Unified rules and version formats can be shared between all users and vendors of the service registry. Validation of the version before service invocation works for in-call performances and prevents wasting resources on incompatible calls. So, let's see what we can do for a service registry to represent a compound service version.

Finding the Version of the Service

Released SOA services are usually registered in different registries as mentioned. For example, assume we have an EJB-based service. The service has two interfaces – Web Service (J2EE 1.4) and RMI/IIOP; the Web Service interface is registered in a UDDI and the RMI/IIOP is registered in a JNDI directory. To a client it's the same service providing the same business function so there should be an interface-independent service version available in both the UDDI- and JNDI-based registries.

As we know, UDDI has a special structure that permits registering the interface, usually defined by the <overviewURL> element, separately from registering the service component defined by the <businessEntit> element. Binding between the service component and its Web Service interface is specified in the <bindingTemplate> element, which is hidden inside the <businessEntit> element. Such a model causes discovery issues not only for the Web Services themselves but appears as an error-prone approach to SOA services because it allows registering the interface without registering the service component. Looking at a service registry from the client's perspective, the service features and the SLA are the primary interests while the service interface and the vendor name are secondary. Unfortunately, we don't have a real alternative to UDDI so far.

Using a UDDI taxonomy, i.e., its tModel concept, we can create and, via the UDDI API, register as many version entries as we need for a particular SOA service. The version entry may carry additional information about its state like "current" or "deprecated." Every vendor can specify its version namespace and use "version name" and "version information" accordingly. Since a VCP is created for a particular service, the client is aware of related version namespace and the version information can be actively utilized. Listing 1 is an example of a version namespace definition in a UDDI.

When the service version is in the UDDI, both the service component entry and its interface entry have to refer to the "uuid: whatever_key_for_theCompanyName_version_entry" and specify their version in the <keyedReference> element of the <categoryBag> element. Listing 2 offers an example of a version reference in an interface entry.

The definition of a version namespace allows a vendor to provide a strong naming convention regarding the keyName and keyValue. Until it is standardized, we can expect different vendors to define the different version namespaces the clients have to deal with differently. At least, for one company, the version namespace may be unified at the enterprise level and simplify the life of enterprise clients.

In spite of the version namespace definition, we inherit the UDDI problem and encounter the risk of having only one entry with a certain version value instead of two – for the interface and for the service component – in the service registry. I suggest this risk can be mitigated by a PEC service until a UDDI provides an appropriate control.

Talking about EJB and JNDI, we have to distinguish between

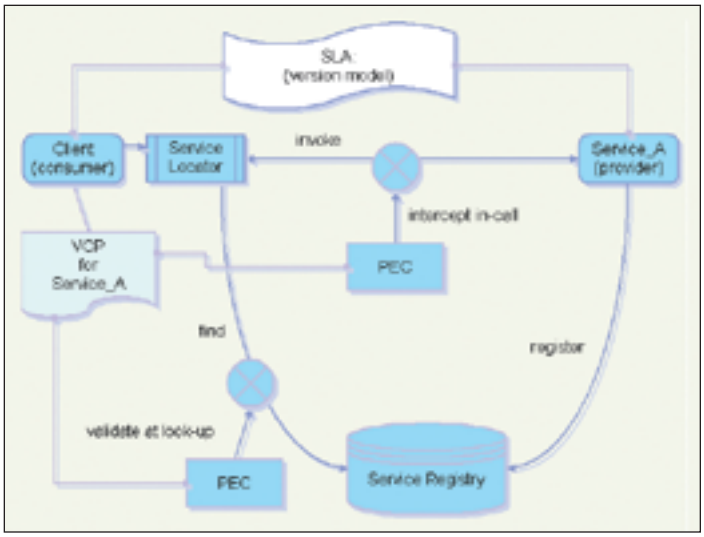


Figure 1: Examples of SOA service version controls: in-call interception and look-up validation

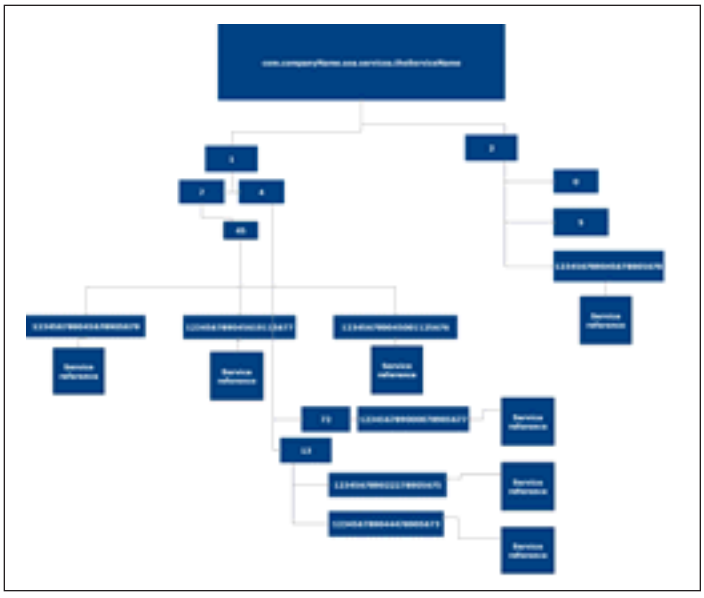


Figure 2: An example of a version name hierarchy for a directory like JNDI or Active Directory that the Service Reference can be bound to

service and application types of the versioning models. For applications, version control can be done at the build and deployment stages via, say, Manifest files specified in the application archives (Graham Hamilton of Sun Microsystems offered more information about versioning and dependency mappings for Java applications at the last JavaOne conference). For services, we have no more choices than we discussed before – controls on every invocation or controls at the lookup of the EJB reference (Home interface). Since the former may be ineffective, we'll concentrate on the latter.

The deployment descriptor of an EJB or an RMI service defines a JNDI name that the component reference is bound to. Looking at the name structure, one can easily find that a JNDI name represents a hierarchy of names. This means that a lookup operation performed, for example, by a Service Locator module (see Service Locator pattern) can explicitly navigate along the name-tree instead of making a single call on the composite JNDI name. So the JNDI name may have an immutable version part and a changeable

version part. The Home interface or service reference binds to the changeable version part, which is a child of the immutable service part in the naming hierarchy. The example in Figure 2 demonstrates the version name hierarchy for the following versions:

immutable version part	*changeable version part*
com.companyName.soa.services.theServiceName	*1.7.45.123...8905679* *1.7.45.123...9115677* *1.7.45.123...1125676* *1.4.72.123...8905677* *1.4.13.123...8905675* *1.4.13.123...8905673* *2.0.05.123...8905678*

Thus, doing a lookup operation, the client's Service Locator represents the immutable service part of the JNDI name to the PEC. The latter navigates to all the children of the immutable service part – the names in the changeable version part – combines possible version expressions, and challenges them against the VCP. The service reference bound to the first found 'matching' combination of version names is returned. The client's code can cache the reference and reuse it in consecutive RMI service invocations or for narrowing the Home interface reference to the EJB Object interface references.

If a service provider discards an RMI service, the service invocation returns an exception. It's recommended to make at least one attempt at obtaining another RMI reference under the previous VCP before reporting an error (because the service may be redeployed). If the EJB is redeployed under the same JNDI name, narrowing always returns the current reference to the Object interface transparently to the client. If the service invocation fails under the same VCP, it may mean that the previous version isn't supported anymore. So the client has to do a business review of the new version of the service and, if possible, adjust its VCP.

Conclusion

This article promotes the idea of a SOA client viewing a SOA service as a monolithic entity with an easily interpreted compound version. A compound version can hide multiple versions of service components and separate versions of elements of the Web Services representing the service interface. The proposed version model and an approach to the version control based on the client's version policy open a way for flexible version management even for changes in the SOA services that aren't backward compatible. Compound version specification is illustrated for a UDDI and a directory structure. The example shows two ways of applying the policy enforcement mechanism and promoting policy control at the service registry for service lookup operations.

References

- *UDDI Core tModels.* www.uddi.org/taxonomies/Core_Taxonomy_OverviewDoc.htm
- *Liang Yu. "Understanding UDDI's tMode."* www.codeproject.com/soap/understandingTModels.asp
- *Kyle Brown and Michael Ellis. "Best practices for Web Services Versioning."* www-128.ibm.com/developerworks/webservices/library/ws-version/
- *Chris Peltz and Anjali Anagol-Subbarao. "Design Strategies for Web Services Versioning."* http://webservices.sys-con.com/read/44356.htm
- *Kirk Allen Evans' Blog. "Versioning Web Services."* http://blogs.msdn.com/kaevans/archive/2004/12/03/274271.aspx
- *Heather Kreger. "A Little Wisdom about WSDM."* www-128.ibm.com/developerworks/webservices/library/ws-wisdom/
- *Thomas Erl. "The Principles of Service-Orientation Part 2 of 6: Service Contracts and Loose Coupling."* http://searchwebservices.techtarget.com/tip/1,289483,sid26_gci1171966,00.html
- *WSDM-MOWS 1.0 specification, Draft.* http://www.oasis-open.org/committees/download.php/5664/wd-wsdm-mows_versioning_change_2.23.04a.doc
- *Don Smith. "[Web] Service Versioning Guidance."* http://blogs.msdn.com/donsmith/archive/2006/01/31/520338.aspx
- *Rocky Lhotka. "A SOA Versioning Covenant."* http://www.theserverside.net/news/thread.tss?thread_id=33434
- *Service Versioning.* http://orchestrationpatterns.com/serviceVersioning?PHPSESSID=4dee379138a5d1f6b59e02468f2ef0ec
- *Dragos Manolescu and Boris Lublinsky.* SOA Enterprise Patterns - Services, Orchestration and Beyond. *Book Draft.* http://orchestrationpatterns.com/files/ServiceVersioning.pdf
- *Java Futures at JavaOne:* http://www.theserverside.com/news/thread.tss?thread_id=40569

About the Author

Michael Poulin works as a consulting technical architect for leading financial firms. He is a Sun Certified Architect for Java technology. For the past few years, Michael has specialized in distributed computing, SOA, and application security.

Listing 1

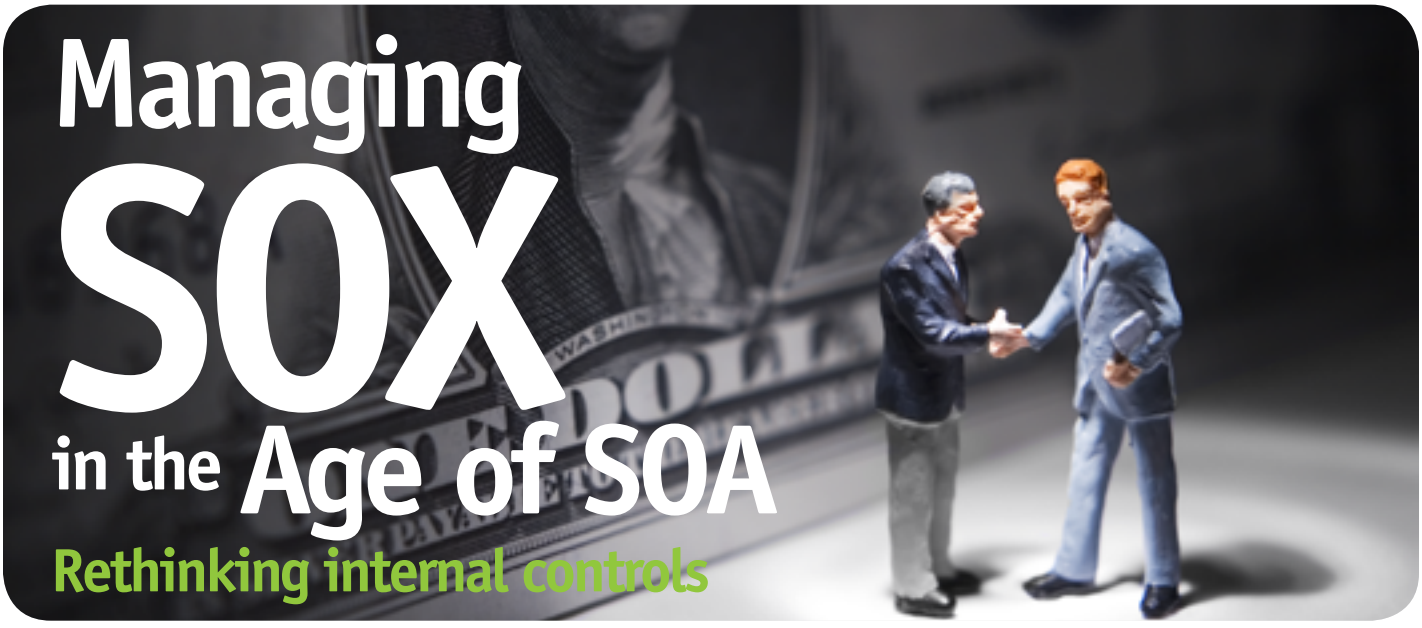
```
<tModel tModelKey="uuid:whatever_key_for_theCompanyName_version_entry">
   <name> theCompanyName_version_tModel</name>
   <description xml:lang="en">definition of theCompanyName_ver-
          sion_tModel</description>
   <overviewDoc>
      <description xml:lang="en">version of SOA Service for theCom
          panyName </description>
      <overviewURL>http://www.theCompanyName.com/releaseDocuments/
          versionDetails.html</overviewURL>
   </overviewDoc>
</tModel>
```

Listing 2

```
<tModel tModelKey="uuid:5DD52389-B1A4-4fe7-B131-0F8EF73DD175">
   <name> theCompanyName ServiceName interface</name>
   <description xml:lang="en">Web service interface for the Service-
          Name of  theCompanyName</description>
   <overviewDoc>
      <description xml:lang="en">The WSDL definnition</description>
      <overviewURL>http://www.theCompanyName.com/public_services/in-
          terfaces/web/serviceName.wsdl</overviewURL>
   </overviewDoc>
   <categoryBag>
      <keyedReference
        tModelKey="uuid:whatever_key_for_theCompanyName_version_entry"
         keyName="version"
         keyValue="1.3.12.123456789045678905678"/>
      <keyedReference
         tModelKey=" uuid:whatever_key_for_theCompanyName_ver-
          sion_entry"
         keyName="status"
         keyValue="current "/>
   </categoryBag>
</tModel>
```

Managing SOX in the Age of SOA

Rethinking internal controls



WRITTEN BY **HUGH TAYLOR**

Service Oriented Architecture (SOA) is at the heart of many major IT initiatives and vendor offerings. However, while SOA has the potential to deliver business value through streamlined application integration, as well as integration with partners and suppliers, the open nature of SOA has the potential to cause problems with Sarbanes-Oxley compliance. This article will look at compliance issues inherent in developing an SOA. Using a practical example, we'll examine COSO Control Objectives, Risks, and their supporting IT systems from the perspective of Sarbanes-Oxley compliance.

This article is meant to help IT professionals, corporate managers, and auditors understand two complex and interconnected sets of activity in the world of corporate computing: Sarbanes-Oxley (SOX) and SOA. Both SOX and SOA are emerging as major areas of focus – some might say distraction – for a growing number of people involved in information technology, management, and audit.

Familiarity with the origins and intent of the law will help you understand why the Sarbanes-Oxley Act is relevant to IT professionals at a public company. Congress passed SOX in 2002 to calm the financial markets after Enron, Adelphia, and Worldcom. To assure investors that the financial statements that public companies make are accurate, SOX expanded the reporting and disclosure requirements concerning their internal financial controls, the process, practice, or structure designed to provide a reasonable assurance of the reliability of financial reports.

Internal controls can be either *preventive* or *detective*. A preventive control prevents fraud or errors that can result in a misstatement of financial results. A locked cash register is a simple example of a preventive control. A detective control enables an accounting staffer or auditor to check to see if a financial statement, or a supporting piece of data for a financial statement, is correct. Bank statement reconciliation is an example of a detective control.

SOX Sections 302 and 404 mandate that a public company documents and tests its internal controls. Management must then certify that the company's internal controls are effective. Then, an external auditor must also test and certify them.

The Public Company Accounting Oversight Board (PCAOB) has directed public companies to adhere to the internal control framework known as COSO in their SOX 404 compliance. The COSO framework pairs risks with control objectives and control practices to provide a level of confidence in a company's internal controls. If they are not effective, the company must disclose the deficiency, which can cause problems with the SEC and others.

If you're involved in IT and SOX then you should understand that you're working on showing that IT supports the COSO control objectives intended to mitigate the risk of financial misstatement. The purpose of your work is to help the company comply with SOX 404 and 302 by establishing, documenting, and testing the effectiveness of IT systems that support COSO Control Objectives.

IT's Place in Internal Controls

Because so much of business today is done using computers and software, IT plays a prominent role in internal controls. Underscoring that point, Gartner reports that 97% of the material weaknesses in internal controls can be mitigated through IT. In practice, there are two essential ways that IT finds a place in internal controls:

1) The IT General Controls as recommended by COSO
2) IT as a component of a non-technological internal control over financial reporting (often an application-level control)

Now we'll look at each of these categories using the example found in Figure 1, which depicts the IT architecture used by a public company. It shows the systems and software applications necessary to process inbound, revenue-producing transactions. While the corporate general ledger system is responsible for financial reporting, much of the supporting data regarding the transactions and inventory comes from two connected systems: A mainframe-based warehouse management application and a customer portal.

IT General Controls

There are numerous IT General Controls. To stay focused, we'll only look at one example – "Control Objective: Controls provide reasonable assurance that financial reporting systems and sub-systems are appropriately secured to prevent unauthorized use, disclosure, modification, damage, or loss of data."

With regard to this control objective, in the context of the architecture shown in Figure 1, the internal auditor would have to document and test the effectiveness of the internal controls that secured that architecture. Specifically, the internal controls would have to prevent unauthorized access to the General Ledger system, the Warehouse system, and the Customer Portal. The internal control would have to establish rigorous password protections, firewalls, hardening guidelines, and so on to assure the auditor that the systems in question were "appropriately secured." We'll return to this point later when we introduce the idea of Service Oriented Architecture.

IT Supporting Non-Technological Controls

Many internal controls over financial reporting are not technological in nature. For instance, subjective valuation of some balance sheet assets usually involves manual processes. However, many of them rely on IT for their effectiveness. Using the COSO framework, an internal control for the company depicted in Figure 1 might look like the pairing of control objective, risk, and control practice shown in Table 1.

Following the COSO framework virtually all internal controls are expressed in the format shown in Table 1. Of course, in reality the details might be different or more specific in any given situation, but the principles apply. Internal controls over financial reporting set out a control objective intended to mitigate a risk using a control practice.

Although the internal control described in Table 1 is procedural in nature, and may in fact be entirely manual, it's likely rooted in IT. In our Figure 1 example, there must be a reasonable level of certainty that the general ledger system is receiving accurate, timely data from the warehouse system and the customer portal. The IT department may be called on to document and test these technological factors that support this procedural control.

Problem Scenarios

If the control isn't effective, the company faces a risk that the control objective, "Accurately record invoices from all authorized shipments" won't be met. If this control is deficient to the point that it could cause a material misstatement of financial results – a "material weakness" in internal controls – then the company could be in real trouble. If a public company discloses a material weakness in internal controls under SOX and fails to remedy it, consequences can include SEC investigations, sanctions, and even delisting from exchanges.

Let's look at an example of what could go wrong. Material weaknesses usually manifest themselves in fraud. Consider the practice known as "channel stuffing." Channel stuffing involves creating bogus revenue by colluding with customers. To earn a high bonus, an executive might ask a customer to place a large order on December 28. The revenue is booked for the year, but on January 2, the goods are returned. This device might seem obvious, but it happens all the time and it can be quite hard to detect or prevent in a large, complex organization.

If the company doesn't have effective internal controls over invoicing and inventory and the IT systems that support those controls then it's more vulnerable to the risk of channel stuffing than it would be if it had robust controls. The channel-stuffing example also highlights one of the key principles of internal controls over financial reporting, which is the segregation of roles. It's usually required that one individual, such as a salesperson, can't be able to book a sale, take possession of the merchandise, request shipping, and book the revenue into the general ledger. A fraud such as channel stuffing is much harder to prevent or detect if role segregation isn't practiced as one of the internal controls.

Consider then, what happens, when the architecture is opened up as an SOA.

Internal Controls in a Transition to SOA

If the company described in Figure 1 transitioned to a Service Oriented Architecture (SOA), its IT architecture would resemble the one shown in Figure 2. What's different? Well, where before the company relied on a proprietary interface to connect its systems with one another, they can now exchange data and operating instructions using the open standard of Web Services. The company has also taken advantage of the universal "machine to machine" interoperation capability of SOA and enabled its customers to have direct programmatic access to its ordering systems. Instead of a portal, the company now has a Customer Web Service hub to which customers can connect directly using their ERP systems.

Control Objective	Risk	Control Practice
Accurately record invoices from all authorized shipments.	Missing documents or incorrect information.	Invoiced amounts are properly recorded as to account, amount, and period.

Table 1: Control objective/risk/control practice pairing

SOURCE: INTERNAL CONTROLS PRIMER. KARL NAGEL & CO.

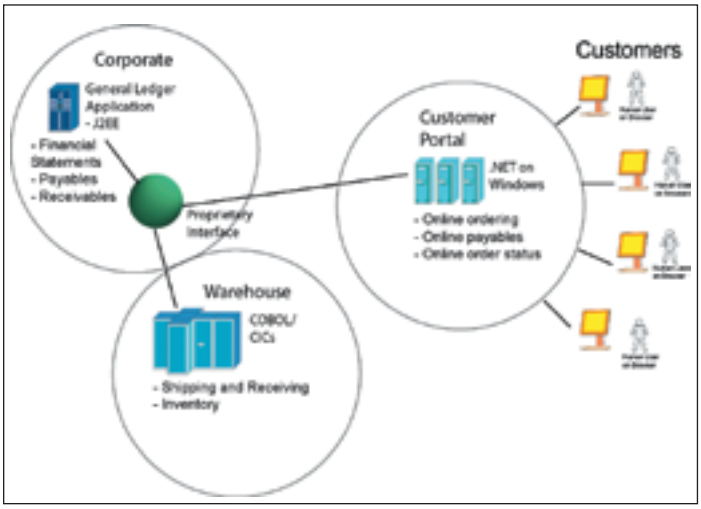


Figure 1: A public company's enterprise architecture for outbound transactions (revenue)

SOA's Impact on Internal Controls

While SOA may be a boon to business executives owing to its inherently flexible nature, this new architectural paradigm can cause difficulties for the IT side of SOX-mandated internal controls. There are several major areas of concern outlined below.

A New Level of Openness

Because an SOA is built on open standards, it can expose critical data and application functionality to a vast new array of users. Any effective set of internal controls over financial reporting that relate to applications in an SOA must take this new level of openness into account. In the example shown in Figure 2, the internal controls must consider the risks inherent in exposing the data in the warehouse system, general ledger, and customer hub to unauthorized access. For example, a SOX auditor may want to test the controls over the integrity of inventory documents that support the inventory asset figures in the company's balance sheet. To certify that the control is effective, the auditor will probably want to see documented evidence that access to the software that generates these inventory reports is restricted to authorized personnel. The open nature of SOA creates the added challenge of establishing and testing this kind of internal control.

Machine-to-Machine Security

The fact that Web Services, the fundamental building blocks of most SOAs, are based on machine-to-machine interactions creates another internal control hurdle for IT professionals involved in SOX compliance. While not a revolutionary shift, the machine-to-machine nature of SOA changes the nature of many existing internal controls that assume that the user of a given application is a person.

Many standard internal controls in place today involve the authorization and authentication of specific individuals and their right to access financial applications and modify the data in those applications. In the age of SOA, the focus has to change to accommodate the reality that many of the new "users" of financial applications are in fact other applications that can't be authenticated or authorized using a traditional identity store or access management system.

In the example shown in Figure 2, the shift to SOA has changed the nature of the customer's interactions with the company. Before, specific individuals could log onto the customer portal and transact business with the company. Internal controls related to revenue recognition, as depicted in Table 1, were based on a process of authenticating and authorizing those individual users against an identity store that was under the company's control. In the new SOA, the "users" of the customer hub are actually the customers' ERP systems. There are people using those ERP systems, of course, but there has to be a way for the company to authenticate and authorize those users before granting access to financial applications that have been exposed as Web Services. If there is no such authentication or authorization going on, then the open access to financial systems by unknown persons working through an ERP system at another company would probably result in an internal control deficiency.

Segregation of Roles

Segregation of roles is a core technique of internal controls over financial reporting. Continuing with the machine-to-machine authorization issue described in the previous section, note that it may be impossible to establish clear role segregation in an SOA. Why? If the "user" of a Web Service-exposed financial application is actually another application, but the internal controls use role-based authorization for a human user, then the control will be deficient.

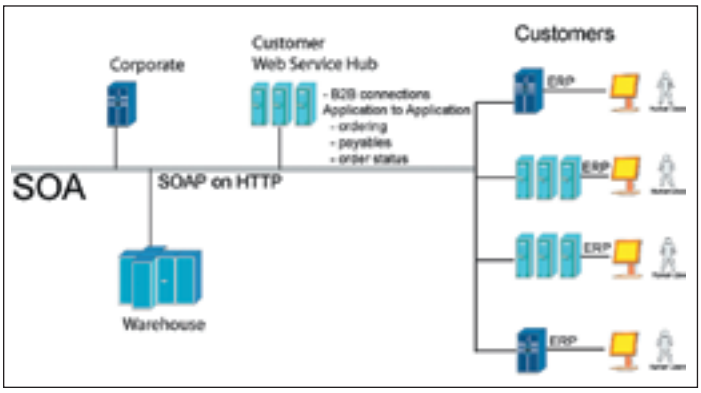


Figure 2: Transition to a Service Oriented Architecture

For example, in Figure 2 a sales rep shouldn't be able to access the general ledger and create a sale that would give him a bonus or access the warehouse system and move inventory around. The potential results of such role-based control lapses are error and fraud. If the sales rep can access those systems using a Web Service-consuming application on the SOA that doesn't authorize him directly, then there can be trouble. In the transition to SOA, those responsible for internal controls involving financial systems need to evaluate whether or not they are addressing the potential for deficient control over authorization and role segregation.

No Perimeter Emphasizes Application Controls

Overall, the move to SOA puts greater emphasis on application level controls than may have been required in a conventional IT architecture. While many of the IT general controls focus on the perimeter – firewalls, network access, passwords, baseline standards, and so on – the SOA renders much of perimeter security irrelevant. If access to critical financial applications is open to direct use by virtually any application in the world, then the perimeter is necessarily less significant as a component of an internal control practice.

Conclusion

Service Oriented Architecture requires some rethinking of internal controls over financial reporting. In terms of IT general controls, SOA changes some of the underlying assumptions that exist today, including the importance of the perimeter and the role of individual users versus machine users of critical applications. For IT systems that support non-technological internal controls, the transition to SOA should stimulate analysis regarding access rights, segregation of roles, and integrity of data.

The good news is that SOA represents an incremental shift in the IT aspects of internal controls and Sarbanes-Oxley compliance. SOA is not a categorical revolution in technology that shatters previously understood notions of internal controls.

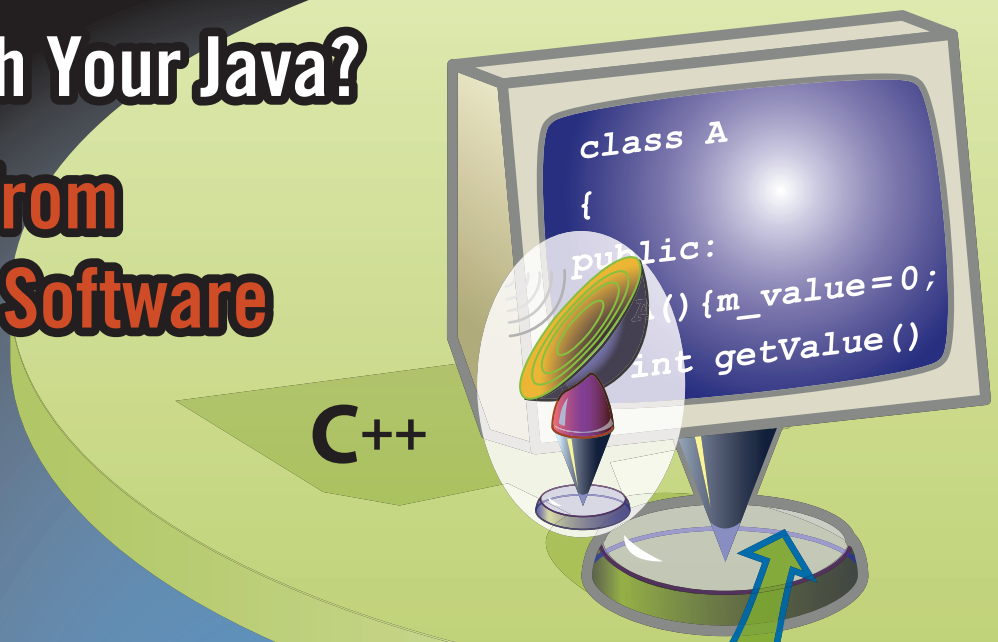
However, one thing should be clear: A poorly governed SOA could easily result in deficient internal controls and problems with Sarbanes-Oxley compliance. ■

About the Author

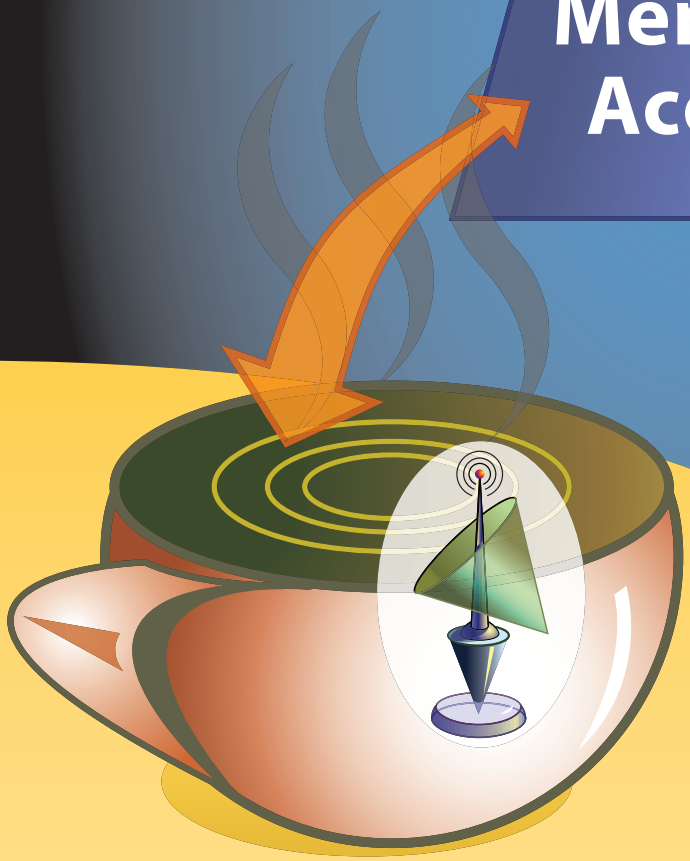
Hugh Taylor is vice president of marketing communications at SOA Software, the provider of management and security solutions for enterprise Service Oriented Architecture. He is the co-author, along with Eric Pulier, of Understanding Enterprise SOA *(Manning, 2005) and* The Joy of SOX *(Wiley 2006). The author of more than a dozen articles and papers on the subject of Web Services and Service Oriented Architecture, Hugh is an authority on business process management, SOA, and compliance issues. He got his BA, magna cum laude, from Harvard in 1988 and his MBA from Harvard Business School in 1992. He lives in Los Angeles.*
hugh.taylor@soa.com

Take C++ with Your Java?

HydraSDO™ from Rogue Wave Software



Shared Memory Access



Java

ROGUE WAVE SOFTWARE
A QUOVADX DIVISION

Free Evaluation

www.roguewave.com/developer/downloads/

Business Rules Café

How IT Can Stop Worrying and Love Change

(Our Apologies to Stanley Kubrick)

WRITTEN BY **JAMES TAYLOR**

One of the hardest things for most IT departments is change. Not only do they have to cope with the technology change that is inherent in their business, they have to cope with all sorts of other change – regulatory changes, business changes, competitive changes, requirement changes, process changes, policy changes. All this change creates a maintenance nightmare so that in many IT shops most of the time is spent not in building cool new applications but editing and "fixing" code in old systems. Business rules, one of the fastest-growing markets in application development technology, offers a way to stop worrying about change and learn to love it.

Change Is Inevitable

So why is there so much change? Well, the nature of IT inflicts a certain undercurrent of change in everything an IT department does. Systems get faster, IT standards come and go, new languages and methodologies become popular, magazines are filled with great new tools and ideas for building systems better. By and large most IT people love this kind of change – it's fun, it's often why they got into IT in the first place. This might be considered "good" change.

The problem is that this is not the only kind of change. IT people, like most people, like to "finish" things and get the rewards and kudos that come with it. But constant change in the business environment can make it impossible to finish applications or at least create an environment where users feel that the system is never quite right – by the time each version goes live there are already new requirements and changes required. So what causes this kind of "bad" change?

Regulations

In any regulated industry – banking, insurance, healthcare – a key risk is changing regulations. Most state legislatures love to tinker with the rules as does the federal government. If your system has to enforce these regulations then you run the risk that the regulations will change before you get them coded or after you go live. There's always the risk of an "out of cycle" change to the law resulting from a court ruling. These might overturn regulations, interpret them more broadly than expected, or have some other impact. Not only does this change happen, it's hard to control and even harder to predict. Plus, of course, you don't have the option of ignoring it!

Business Policies

Even if your industry isn't regulated, your business has policies and procedures. These too can change, although you may have some ability to control them. If you work well with your business owners, you may be able to get them to delay changing their policies or changing them only to make them easier to implement. Of course, you may not. Even if your immediate contacts are cooperative, corporate may have something to say about it and force your collective hand. Either way, changing business policies can be a bear.

Competitive Environment

This one is a shoo-in for any company with competitors. If you're building a system that in any way manages customer interactions – decides on pricing, helps them configure orders, makes them offers, or does other self-service tasks – then changes in the ways your competitors do business can impact you. If your company finds it's being outmaneuvered by a competitor, it will have to respond and that could mean changing your system.

Customer Expectations

Even if your direct competitors don't change the way they do business, your customers can be impacted by changes to other products or services they buy. If your customer base overlaps with the customer base of another company that makes some great innovation in customer service, for instance, you may come under intense pressure to do the same thing as your customer satisfaction ratings fall. No matter how well you watch your competitors (and you do watch your competitors, don't you...) this can catch you out and force changes.

Business Rules – What Do They Do?

So if change is inevitable and nasty, what can you do about it? Well, business rules management solutions, sometimes called business rules engines, can play a key role in letting you tolerate, if not love, change. So how do they do this?

Separate Business Rules from the Plumbing

The first key thing a business rules management solution (BRMS) does is separate the business rules from the plumbing. A good BRMS will store the business rules in a repository and allow you to package them up and deploy them as components in your systems. Really good ones will track changes to the repository and help you automatically deploy new rules to existing deployed components in some controlled, sensible fashion.

This separation can be done as part of a Service Oriented Architecture (SOA) by developing and deploying rules services and calling them using standard service-oriented constructs like Web Services. It can also be done without going that far by packaging

the rules as EJBs, servlets, .NET assemblies, or even COBOL code and integrating the result normally. Even if the final packaging isn't service-oriented, the business rules used to drive the rules components have still been separated from all the rest of the system.

Centralize and Manage Business Rules as a Corporate Asset

A BRMS puts the rules into a repository. Managed correctly, this lets you develop and manage the rules as a corporate asset in much the same way a well-designed data architecture lets you treat customer data as an asset to be shared. Such a central repository of business rules lets you ensure consistency across multiple systems and processes by embedding components that execute the same rules, as well as eliminate the problem of too many definitions of "bad payers" or "gold customers." If your BRMS lets you deploy your rules to all your platforms you start to get a return on being able to reuse rules across both legacy and new systems. You might build cross-sell rules as part of a web redesign, for instance, but then deploy the same rules to your mainframe to print targeted cross-sell offers on customers' monthly statements. One definition used everywhere.

This kind of reuse has a side benefit. As you use a set of rules to automate a decision in more places, you can justify further investment in making that decision smarter. So, if the same cross-sell rules get used everywhere it makes sense to invest in making those rules as smart as possible. By focusing investment and leveraging it you can justify more advanced analytics and reporting to improve the decision.

Provide Tools To Let Business Users Own Their Business Rules

The last key component is the ability to put business users in charge of their own rules. A good BRMS will let you do this with enough control, and enough restrictions, that you don't need to hyperventilate each time your users tell you they've made an update: templates, allowing your business users to fill in the blanks; controlled-value fields driven from database tables and existing systems; clear use of business terminology to avoid confusion. In fact this area is key to the whole value proposition so let's discuss it in more detail.

Putting Business Users in Charge of Their Change

So how do you give business users control of their business rules and business process? What does it really take to allow business users to do their own maintenance? Well, one of the first things to remember is that business users do not, in fact, want to maintain their systems or their business rules – they think that's your job! What they want to do is promote slow-moving products through the Web channel, personalize messages on customer statements, allow call center reps to approve requests for insurance policies, enforce new regulations or whatever – they want to run their business, not your system. If you can make maintaining your system look to them like they're running their business, they'll do it for you. If you can't, they won't. So what does this take?

A Zero-Training Environment

Well, perhaps not zero training but pretty close. They're not going to want to have to be trained on some new interface or new approach just to help you out. Instead they need to be given a familiar interface, probably a Web page that looks like the other Web-based or intranet-based applications they use already. They'll have to be able to point and click, select from lists, and do all the other things they are used to with decent systems. So the first rule of business user rule maintenance is don't use a special interface, make it familiar.

Integration with their Environment

Part and parcel of making this familiar is going to be the need to integrate it tightly with their environment. If they have a place where they see reports that might cause them to change, say, their product promotions then they should be able to change the rules right there in the same application – it does no good to separate out rule maintenance from everything else they do. It also helps a lot to reuse things they've already told you. If there's a database table with territories in it then the list of territories to use in a rule should be that list, not one that has to be maintained separately. If they write a rule to define a new customer segment then other rules that use the segment a customer is in need to reflect that. And so on.

Templates

Most business users aren't going to save using even the user-friendly rule syntaxes offered by a modern BRMS. They're going to have to have a template that controls what they can change, how they change it, what options they can use, and so on. Otherwise some fool will do something like use "<" to compare two strings without understanding how case impacts that or use a piece of data that requires expensive processing in a rule without considering the value. Templates, controlling which data can be accessed and how they can be evaluated are key to letting your users build good conditions into rules. Control over what they can change as a result of a rule being true is even more important – that let's you make sure they set the flags you created for them and don't overwrite someone's SSN.

Good templates also let you control how each rule is presented. Does an if/then structure make sense or does the user really just want to see the list of conditions? Should a look-up table or tree be used or is a set of rules okay? Are there uneditable rules that should be displayed alongside the editable ones to help users understand them. And so on.

Customized

The other thing to bear in mind here is that different users will have different environments and will have different rules to edit. The warehouse manager might want to manage restocking rules as part of her warehouse management environment while the marketing manager wants to manage his promotion rules as part of his sales tracking system. Your BRMS needs to let you define multiple sets of rules and expose them in a controlled, appropriate, and integrated way for each user community.

Auditing & Control

Last, but not least, you'll have to provide audit trails of the changes made and changes deployed and controls to make sure updates and changes are managed and restricted. Your users may not appreciate this, but you will....

Loving Change

So will a business rules management solution let you love change? Perhaps not. Will it give you more time to focus on the kind of change that's actually fun? Probably. Will it get some of your users off your back? Definitely.

About the Author

James Taylor is vice president for Fair Isaac enterprise decision management technologies where he is responsible for working with clients to identify and bring to market advanced decision management solutions that will better solve their business needs. James writes a popular blog on the subject at http://edmblog.fairisaac.com.

jamestaylor@fairisaac.com

The Performance Woe of Binary XML

The scapegoat's story

WRITTEN BY **JIMMY ZHANG**

Since its inception, XML has been criticized for the overhead it introduces into the enterprise infrastructure. Business data encoded in XML takes five to 10 times more bandwidth to transmit in the network and proportionally more disk space to store. While most agree that verbosity is inherent to XML's way of encoding information (e.g., extensive use of tags and pointy brackets), the explanation of XML's perceived performance issue remains inconclusive. A popular belief is that since XML is human-readable text, it has to be slow and inefficient. And by the same token, proponents of binary XML seem to suggest that a compact encoding format, most noticeably the binary XML, would automatically lead to better processing performance.

Does it make sense for doctors to prescribe medicine without a diagnosis? Whether those perceptions and beliefs have a grain of truth or not, one thing is certain: Without a solid understanding of XML's performance issue, it will be difficult, if not impossible, to devise meaningful solutions. So in this article, I'll attempt to dissect XML's performance issue by focusing on three key questions: (1) Does XML have a performance issue? (2) What is the real culprit behind XML's slow performance? (3) Can binary XML fundamentally solve the problem.

Performance Is Not an XML Issue Per Se

In networking system design, the OSI (Open System Interconnect) stack is the standard model that divides the functions of the network into seven layers. Each layer only uses the layer below and only exports functionalities to the layer above. Compared with monolithic approaches, the advantages of OSI's layered approach are the robustness, resilience, and extensibility of the networking system in the face of rapid technology evolution. For example, any Voice over IP applications will work without knowing the physical layer of the networks (e.g., using copper, fiber cable, or Wi-Fi), or the data link layer (e.g., Ethernet, Frame Relay, or ATM).

Likewise, we can take a similar layered approach to modeling XML-based applications. Figure 1 is a simplified view of this "XML protocol stack" consisting of three layers: the XML data layer, the XML parsing layer, and the application layer. The application layer only uses functions exported by the XML parsing layer, which translates the data from its physical representation (XML) into its logical representation (the infoset).

Several observations can be made concerning the perceived performance issue of XML.

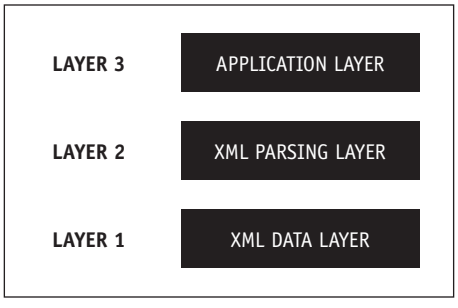


Figure 1: XML protocol stack

First and foremost, because the XML application can only go as fast as the XML parsers can process XML messages, performance is actually not an issue of XML per se, but instead an issue of the XML parsing layer. If an XML routing application (assuming minimum overhead at the application layer) can't keep pace with incoming XML messages at a gigabit per second, it's most likely because the throughput of XML parsing is less than that of the network. Consequently, the correct way to boost the application performance is to optimize the XML parsing layer. Just like tuning any software applications, the best way to do it is to discover then find ways to reduce or eliminate the overhead in XML parsers.

Object Allocation – the Real Culprit

To get a feel of the performance offered by current XML parsing technologies, I benchmarked the parsing throughput of the two types of widely used XML parsers – Xerces DOM and Xerces SAX. The benchmark applications are quite simple. They first read an XML document into memory then invoke parser routines to parse the document for a large number of iterations

and the parsing throughput is calculated by dividing the file size by the average latency of each parsing iteration. For SAX parsing, the content handler is set to NULL. Several XML document in varying sizes, tagginess, and structural complexity were chosen for the benchmark. The results, produced by a two-year old 1.7GHz Pentium M laptop, are summarized in Figure 2. The complete report, which includes test setup, methodology, and code, is available online at http://vtd-xml.sf.net/benchmark.html.

The benchmark results are quite consistent with the well-known performance characteristics of DOM and SAX. First of all, the raw parsing throughput of SAX, at between 20MB/sec~30MB/sec, is actually quite respectable. However, by not exposing the inherent structural information, SAX essentially treats XML as CSV with "pointy brackets," often making it prohibitively difficult to use and unsuitable as a general-purpose XML parser. DOM, on the other hand, lets developers navigate in-memory tree structure. But the benchmark results also show that, except for very small files, DOM is typically three to five times slower than SAX. Because DOM parsers internally usually use SAX to tokenize XML, by comparing the performance differences, it's clear that building the in-memory tree structure is where the bottleneck is. In other words, allocating all the objects and connecting them together dramatically slow everything down. As the next step, I ran JProfiler (from EJ-technology) to identify where DOM and SAX parsing spend all the CPU cycles. The results confirmed my early suspicion that the overhead of object allocation overwhelmingly bottlenecks DOM parsing and, to a lesser (but still significant) degree, SAX parsing as well.

Some readers may question that DOM – only an API specification – doesn't preclude efficient, less object-intensive, implementations. Not so. The DOM spec is in fact based entirely on the assumption that the hierarchical structure consists entirely of objects exposing the Node interface. The most any DOM implementation can do is alter the implementation of the object sitting behind the Node interface, and it's not impossible to rip away the objects altogether. So, if the object creation is the main culprit, the DOM spec itself is the accomplice that makes any performance-oriented optimizations prohibitively difficult. And this is why, after the past eight years and countless efforts by all major IT companies, every implementation of DOM has only seen marginal performance improvement.

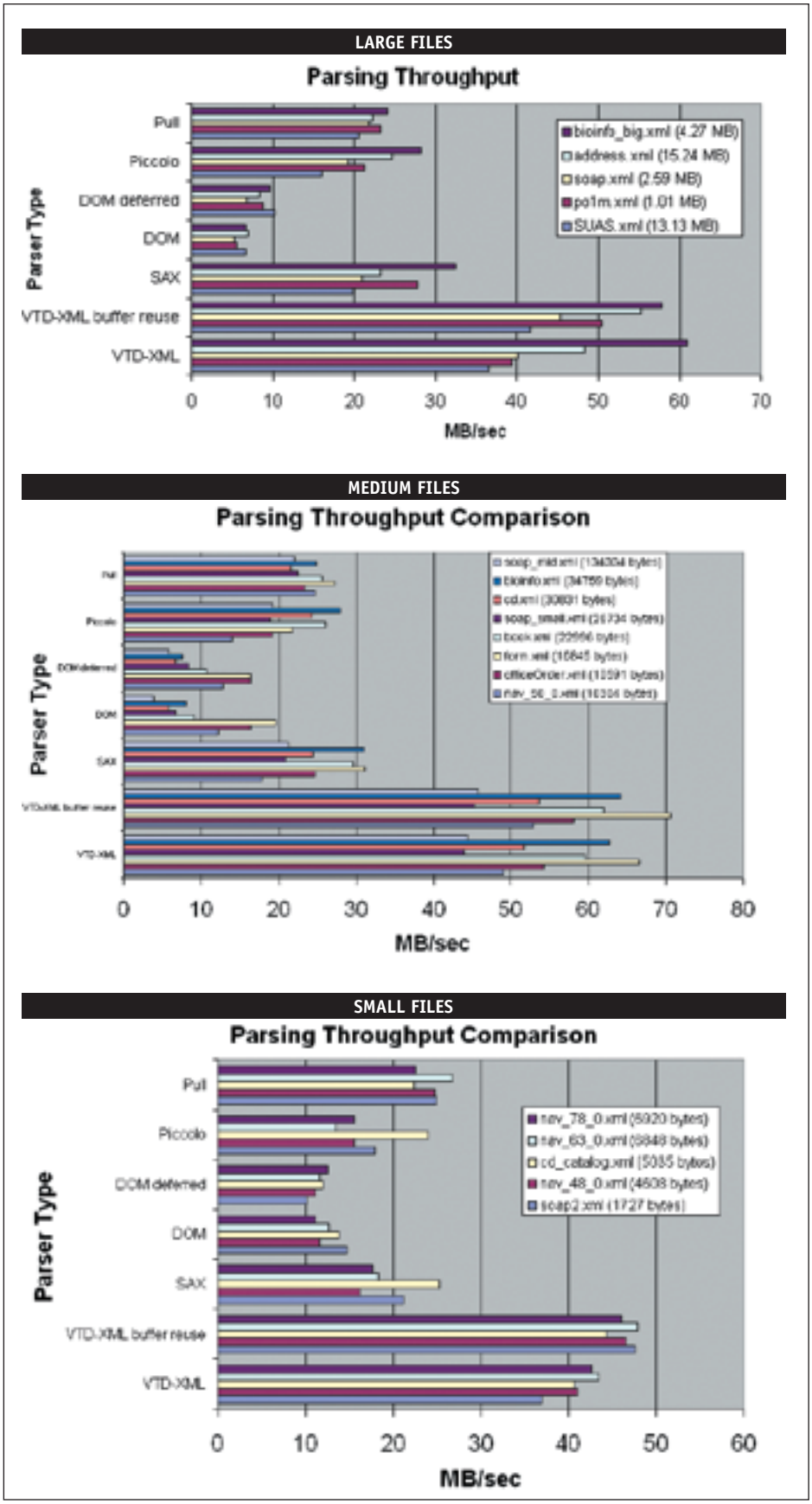


Figure 2: DOM and SAX parsing throughput summary

> **Actually performance isn't an issue of XML per se but of the XML parsing layer**

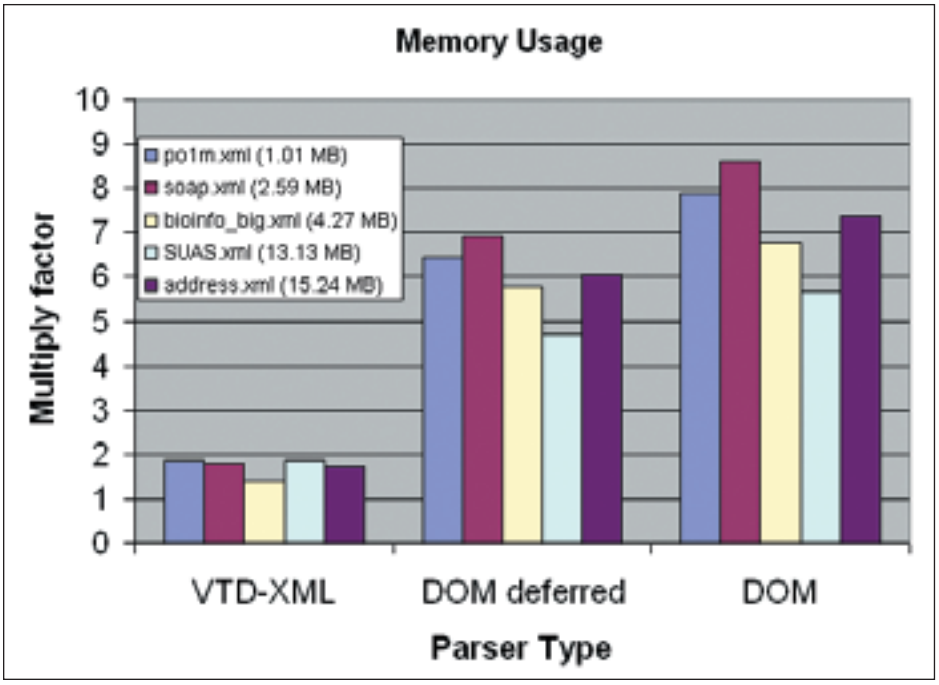
Binary XML Solves the Wrong Problem

Can binary XML fundamentally solve the performance issue of XML? Well, since the performance issue belongs to XML parsers, a better question to ask is whether binary XML can help make parsing more efficient. The answer has two parts – one for SAX, the other for DOM.

Binary XML can improve the raw SAX parsing speed. There are a lot of XML-specific syntax features that binary XML can choose not to inherit. For example, binary XML can replace the ending tags with something more efficient, entirely avoiding attributes so SAX parsing no longer does uniqueness checking, or find other ways to represent the document structure. There are many ways to trim CPU cycles off SAX's tokenization cost. And proponents of binary XML often cites up to a 10x speedup for the binary XML version of SAX over text XML.

But they ignored the simple fact that SAX has serious usability issues. The awkward forward-only nature of SAX parsing not only requires extra implementation effort, but also incurs performance penalties when the document structure becomes only slightly complex. If developers choose not to scan the document multiple times, they'll have to buffer the document or build custom object models. In addition, SAX doesn't work well with XPath, and in general can't drive XSLT processing (binary XML has to be transformed as well, right?). So pointing to SAX's raw performance for binary XML as a proof of binary XML's merit is both unfair and misleading. The bottom line: for a XML processing model to be broadly useful, the API must expose the inherent structure of XML.

Unfortunately, binary XML won't make much difference improving DOM parsing for the simple reason that DOM parsing generally spends most CPU cycles building in-memory tree structure, not on tokenization. So the speedup of DOM due to faster SAX parsing is quite limited. In other words, DOM parsing for binary XML will be slow as well. And going one step further, object-graph based parser will have the same kind of performance issue for virtually any data format such as DCOM, RMI, or CORBA. XML is merely the scapegoat.



Because SAX and Pull do not build data structures in memory, so the meaningful comparison is between DOM and VTD-XML. To that end, we benchmark the multiplying factor which is the ratio between the memory usage and the document size.

Reduce Object Creation – the Correct Approach

From my benchmark and profiling results, it's quite easy to see that the best way to remove the performance bottleneck in XML parsing is to reduce the object-creation cost of building the in-memory hierarchical structure. And the possibilities are, in fact, endless, and only limited by the imagination. Good solutions can and will emerge. And among them is VTD-XML (http://vtd-xml.sf.net). To achieve high performance, VTD-XML approaches XML parsing through the two object-less steps: (1) non-extractive tokenization and (2) hierarchical-directory-based random access. The result: VTD-XML drastically reduces the number of objects while still exporting the hierarchical structure to application developers and significantly outperforming SAX.

Summary

To summarize, the right way is to find better parsing techniques beyond DOM and SAX that significantly reduce the object-creation cost of building XML's tree structure. Binary XML won't fundamentally solve XML's performance issue because the problem belongs to XML parsers, not XML. ■

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 graduated from U.C. Berkeley with both an MS and a BS from the department of EECS.

crackeur@comcast.net

Ah, remember when EDI was young and full of promise?

The future of EDI is B2B over IP

EDI sure had promise in the sixties – but its complexity, inflexibility, and the bottleneck of the VAN make it very costly today.

Simple to integrate, easy to manage, and blazingly fast, **terminalONE** is *the* B2B over IP platform. It intelligently transports, transforms, and routes all your data transactions.

We're about ebusiness adaptability: your business world changes fast – wouldn't it be nice if your data interchange adapted quickly and painlessly along with it?

terminalONE™

Secure, intelligent ebusiness transactions

*high-**velocity***
*high-**volume***
*high-**availability***

Take **terminalONE** for a test drive – ***today***
www.xenos.com/VAN

1 888 242 0695
1 905 763 4468
terminalONE@xenos.com

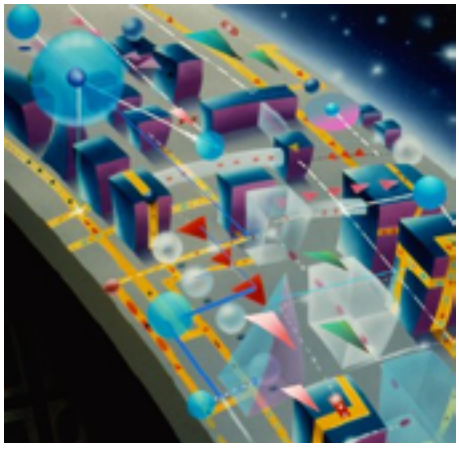
xenos

Turning Service-Oriented Events into Business Insight

Event-Stream Processing – tools for an event-driven service oriented architecture

WRITTEN BY **MARK PALMER**

The quest for agility has spurred the recent rise of Service Oriented Architecture (SOA) and the face of modern IT integration architecture is changing. Technology stovepipes of the past are now being connected by Enterprise Service Bus (ESB) technology, which provides the backbone for the networking, communication, mediation, and service container management needed to support an SOA. Every integration software vendor provides some form of ESB in its products and the ESB has risen to the status of a de facto standard for SOA integrating. But what's the next step in the evolution of the IT integration fabric?



The next step is a new class of software called event-stream processing, or ESP. ESP has been hailed as the "next big thing" by both software vendors and analysts because it helps an SOA integration fabric become intelligent and responsive. ESP empowers a business to think differently about its operations and IT infrastructure because it can understand the state of a business in the now, rather than just in the past.

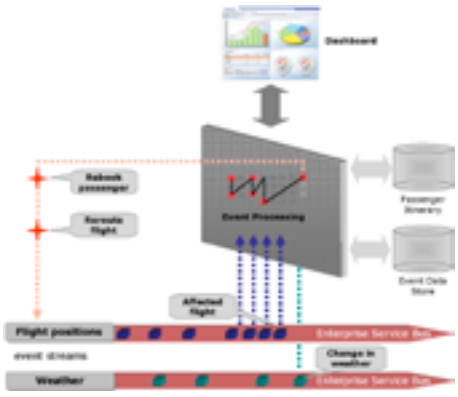
But aren't ESBs already emitting events and aren't all SOA infrastructure elements capable of emitting and carrying events? The answer is "yes," ESBs are event-capable today. However, they don't prescribe what to do with the events that are emitted by their services. That's the critical value that ESP brings.

ESP enables an event-driven SOA to decipher causal (if A is followed by B followed by C), temporal (within four seconds), and spatial (within 10 feet) relationships among events – and can do so in real-time. This kind of "enterprise wiretap" lets a business continuously analyse key performance indicators in real-time, identify threat and opportunity in real-time, and act instantly. These capabilities require a new style of computing – stream computing – that can deliver the missing link between an event-driven SOA and real-time business insight.

This article describes the characteristics of stream computing, how an ESP works, and how ESP and the ESB combine to provide an agile, intelligent integration fabric, and an intelligent, real-time enterprise.

Event-Driven SOA Delivers Real-Time Insight

The goal of an event-driven SOA is not merely the capture of all – or some – of the events generated by the architecture. Its goal is to derive business insight from those events – and derive that insight quickly enough to act. An example of a system that delivers real-time intelligence is the "Delta Nervous System."



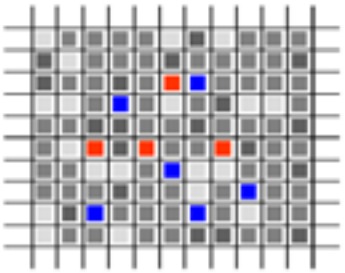
Delta Airlines, under pressure from competitors like Southwest Airlines, described a concept in 2003 designed to alter its operational infrastructure. The Delta Nervous System monitors operational events, analyses them in real-time, and makes operational decisions in real-time.

The Delta Nervous System monitors weather forecasts and flight positions, identifies patterns among those events – such as changes in the weather – that determine flights that need rerouting. If flights have to be rerouted, that action triggers further events that cause passengers to be rebooked in real-time. Dashboards alert Delta flight operations personnel of the status of flights, with messages sent to kiosks in terminals and gate agents that keep the whole operational organization synchronized.

This real-time operational infrastructure is an example of an event-driven SOA: with flight positions transported by ESB; passenger itineraries checked by database query; dashboards updated with the state of flights, crews, and passengers; and decisions enacted by emitting events. This is a system that needs event processing to provide real-time insight.

The Missing Piece of an Event-Driven SOA: Event-Stream Processing

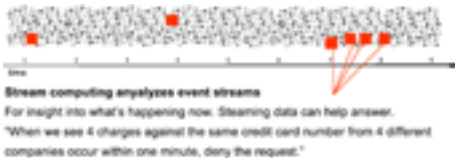
Real-time intelligence lets a business think differently about its operations and IT infrastructure because it can understand the state of the business in the now rather than just the past. Stream computing is a new style of computing that enables instant event pattern recognition, and it's vastly different from traditional styles of computing.



Static computing analyzes static data to analyze a snapshot in time and answer questions like: "How many shoes did we sell in our New York stores last week?"

Traditional computing is static computing, and static computing uses static data. Static data is like a photograph, which captures information about a moment in time. For example, static business data is a table of customer data, transactions at a retail store, or records of shipments that have occurred in a company's supply chain. Static computing can be used to answer questions like: "How many shoes did we sell in our New York stores last week?" Over $200 billion of software is sold worldwide each year, and almost all of it is designed for static data. Relational databases, for example, are designed to manage static data.

Real-time computing is stream computing, and stream computing uses events. Events arrive in streams that resemble a movie in which streams of images and sounds flow by your senses. Patterns in the stream of images and sounds can make you laugh, cry, or scream. Much like a movie, streaming events in an enterprise let the business feel the pulse of operations as events travel through its arteries – the ESB. With stream computing, a business can identify patterns and make instant decisions while they still matter: "When four transactions against the same credit card number from four different shops happen in one minute, deny the next request, flag the account, and send a message to the fraud detection dashboard." And like a movie's impact on its audience, these patterns might well make a CFO laugh or cry.



Stream computing analyzes event streams

For insight into what's happening now. Steaming data can help answer. "When we see 4 charges against the same credit card number from 4 different companies occur within one minute, deny the request."

Static computing and stream computing have fundamentally different physical characteristics – stream computing enables instant, real-time, intelligent decisions based on the patterns of business operations. These patterns are determined by the temporal, causal, or spatial relationships among events in the stream.

The concept of stream computing isn't new, but the efficacy of the tools that implement it is. In 1985, database researchers began to explore the element of time by extending the relational database with time-series extensions. But this approach was fatally flawed, since databases are designed to answer questions about the past, and event processing must make decisions in the now. So, beginning in 1998, a new approach – ESP – attacked the event-processing problem afresh, introduced the first wave of commercial ESP products, and ushered in a new physics of computing.

A BAM Dashboard Isn't Enough

An event-driven SOA is a technical architecture that lets software systems interoperate dynamically. But an effective SOA doesn't satisfy the requirements for instant business insight alone since it lacks the ability to identify temporal and causal relationships among events in real-time. Some software vendors promote Business Activity Monitoring, or BAM, as a new means of business insight:

BAM is "a business solution that is supported by an advanced technical infrastructure that enables rapid insight into new business strategies, the reduction of operating cost by real-time identification of issues, and improved process performance."

Unfortunately vendors have used BAM as a quasi-technical banner with which to sell flashy dashboards. But serious real-time intelligence demands more than a flashy dashboard. ESP provides a computing foundation from which it can answer real-time questions about streaming data, such as:

- "How many plasma screens do we have in stock, how many have been ordered, what is our rate of consumption, and how are we doing against our forecasted rates – right now?"
- "Is bad weather forecast within 10 miles of this flight's path? If so, re-route the plane over New York, determine which passengers will be impacted by the delay, rebook the passengers, update kiosks and gate agents at all locations – and update our real-time P&L."
- "If transactions have occurred in the last five seconds against this credit card from different businesses in different physical locations deny the most recent request and alert fraud detection of the account in question."

Visualization technology isn't enough to answer these questions. An enterprise can ask and answer these questions with the temporal, causal, and spatial capabilities of ESP, and can visualize the results with the help of a dashboard. As a result, dashboards to visualize streams are an insufficient element to deliver event stream processing.

The Elements of Event Stream Processing

To deliver the benefits of ESP requires a range of architectural elements that include event-processing servers, event data man agement, event visualization, system management, ESB integration, and ESP development tools. Conceptually, these

basic elements are similar to their static computing counterparts. A static system might, for example, use Java as a programming language, Visual Basic for the graphical user interface, and SQL to access data, and a relational database for storage. In the world of ESP, these are more likely to be a Java-based event-programming language (EPL), a dashboard for visualisation, and an event data store for event storage and retrieval.

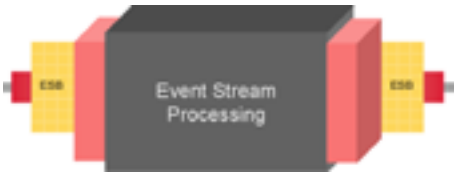
In total, there are seven elements that deliver a comprehensive ESP software platform:

1. ***Event Processing Engine*** – Just as an application server is the brain of a multi-tier architecture so an event-processing engine is the brain of an event-driven SOA. The event-processing engine monitors multiple event streams, identifying relationships among events by applying ESP rules that are expressed with an EPL. It may utilize non-event data (sourced from an RDBMS) or previously received events (from an event data store) as part of its analysis. Its output represents "derived" or "complex" events that can be sent to other applications, often via an ESB.
2. ***Source and Sink Stream Adapters*** – Event-driven SOA emits events via messages on an ESB; tapping into those event streams via source (inbound) and sink (outbound) stream adapters. For deployments without an ESB, other (often application-specific) transports can be used to receive and/or distribute the events. For example, in event-processing applications involving RFID (radio frequency identification), Application Level Events (ALE) is a standard interface for capturing radio signals from RFID tags. In financial event stream applications, trading systems integrate with various exchanges via market data feeds (e.g., Reuters) or order management protocols such as FIX.
3. ***Event Development Tools*** – An EPL lets developers express complex event processing (CEP) rules that detect temporal, causal, and spatial relationships among events. Recent innovations in ESP have resulted in EPL tools that let business users create, deploy, and modify rules themselves – further aiding "agility" by enabling business users to participate in the expression of business logic. An in-depth EPL example and its ability to discover complex relationships among events is discussed below.
4. ***Event Visualization and Multi-Channel Output*** – ESP processing immediately notifies users through alerts on a computer screen dashboard. But dashboards are just one vehicle for event delivery. Multi-channel output enables an ESP processing system to send e-mail, control automated devices via software interfaces, issue messages via pagers and other wireless devices – or send data to kiosks for consumers to see.
5. ***Event Data Stores*** – The characteristics of event data – both in volume and type – suggest the need of a new class of data management infrastructure to meet the demands of stream-processing applications. Such event stores should support configurations that 1) capture every event (meaningful or not) that is carried by the ESB and 2) capture filtered subsets of events as determined by ESP logic to be meaningful or 3) new "derived" events ("complex" events) that are created from other events. Rather than store events in RDBMS tables that focus on static attributes such event data stores will typically organize and retrieve data using time as the key value.
6. ***Relational Database Caching*** – As the processing engine operates on event streams, it often needs access to structured data to inform decision-making. For example, to rebook passengers correctly, the Delta Nervous System may have to access passenger itineraries to determine if rerouting is practical. Relational caching technology provides in-memory access to cached relational data sets. By caching this data, an ESP platform can avoid repetitive SQL requests to a relational server when checking the impact of flight rerouting on a plane's entire passenger manifest.
7. ***Management, Fault Tolerance, State Recovery*** – Enterprise-class "abilities" – reliability, manageability, and scalability – are critical elements in any software platform and ESP is no different. Although typical management and monitoring facilities are critical for an effective ESP architecture, it poses some unique management challenges. For example, since each event-processing engine holds its own view event state, distributed state management and recovery become more challenging.

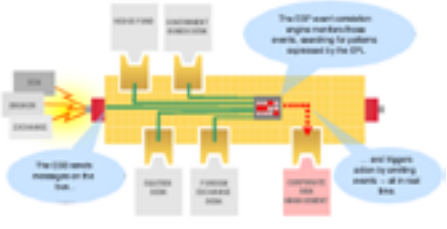
Each of these elements of ESP can be explored in depth elsewhere; the balance of this article focuses on the intersection of ESP and ESB together with the foundational programming and architectural issues of ESP.

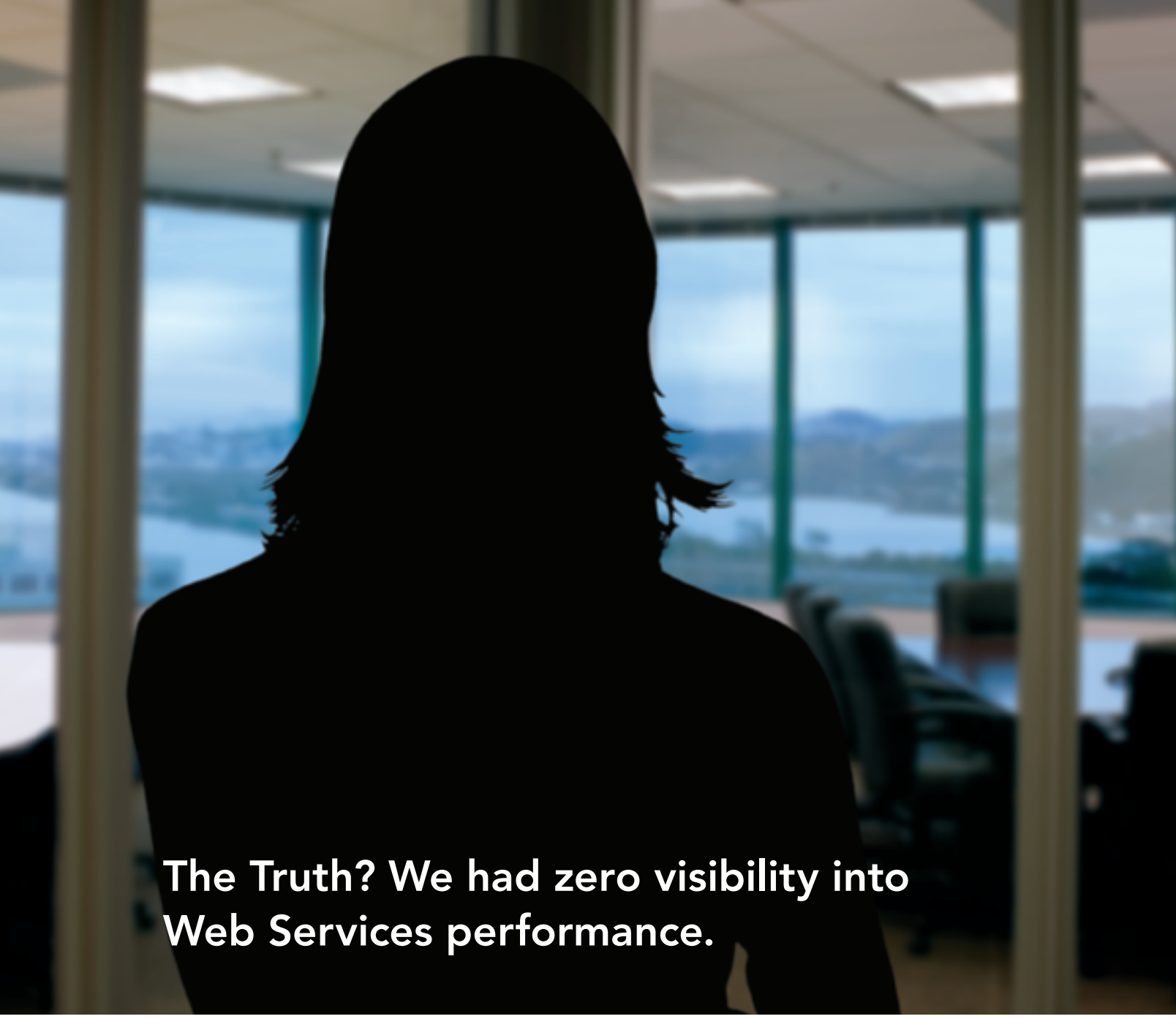
The ESB Is the Stream

ESB service endpoints can both generate and respond to asynchronous events. Though the ESB provides the integration backbone, it doesn't inherently deliver the analytics layer that can oversee and understand all the resulting events that are spawned by such an architecture. ESP adds insight to ESB-generated events.



In the example below, the trading desks of an investment bank are integrated with an ESB. Equities (stocks), government bonds, and foreign exchange are traded by independent, loosely coupled systems, whose service-oriented interfaces are exposed on an ESB. A key benefit of an ESB is that it normalizes the enterprise event stream into XML; JMS, application events, and FTP events are transformed into a normalized form by the ESB fabric and these events are delivered in streams that can be analysed for patterns in real-time by ESP. In this example, each trading desk executes trades and emits events via the ESB that represent trades. The ESP processing engine monitors these events, as well as other external events, and provide a real-time check that the firm's overall risk (a compilation of equities, bonds, and foreign exchange) hasn't been exceeded. Allowable trading levels gyrate according to real-time activity; when one trading desk takes an aggressive position, another might have a conservative one; combined, the risk is acceptable. However, if two desks take an aggressive position at the same time, they must be instructed to modify that position, perhaps by sending an event that automatically limits the amount of a security it can trade. If the desks don't alter their risk positions quickly enough, ESP can send an event that reports the situation to a risk management authority. ESP can monitor and moderate the real-time ebb and flow of risk in a large firm.





Wily can handle the truth. We've helped hundreds of organizations around the world manage critical Web Services and proactively identify problems before customers are affected. Using a single solution, you gain control over both your application infrastructure and the customer experience. To learn more about how Wily can help you achieve customer success with Web Services, visit **truth.wilytech.com**.

Get Wily.™

Enterprise Application Management

wily technology™

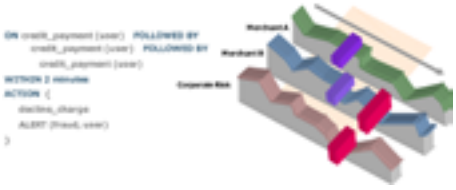
The use of ESP and an ESB together has been described as an advanced stage of maturity of an SOA in Sonic Software's "SOA Maturity Model." How do ESP and an ESB combine to make an effective event-driven architecture work?

ESP – Finding ESB Event Patterns Using Causality and Temporal Constraints

To illustrate how the ESB and ESP can work together, let's examine a specific example: credit card fraud detection. The goal is to monitor purchase activity in the system and capture authorization requests that can be analysed in real-time to detect fraudulent activity. Such operations fully express the three-stage event-processing paradigm of 1) monitor, 2) analyse, and 3) act.

First, we need electronic access to the events on the ESB. Source event streams, show below as "Merchant A" and "Merchant B," represent purchase activity events delivered via messages on the ESB to the event engine.

Second, we need rules that act on those events. Since ESP engines process events asynchronously, events can come from anywhere, be of any type, and be received in any order. An EPL uses events, their time of occurrence, any causal relationship among events, and event abstraction as its fundamental elements, rather than the structured data and relational algebra of SQL. An EPL processes queries "partially" – for example, when A and B are true then if C occurs in N seconds, take some action. An example of one rule is shown below.



This EPL code begins with an event filter ("ON credit_payment (user)"). This statement instructs the engine to monitor events on the ESB that represent charges to a credit card; as events flow through the ESB, this event pattern is partially satisfied. Next, the "FOLLOWED-BY" statement instructs the ESP engine to monitor subsequent credit_payment events against the same account ("user"). If three charges occur in two minutes then the code has identified a potentially fraudulent activity. Though not depicted above, an example of including "spatial" logic might be to flag any charges that fall outside a pattern of purchases geographically or flag a subsequent charge if the geographic distance between the purchase locations suggests that the legitimacy of one of the charges is improbable, if not impossible. These examples illustrate the first central concept of ESP: the inference of causality. ESP has inferred from the relationship among the charges that one or more of the charges were caused by caused by fraud.

The "WITHIN" statement illustrates another key concept in ESP: namely time. In this example, if the third credit_payment event isn't detected within two minutes of the first charge, the activity isn't flagged as potentially fraudulent, and the scenario completes. In stream computing, as in the "real-time, agile enterprise," the significance of any individual event – in terms of business importance – quickly depreciates. The window of opportunity to act on an event is often brief and transient. Unless the event processing architecture can rapidly discern the significance and respond, the opportunity to exploit the situation will quickly pass, with the circumstances altered by subsequent events or other factors.

Finally, we see the third key element of ESP: action. Automated systems like a credit fraud detection application often invoke event-driven actions once a pattern has been detected. In this example, the current request for a charge is declined and the account is flagged for fraud management by sending a new derived event on the ESB.

Conclusion

As the Enterprise Service Bus becomes the backbone of the enterprise IT integration infrastructure, it provides a stream of events that can make real-time insight a reality. Stream computing and the ESP tools that enable it provide the ability to detect time, cause, and spatial-based patterns among events in the ESB stream. By combining ESP rules with an ESB-normalized integration fabric, the enterprise can become truly agile. ■

References

- ***Event Programming:***
  Mark Palmer and Gareth Smith.
  "An Event Processing Language Tutorial."
  Progress Software.
- ***General ESP Product and Academic Information:***
  A is a web portal of ESP term definitions, links to ESP products, academic papers, and community discussions.
  www.eventstreamprocessing.com
- ***ESP Architecture and Technology:***
  John Bates.
  "Apama Technical Overview." Progress Software. *Available on request.*
  Mark Palmer.
  "Event Stream Processing, a New Physics of Computing." DMReview. *July, 2005.*
  www.dmreview.com
- ***Glossary of Terms:***
  A glossary of industry terms, including BAM, CEP, EDA, Event, and Event Streams can be found at
  www.eventstreamprocessing.com/glossary
- ***Community and Discussion Group:***
  The CEP-interest group, co-moderated by Mark Palmer and David Luckham, has been joined by enterprise architects, members of academia, software vendors, and industry analysts interested in ESP and CEP.
  http://groups.yahoo.com/group/CEP-Interest.
- ***Radio Frequency Identification (RFID) and ESP:***
  Mark Palmer.
  "The 7 Principles of RFID Data Management."
  www.esj.com/enterprise/article.aspx?EditorialsID=1076
- ***Algorithmic Trading and ESP:***
  Dr. John Bates. "The Apama Technical Overview." 2005.
- ***Complex Event Processing:***
  David Luckham. "The Power of Events, An Introduction to Complex Event Processing in Distributed Enterprise Systems." Indianapolis. 2002. A comprehensive list of academic papers and resources can be found at www.eventstreamprocessing.com/webindex.htm.
- ***Academic Resources on complex event processing, data stream processing:***
  The event-stream processing resource site is at www.eventstreamprocessing.com
  See the index.

About the Author

Mark Palmer is vice president of event processing with the Progress Software Real Time Division. Mark has over 15 years of experience in technology leadership as a developer, consultant, and CTO. In 2005, InfoWorld named him one of the year's leading innovators and to the InfoWorld Media Group's Innovators Hall of Fame. He is widely published on technology topics that include event-stream processing, radio frequency identification, algorithmic trading, and large-scale distributed systems.

"Everyone wants faster, better, cheaper. We thought they might appreciate smarter, too." Entering the integration market we had three advantages: we knew the costs and hassles of traditional integration solutions were limiting their adoption, we saw that a standards-based service oriented architecture would solve these problems, and we had the world's most scalable enterprise messaging server, SonicMQ®, as a core technology. We combined SonicMQ's performance and security with Web services, XML transformation, intelligent routing and a new distributed deployment and management infrastructure to develop the world's first Enterprise Services Bus, Sonic ESB™. With it businesses can easily integrate existing and future applications to create unprecedented business agility, and they can start today knowing they can scale to meet tomorrow's needs. We call it incremental integration. It's smarter. It's also faster, better and cheaper.

sonic
SOFTWARE®
CONNECT EVERYTHING. ACHIEVE ANYTHING.™

Gordon Van Huizen, Sonic Software

www.sonicsoftware.com

Creating Secure Web Service Sessions

SSL can complement your Web services security solutions to achieve optimal performance – believe it or not!

WRITTEN BY **KEVIN T. SMITH**

Over the past five years, the promise of enterprise information sharing has made great strides with the evolution of Web Services and the promise of Service Oriented Architectures (SOA). An architectural shift that moves us away from point-to-point client/server systems,

SOA provides new challenges related to propagating trust between services at multiple points in an n-tiered architecture. Many government and industry applications have mission-critical security requirements that make achieving strong enterprise security goals mandatory, and over the past few years, there have been great strides in creating mechanisms for achieving these goals in a SOA. The security solutions that exist, however, directly impact performance. The cryptography used to achieve enterprise security goals will also slow down your Web Service consumers and producers, so any good SOA architect will have to be smart and look at all the options. Focusing on performance, this article presents an approach of using SSL to complement some of the existing Web Services security mechanisms – in situations where it makes sense.

Standards such as WS-Security, the Security Assertion Markup Language (SAML), XML Signature, and XML Encryption are the "glue" for achieving classical security goals in Web Services. For example, using WS-Security SOAP messaging using XML Signature for signing the contents of the entire message would provide non-repudiation (and message integrity) for all messages. If the lifecycle of a SOAP message involves Web Service chaining, then certain cryptographic operations may have to take place between each Web Service consumer and every Web Service provider in the lifecycle to do integrity checks and signature validations. When client applications repeatedly send several hundred Web Service messages along this same chain, SOA architects should re-think this model. After analyzing your security requirements, it may make sense to re-factor your Web Services security solution in these situations. If there are Web Services consumers that constantly have interactions with the same Web Service providers, there may better-performing ways of achieving your security goals than having signatures and validations happening between them every time.

Figure 1 shows a simple example of a Web Service chaining scenario. A portal sends search requests to an enterprise search Web Service that delegates search requests to search providers that do data source searches and return the results. Looking at this scenario, we can effectively demonstrate a dilemma that we face when all messages are digitally signed with WS-Security SOAP messaging, and where much of the same chain is used repeatedly.

In our example, a Web portal sends searches to one enterprise search service, which propagates searches to multiple search provider Web Services in the enterprise. At every point of the message lifecycle, there will be four cryptographic operations – a digital signature when the request message is created, a validation on the receipt of the request message, a signature when the returned message is created, and a validation on the returned message.

Each one of these cryptographic op-

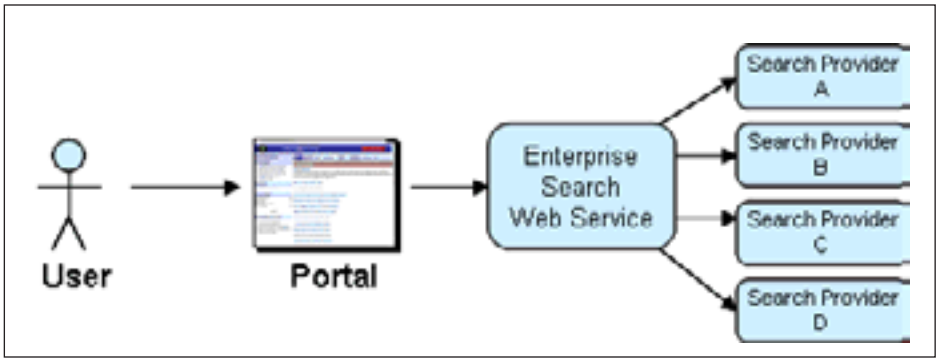


Figure 1: A simple Web Service chaining scenario

erations uses asymmetric (public key) cryptography, which is computationally expensive. All in all, in a scenario with just four data sources in this scenario, 20 cryptographic operations have to take place for one search, which will have a direct negative impact on performance. For 1,000 searches, 20,000 cryptographic operations have to take place. In such scenarios where there's a promise of long-standing relationships between Web Service consumers and Web Service providers in your Web Service lifecycle, the concept of a "Web Service session" should be factored in.

SSL (in this article I'll use the term SSL because the term has been commonly used to mean its replacement, TLS) has been in use for over a decade, and because it's typically used for client/server Internet solutions, it's gotten a bad rap when it comes to Web Services. While SSL doesn't lend itself to providing an overall solution for the distributed nature of Web Services, it can be used as a piece of the puzzle, complementing Web Services security standards to achieve security objectives between two points in a Web Service solution. Long-lived mutually authenticated SSL connections between two endpoints in the Web Service lifecycle can be used to create "Web Service user sessions" in a multi-tiered model where the same Web Service consumers and Web Services constantly communicate. If it's known that a relationship between two nodes in the solution is long-lived, it's a fair estimate that having a long-lived secure connection between those two nodes would eliminate the overhead of repetitive cryptography, providing better performance in repeated messaging.

Figure 2 provides an example of setting up such an SSL session between two points. If the portal is used mainly for enterprise searches via the enterprise search Web Service, a mutually authenticated SSL session can and should be initiated between the Web portal and the enterprise search service. It's well documented that establishing a SSL connection is where the performance penalty lies, since asymmetric (public key) operations are used and are computationally expensive. Once a long-lived session is set up between the two parties, however, symmetric (or "secret key") cryptography is used to renew the session and is much faster. A mutually-authenticated session between the two points in the solution would provide non-repudiation of the identities of the Web Service consumer and provider, and it would provide confidentiality and message integrity between the two points. Plain SOAP requests and responses can be used between the two points. Finally, it should be noted that after the SSL session is set up between the two points in the solution there will be four fewer public key operations for every search. Think of the impact of having 40,000 less public key operations for every 10,000 searches. The longer the SSL session is used between the portal and the enterprise search service in this example, the better the performance average will be over time.

Figure 3 shows a good performance comparison chart, showing simple SOAP requests between a Web Service consumer and a Web Service over time. Obviously performance will vary depending on the WS-Security toolkits you're using and depending on the ciphers used in SSL, but the purpose of this chart is to get an important point across. Initially the SSL setup makes SOAP-over-HTTPS much slower than signed WS-Security SOAP messaging. After a certain amount of time, the performance of SOAP messaging in a long-lived SSL connection is significantly better than the repeated cryptography used in signed WS-Security messaging. The response time of the request/response of signed SOAP messaging is constant; the SSL connection has a performance hit at the beginning, but the subsequent SOAP messages are faster. Over time SSL buys you the performance benefit of being much faster, achieving many of the same goals - message integrity and non-repudiation of the identity of the components involved – but with the added benefit of confidentiality, since encryption is used. (Imagine how much better performance would be over time if we compared signed messages

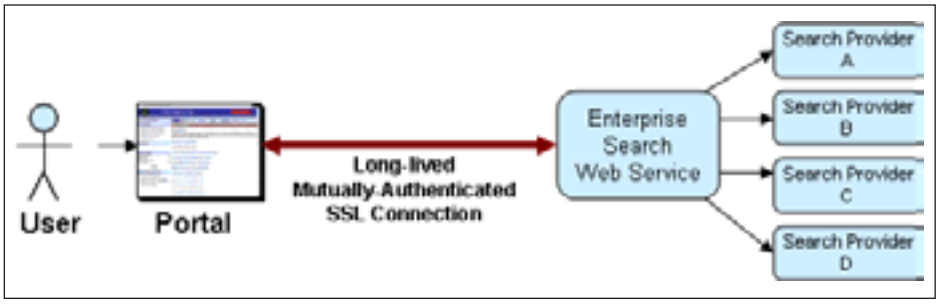


Figure 2: Establishing secure Web Service sessions

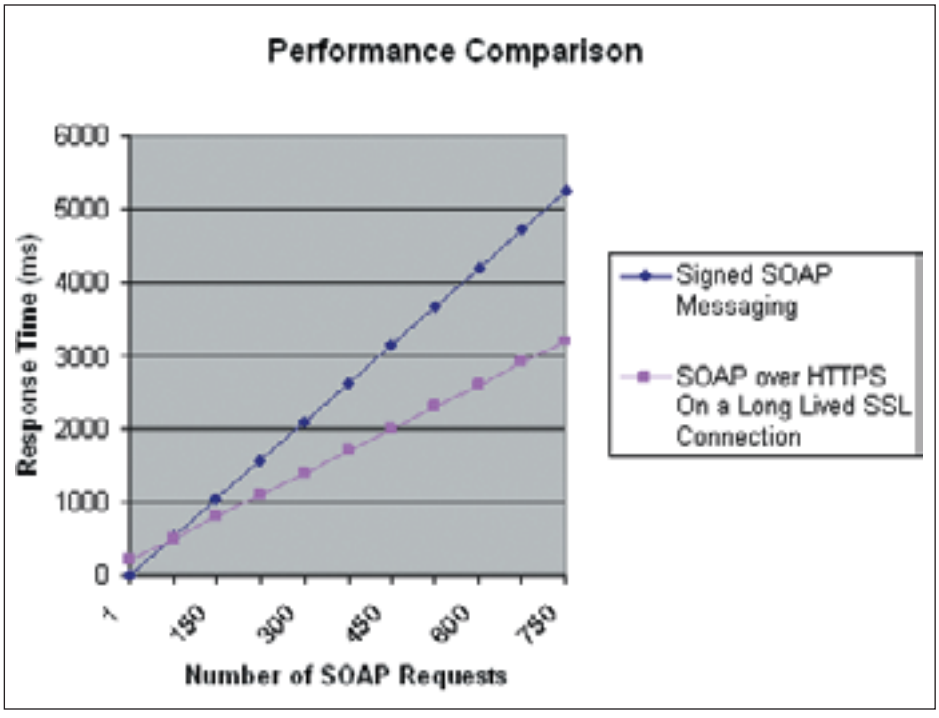


Figure 3: Performance comparison of requests between two points over time

that contained XML Encrypted elements between two points!)

This solution is not the answer for every use case. Because of the dynamic nature of SOA, you may not always be able to predict the paths of Web Service chaining and orchestration solutions. However, when you have more information about the routing and chaining of Web Service messages, using SOAP over long-lived mutually authenticated SSL sessions between constantly communicating points may satisfy your security requirements and improve performance.

Propagating Identity

Propagating identity adds an additional layer of complexity in a Web Service chaining solution. Using standards such as WS-Security SAML Token Profile, a messaging solution can be developed that propagates an end user's identity and authorization credentials in SAML assertions through every point of a SOAP request.

Commonly there are typically two trust models used in SOAs for propagating identity:

- A trusted infrastructure model, where a single SAML-issuing authority exists in the enterprise. In this model, the issuing authority validates the credentials of the user that may generate SAML assertions based on the user's identity and authorization credentials.
- A trusted client model, working in a "sender vouches" scenario. In this model, the authenticating client application is trusted to assert the identity of a user that it authenticates and the client is trusted to issue SAML assertions.

Figure 4 shows our scenario, expanded to include the propagation of identity credentials. Each trust model involves the creation of a SAML Assertion that is propagated in WS-Security SOAP Messaging. A SAML assertion with explicit conditions of use (such as time constraints) can be digitally signed by the authenticating entity or issuing party, and this signed SAML assertion can be propagated throughout the life cycle of the SOAP message using WS-Security SOAP Messaging.

A typical example (ignoring some subtleties for simplicity's sake) is shown in Figure 5. For every SOAP request in this model, XML Signature is used by Web Service consumers and providers to digitally sign the entire SOAP message (including the already-signed SAML Assertion), cryptographically tying all elements of the SOAP message together and providing non-repudiation of the message contents by the sender of the message. In such a messaging syntax, the combination of the signature of the SAML assertion by the asserting party and the signature of the entire message by the sending party provide assurance of not only who the sender is, but the identity of the user involved – at every point in the Web Service chain lifecycle.

Identical to our initial example, there will be performance issues when propagating these identity credentials with WS-Security SOAP messaging. SSL can also be used here for performance benefits in situations where there will be long-lived relationships, but any solution would be more complex because the authenticating client (a portal in this example) may work on behalf of thousands of users. While a mutually authenticated SSL connection is a contract between two entities (the portal and the Web Service in our original example), we also have to have the explicit trust of the user who is using the portal. To do that, you can create the concept of a Web Service "user session" standing on the shoulders of SSL:

1) Create a long-lived mutually authenticated SSL connection between the Web Service consumer and Web Service
2) Send the signed SAML assertion from the Web Service consumer to the Web Service in the WS-Security header of a SOAP message (not signing the entire message)
3) At this point, the response of the SOAP message may include in its header a Globally Unique Identifier (GUID) that can be used in subsequent SOAP messages over HTTPS to refer to the original SAML assertion that was validated.
4) On subsequent messages over the SSL connection, send basic SOAP messages, with the GUID in the SOAP header, specifically referencing the "user session." If the SAML assertion's conditions (such as time constraints) ever expire, it's up to the Web Service implementation to throw a SOAP Fault, explaining that a

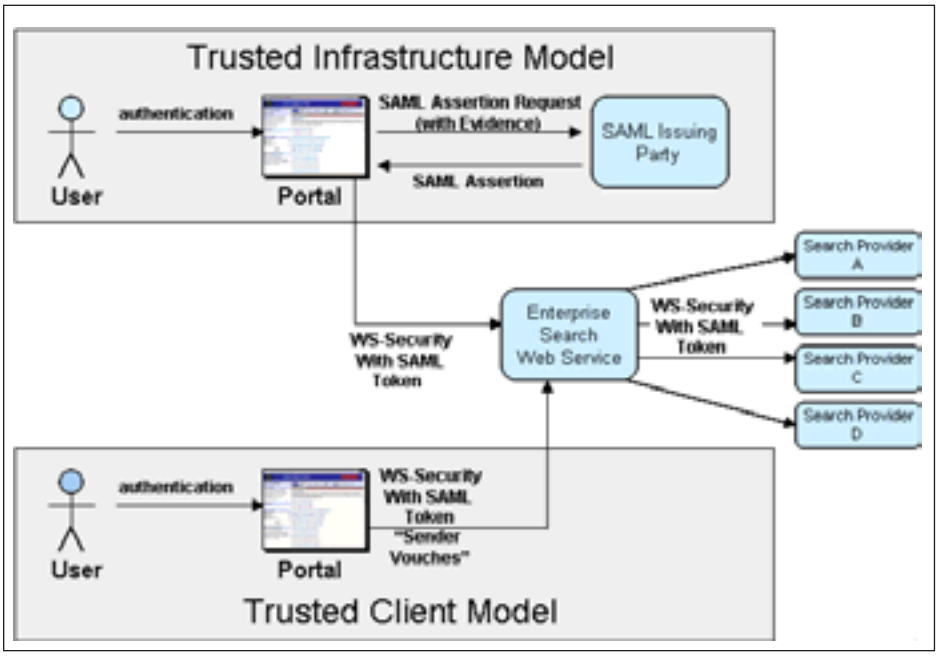


Figure 4: Trusted infrastructure & client models using SAML tokens

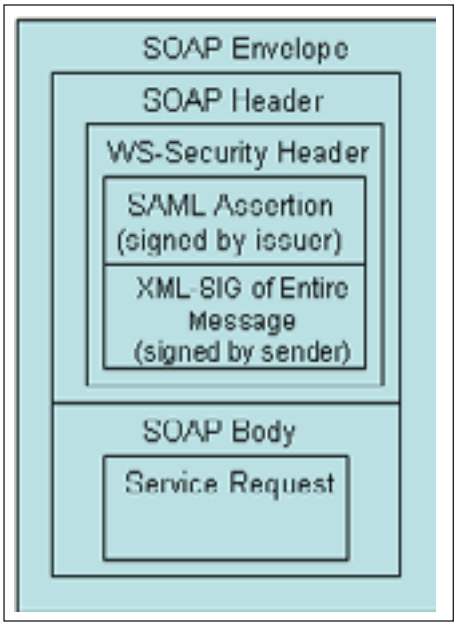


Figure 5: Typical format of WS-Security Messaging with SAML

REALWORLD FLEX SEMINAR

www.flexseminar.com

Learn How to Build the Next Generation of Web Apps **from the Experts!**

August 14, 2006
The Roosevelt Hotel
New York City

Adobe Flex 2 is a complete, powerful application development solution for creating and delivering cross-platform rich Internet applications (RIAs), within the enterprise and across the web. Not until now has there been a way for enterprise programmers and architects to work with existing tools of choice, familiar programming models and integration with existing systems and infrastructure. Multi-step processes, client-side processing, direct manipulation and data visualization are all key factors in the Flex solution.

The "Real-World Flex" One-Day Seminar will delve deep into the central workings of Flex so that the seminar delegates can integrate this timely new technology into their applications, creating powerful interactive content. Delegates will learn from the experts who not only created the product but also those who are using it to the massive benefit of their employers, their customers and the Web.

"Go Beyond AJAX with Flex."

The list of topics at the Real-World Flex Seminar includes:

- ***The Flex Approach to RIA Development***
- ***Bridging Flex and AJAX***
- ***Integrating The SPRY Framework***
- ***Using Flex Builder 2***
- ***Flex for Java Developers***
- ***ActionScript 3.0 Tips and Tricks***
- ***How To Use ColdFusion with Flex***
- ***Leveraging Flash***
- ***Preparing for Adobe Apollo***
- ***MXML Master Class***

Who Should Attend?

- ***CEOs and CTOs***
- ***Senior Architects***
- ***Project Managers***
- ***Web programmers***
- ***Web designers***
- ***Technology Evangelists***
- ***User Interface Architects***
- ***Consultants***
- ***Anyone looking to stay in front of the latest Web technology!***

Early Bird:
(Hurry for Early Bird Pricing) $395*

Conference Price:
(Register Onsite) $495*

OFFER SUBJECT TO CHANGE WITHOUT NOTICE, PLEASE SEE WEBSITE FOR UP-TO-DATE PRICING

* Golden Pass access includes Breakfast, Lunch and Coffee Breaks, Conference T-Shirt, Collectible Lap-Top Bag and Complete On-Demand Archives of sessions in 7 DVDs!

Sponsored by Adobe, Web Developer's & Designer's Journal

Produced by SYS-CON EVENTS *The World's Leading i-Technology Event Producer* www.events.sys-con.com

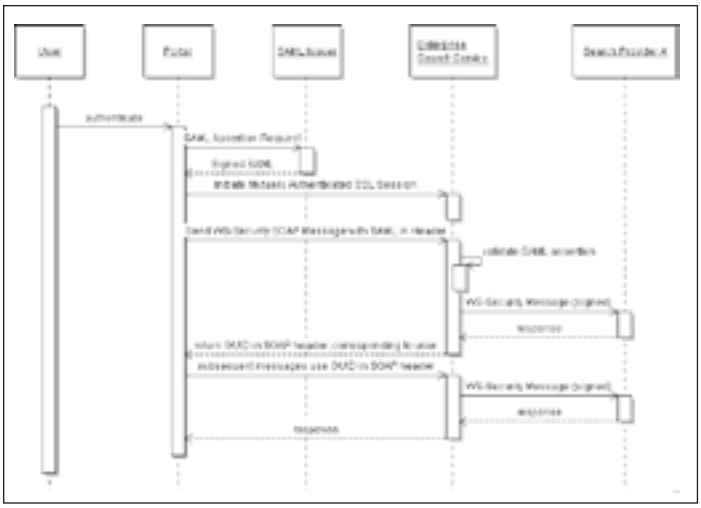


Figure 6: Sequence diagram of using Web Service user session

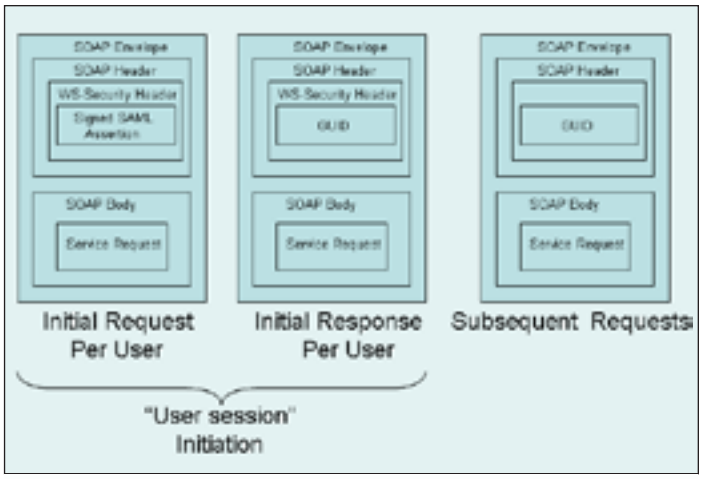


Figure 7: The Web Service messaging profile for establishing user sessions in a mutually authenticated SSL session

new signed SAML assertion for that user is expected for continued activities on behalf of that user.

Using this kind of messaging will only be effective for the life of the SSL session. Because there's mutual authentication using public key cryptography, there's strong assurance of the identity of both parties, and because of the existence of the SSL session, there's a high degree of confidentiality of the data that is passed. Therefore, any reference to the GUID in the session would only be valid while that SSL session lasts.

Figure 6 shows a UML sequence diagram using our portal scenario, applicable to both trust models discussed – in the trusted client trust model, the SAML issuer is a component of the portal, and in the trusted infrastructure model, the SAML issuer is the SAML-issuing authority trusted by the enterprise.

Looking at Figure 6, once a mutually authenticated SSL session is established, the portal passes the signed SAML Assertion in the header of the WS-Security message to the Enterprise Search Service. It should be noted that the full message doesn't have to be digitally signed, because there's already a high assurance of the identity of the portal achieved in the SSL mutual authentication step. The signature of the SAML assertion, however, would be necessary to convey the non-repudiated assurance of the user's identity. Once the Enterprise Search Service validates the authenticity and trust of the SAML assertion, it caches the SAML assertion locally and assigns it a GUID. The final response to the portal will include the GUID in the SOAP header, which can be used for subsequent SOAP messages corresponding to that same user identity.

It should be noted that Figure 6 also propagates the identity between the Enterprise Search Service and one of the search provider Web Services. Because we didn't establish an SSL connection between the search service and the search provider Web Services in this example, WS-Security SOAP messaging would be used, where the Enterprise Search Service would need to sign the entire message using XML Signature, and would include the original signed assertion from the SAML issuer in the message.

Because of this, there are two message profiles that can be used for propagating identities – one based on a mutually authenticated connection and one based on basic WS-Security SOAP Messaging. The messaging profile for the former is seen in Figure 7.

It's important to note that the proposed model can only occur in an SSL session that's been established – otherwise the GUID returned wouldn't be confidential and identity spoofing could easily happen. Another important note is that for each user there must be a "user session" initiation in the SSL session, where the Web Service must gain explicit and non-repudiated trust of the sender's identity. Once the Web Service can validate that trust, it will return the GUID that can be used for that user.

Because a certain "user session" negotiation must also take place for each user, it's important to weigh this in considering using such a solution. If you don't anticipate a Web Service sending a significant number of messages on behalf of its users, you shouldn't use it. If, however, you're providing a solution where strong security and mutual authentication is needed, and where you believe performance would benefit from long-lived SSL connections between certain nodes in your solution, consider using this model.

Conclusion

This article has presented an approach for complementing Web Service Security solutions with long-lived SSL sessions. As with any approach, it's important not to choose security solutions blindly – much of this will depend on your security requirements and enterprise security policy, and the solution that this article presents is based on foreknowledge of the use of some Web Services in your enterprise. If you have similar security requirements, and if you know ahead of time that certain nodes in your Web Service chain will repeatedly communicate (like the enterprise search example presented here), consider using this model. Long-lived SSL connections between certain nodes in your solution could make a big difference in performance.

About the Author

Kevin T. Smith is a technical director at McDonald Bradley, where he leads the SOA & Semantics Security Team (S3T) focusing on securing Web Services for multiple projects. An author of several technology books on XML, Web Services, Java development, and the Semantic Web, he is a frequent speaker at many conferences such as JavaOne, OMG Web Services, the Association for Enterprise Integration (AFEI) and Net-Centric Warfare.

kevintsmith@comcast.net

Welcome to the Future of Video on the Web!

REGISTER NOW!
www.iTVcon.com

CALL FOR PAPERS NOW OPEN!

LIVE SIMULCAST!
AROUND THE WORLD ON SYS-CON.TV

iTVCON.COM
INTERNET TV CONFERENCE & EXPO 2006

Coming in 2006 to New York City!

"Internet TV is wide open, it is global, and in true 'Web 2.0' spirit it is a direct-to-consumer opportunity!"

For More Information, Call 201-802-3023
or Email itvcon@sys-con.com

Welcome to the Future!

Did you already purchase your ".tv" domain name?
You can't afford not to add Internet TV to your Website in 2006!

2005 was the year of streaming video and the birth of **Internet TV**, the long-awaited convergence of television and the Internet. Now that broadband is available to more than 100 million households worldwide, every corporate Website and every media company must now provide video content to remain competitive, not to mention live and interactive video Webinars and on-demand Webcasts.

20 years ago the advent of desktop publishing tools opened the doors for the creation of some of today's well-known traditional print media companies as well as revolutionized corporate print communications. Today, with maturing digital video production, the advent of fully featured PVRs, and significant advances in streaming video technologies, **Internet TV** is here to stay and grow and will be a critical part of every Website and every business in the years to come.

It will also very rapidly become a huge challenge to network and cable television stations: **Internet TV** is about to change forever the $300BN television industry, too.

The Internet killed most of print media (even though many publishers don't realize it yet), Google killed traditional advertising models, and **Internet TV** will revolutionize television the way we watch it today. You need to be part of this change!

Jeremy Geelan
Conference Chair, iTVCon.com
jeremy@sys-con.com

PRODUCED BY SYS-CON EVENTS

List of Topics:

- 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

Track 1	Corporate marketing, advertising, product and brand managers
Track 2	Software programmers, developers, Website owners and operators
Track 3	Advertising agencies, advertisers and video content producers
Track 4	Print and online content providers, representatives from traditional media companies, print and online magazine and newspaper publishers, network and cable television business managers

Rich Internet Applications: AJAX,

www.AjaxWorldExpo.com

AJAXWORLD™

CONFERENCE & EXPO

SANTA CLARA SILICON VALLEY

SYS-CON Events is proud to announce the first-ever AjaxWorld Conference & Expo 2006!

The world-beating Conference program will provide developers and IT managers alike with comprehensive information and insight into the biggest paradigm shift in website design, development, and deployment since the invention of the World Wide Web itself a decade ago.

The terms on everyone's lips this year include "AJAX," "Web 2.0" and "Rich Internet Applications." All of these themes play an integral role at AjaxWorld. So, anyone involved with business-critical web applications that recognize the importance of the user experience needs to attend this uniquely timely conference – especially the web designers and developers building those experiences, and those who manage them.

CALL FOR PAPERS NOW OPEN!

We are interested in receiving original speaking proposals for this event from i-Technology professionals. Speakers will be chosen from the co-existing worlds of both commercial software and open source. Delegates will be interested learn about a wide range of RIA topics that can help them achieve business value.

Flash, Web 2.0 and Beyond...

REGISTER TODAY AND $AVE!

"It Was The Best AJAX Education Opportunity Anywhere in the World!" —John Hamilton

October 3-4, 2006

Santa Clara Convention Center
Hyatt Regency Silicon Valley
Santa Clara, CA

To Register
Call 201-802-3020 or
Visit www.AjaxWorldExpo.com

May 7-8, 2007

First International AjaxWorld Europe
Amsterdam, Netherlands

"Over the two information-packed days, delegates will receive four days' worth of education!"

Topics Include...

Themes:
- > *Improving Web-based Customer*
- > *Interaction*
- > *AJAX for the Enterprise*
- > *RIA Best Practices*
- > *Web 2.0 – Why Does It Matter?*
- > *Emerging Standards*
- > *Open Source RIA Libraries*
- > *Leveraging Streaming Video*

Technologies:
- > *AJAX*
- > *The Flash Platform*
- > *The Flex 2 Framework & Flex Builder 2*
- > *Microsoft's approaches: ASP.NET, Atlas, XAML with Avalon*
- > *JackBe, openLaszlo*
- > *JavaServer Faces and AJAX*
- > *Nexaweb*
- > *TIBCO General Interface*

Verticals:
- > *Education*
- > *Transport*
- > *Retail*
- > *Entertainment*
- > *Financial Sector*
- > *Homeland Security*

GROUP DISCOUNTS AVAILABLE:
— 5 Delegates from Same Company for only $995 (each) —
— Register a Second Person for only $1195 —

LIVE SIMULCAST!
AROUND THE WORLD ON SYS-CON.TV

Receive FREE WebCast Archives of Entire Conference!

The best news for this year's conference delegates is that your "Golden Pass" registration now gives you full access to all conference sessions. We will mail you the complete content from all the conference sessions in seven convenient DVDs after the live event takes place.

► **This on-demand archives set is sold separately for $995**

Early Bird*
(Register Before August 31, 2006)
..... **$1,495****
See website or call for group discounts

Special Discounts*
(Register a Second Person)
..... **$1,395****
See website or call for group discounts

(5 Delegates from same Company)
..... **$1,295/ea.****
See website or call for group discounts

On-Demand Online Access
(Any Event)
..... **$695**

*Golden Pass access includes Breakfast, Lunch and Coffee Breaks, Conference T-Shirt, Collectible Lap-Top Bag and Complete On-Demand Archives of sessions in 7 DVDs!

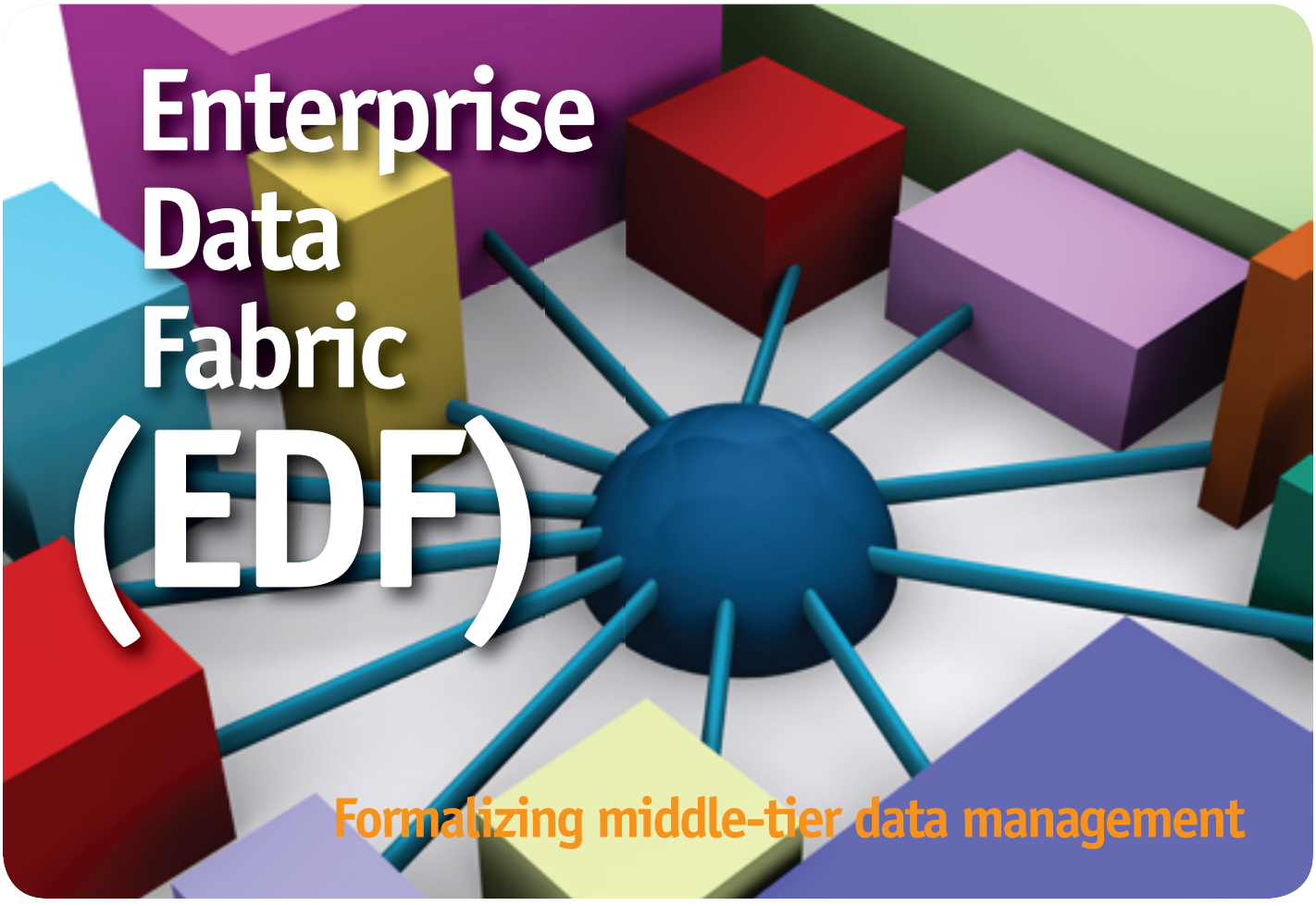
**OFFER SUBJECT TO CHANGE WITHOUT NOTICE, PLEASE SEE WEBSITE FOR UP-TO-DATE PRICING

SYS-CON EVENTS

For more great events visit www.EVENTS.SYS-CON.com

VISIT WWW.AJAXWORLDEXPO.COM FOR THE MOST COMPLETE UP-TO-DATE INFORMATION

Enterprise Data Fabric (EDF)



Formalizing middle-tier data management

WRITTEN BY **BHARATH RANGARAJAN**

Performance, scalability, and availability of traditional n-tier architectures have become enormously important to most IT architects. First, most meaningful business processes have at least a few real-time sub-processes, where latency and throughput are extremely critical factors to satisfy the mandated service level agreements. Second, data is becoming a dynamic entity, with an explosive increase in the number and types of data sources. With the advent of industry trends like algorithmic trading in capital markets and Radio Frequency Identification (RFID) in retail supply chains, instant analysis and distribution of relevant data streams is becoming a key requirement in most industries.

To address these challenges of modern IT, infrastructures such as J2EE (or Java EE going forward) need to support event-driven architectures in addition to traditional request-reply models. Such IT prerequisites drive the need for sophisticated middle-tier data management as a foundation for building, deploying, and operating mission-critical applications.

Enter EDF

Conceptually, an enterprise data fabric (EDF) represents a distributed middle-tier operational infrastructure that leverages main memory and disk across multiple disparate hardware nodes to store, analyze, distribute, and replicate data. By managing operational information in the middle tier, an EDF promotes the co-location of applications and the data they require as opposed to a typical siloed approach. By combining the essential features of caching, databases, messaging, and event processing, an EDF engenders a "distributed, active" data management approach that's quite antithetical to the "centralized, passive" approaches traditionally adopted.

Data Persistence and Virtualization

An EDF supports multiple distributed data caching and persistence topologies, virtualizing information from multiple data

sources to guarantee high performance with location transparency, which enables consuming applications to use a single API to instantly access data irrespective of the underlying data source. Data can be managed in multiple heterogeneous formats (objects, tables, XML documents) and accessed via multiple programmatic and query languages such as Java, C++, SOAP, Xpath, SQL, and OQL.

Reliable Data Distribution

To support deployments like computational grids that potentially span hundreds or even thousands of nodes, an EDF provides a high-speed transport layer that utilizes multiple protocols like TCP/IP and Multicast. This layer enables scalable delivery of data across distributed members in a reliable fashion. Further, an EDF's distributed event notification service lets updates to a data element be propagated through a listener framework. Data distribution in an EDF may be push-based (changes propagated on update) or pull-based (changes propagated only on request). With the push-based model, distribution can be configured as synchronous or asynchronous. Unlike an enterprise messaging system, the object-based programming model is intuitive without the headache of message format, message headers, payload, etc. Sharing data objects combined with event notification enables parallel data processing in grid-like topologies.

Continuous Analytics

Besides storing and virtualizing static data elements, an EDF can also manage fast-changing data streams through a continuous querying model in which new events are constantly analyzed against pre-defined queries or patterns of interest. This model is optimized to rapidly determine queries affected by a particular event and notify relevant client applications via callback interface with almost no latency. An EDF provides a storage model for events that simultaneously allows correlation and querying across both real-time and non-real-time data.

High Availability

For business-critical operations, it's essential to ensure data availability at all times and avoid single points of failure (SPOF). An EDF provides high availability through replication to one or more "mirror" (backup) cache nodes. A mirror synchronously gets all the data changes from the primary nodes across the entire distributed system guaranteeing 100% backup of the data at all times. An EDF also supports disk persistence to guarantee data recovery even in scenarios involving complete application(s) failure.

EDF in Today's SOA Infrastructure

As IT organizations strive to deploy SOA infrastructures to support real-time business process, middle-tier management via an EDF-like infrastructure becomes an absolute necessity. An EDF serves as data middleware augmenting other Service Oriented Architecture (SOA) components. Just as application servers like WebLogic provide a container for applications, an EDF acts as a data container that can be embedded in the JVM of an application server instance or shared across a cluster(s) of application servers. In relevant scenarios, an EDF can act either as a JTS-compliant transaction manager or participate in container-managed transactions as an XA resource.

In the broad context of an SOA, an EDF serves as a data middleware complement to an Enterprise Service Bus (ESB), handling service/application integration, routing, and data transformations. Just as an ESB manages service flow and control, an EDF provides a distributed operational data storage layer from which ESB components can access data in multiple formats and in a highly available fashion. Alternatively, an EDF can expose the data held in the fabric as a service (with WSDL port types and operations) easily plugging into an ESB as one of the services. An EDF also complements the data mediation services offered via typical Service Data Objects (SDO) implementation by providing supplementary functions such as data caching, data distribution, and continuous analytics. Similarly, business process management (BPM) engines in SOAs can use a data fabric as a distributed persistence layer for contextual state and frequently accessed data. These strategies address another extremely important dimension of service orientation, which relates to service latency and availability. ■

About the Author

Bharath Rangarajan is director of product marketing at GemStone Systems, where he oversees product positioning and market strategy for the GemFire product line. He has more than seven years of experience in enterprise software dealing with data management issues in functional areas such as EAI, B2B collaboration, and supply chain management. He has led technical and product teams at companies including i2 Technologies, SeeBeyond Corp. and Candle Corp.
bharath.rangarajan@gemstone.com

XML Compression and its Role in SOA Performance

Dealing with the increased size of SOA payloads

WRITTEN BY **AKASH SAURAV DAS** AND **SRINIVAS PADMANABHUNI**

Looking at the uncertainty and volatility of market conditions today, enterprises are depending on new cutting-edge technology to have an edge over their fierce competitors. At the same time, they try extracting more value from their existing IT investments. Adding to these disparate applications and technologies are the acquisitions and mergers that inherently bring in different sets of applications.

Better integration of these myriad applications built on different technologies clearly makes them more valuable. Using Service Oriented Architecture (SOA), enterprises can not only achieve better integration but also be future-ready as an agile enterprise that can swiftly respond to change in business processes.

XML and Its Role in SOA

XML is emerging as the lingua franca of data representation and exchange across applications interacting in an SOA world. A close look at the standards stack for SOA (Figure 1) shows that XML is the foundation for all the Web Services standards like XML Schema, SOAP, WSDL, and UDDI. These standards leverage the core concept of XML-based representations to carry out information interchange between service providers and requestors in a SOA.

Notwithstanding the core syntactic standards of SOA as shown in Figure 1, semantics is another important dimension that plays a crucial part in communication between a service provider and a service consumer in an SOA infrastructure and requires that the contents of the messages be mutually understood, which leads to the requirement of semantic interoperability.

XML solves the semantic interoperability problems associated with working with different data formats in different applications across multiple platforms. Different vertical business domain stakeholders have come to together and defined shared XML-based vocabularies to solve the semantic interoperability issue. (See http://xml.coverpages.org for a comprehensive list of such standardization efforts.) Using XML inherently brings ease of representation since it's text-based, flexible, and extensible. The platform- and language-independence of XML has catalyzed it as SOA's mainstream representation format.

SOA Performance Challenge and XML Compression Solutions

While self-describing XML-based service descriptions and messages in SOA make the data exchange easier, lending reusability and extensibility, they also increase the size of the data significantly. This is because the XML message typically contains not only the data as text, but also the format of the data. It contains all the information about the data presentation to the end user like font, size, and style. The verbosity of text-based representation by itself also tends to increase the data size in SOA payloads. So XML data representation not only increases data storage and data transfer times in SOA but also increases data parsing times in the context of a SOA, creating a performance challenge for a SOA.

The following are the salient points driving the need for compressing XML document in the context of SOA:

1. Redundant data in XML documents, e.g., white space, similar node names.
2. Text-based XML document sizes tend to be large.
3. The need for an efficient way to store files based on XML.
4. Large volumes of XML data sent over the network as SOAP payloads.

One category of industry solutions used to solve SOA performance management problems rely on the notion of XML compression. These solutions leverage use compression techniques to reduce the size of the XML payloads being carried in the SOA messages and transfer data in compressed format. However, there is the additional cost of compression/decompression at either end that has to be accommodated when computing the overall cost.

While the issues related to data storage and data transfer times can be resolved to a significant level by using compression techniques, the problem related to the processing overhead can be solved using both software and hardware solutions. A variety of tools and methodologies are already on the market to overcome XML processing limitations. Some prominent categories of tools and technologies that help overcome the limitations associated with using XML are briefly mentioned here:

XML Hardware

Large XML data processing will consume enormous amounts of CPU, memory, and network bandwidth. Traditionally there were processors that did general-purpose processing, but with the advent of XML and XML-based applications a new breed of custom acceleration processors are being developed. This specialized hardware, called XML accelerators, not only accelerate time-consuming tasks like XSL transformation and schema validation, but security-related features like encryption. These operate over networks and perform XML processing at wirespeed. XML accelerators are network devices that offload overtaxed servers by processing XML at a higher speed.

Compact Representation

A key premise in this approach is to use a compact representation to compact the size of the message being carried around. One mechanism is to have XML transferred in compact encodings like Abstract Syntax Notation. The usual textual format of XML offers no way to determine the end of a data value; hence the application has to examine every byte received. In this case the time consumed is increased and performance isn't that great. A different approach would be to represent XML in a binary format such as Abstract Syntax Notation number One (ASN1). This notation is associated with standardized encoding rules such as the Basic Encoding Rules (BER) and Packed Encoding Rules (PER) and is useful for applications that have bandwidth restrictions. This significantly reduces the time consumed and enhances performance.

XML Cache/Component Parsers

Repeatedly used XML data can be cached to reduce XML processing overheads. Similarly specific XML parsers can be used that cater to the specific needs of an SOA application.

XML Software Compression

Since XML is text-based, we can use gzip, bzip, etc. like techniques that leverage Lempel-Ziv and Huffman Encoding Algorithms for compression. These compression techniques are generic text compressors and they're effective and have very good compression ratios too. These techniques are good for sequential data, but unlike normal text, XML data is tree-structured data. XMILL from AT&T is a focused XML compression technique. It regroups similar XML nodes and uses conventional compressors such as gzip to compress the result of the regrouping of nodes. A comparison of the salient features of gzip and XMILL are as below:

gzip:

- available in both Open Source and commercial implementations
- Provides a good compression rate
- free from patented algorithms
- knowledge of the document structure isn't needed

XMILL

- better compression rate compared to gzip (by a factor of two)
- it separates structure from content
- moderately faster than gzip
- three types of compressors available:
  - atomic compressors for the basic data types
  - Combined compressors
  - User-defined compressors

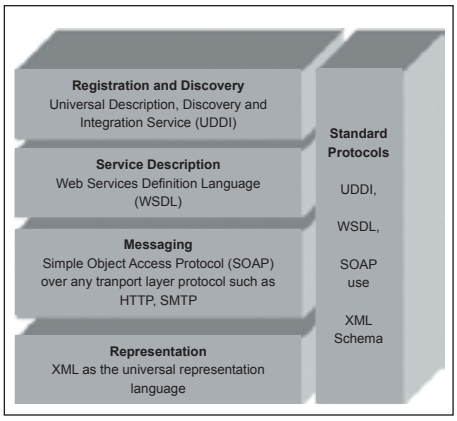


Figure 1

Binary XML Standards: XOP, MTOM & RRSHB

New schemas are being developed to solve the problem of exchanging large documents between the service provider and the consumer. These schemas address the problem of fitting binary data directly into an XML message.

MTOM is a description of how XML-binary Optimized Packaging (XOP) is layered into a SOAP HTTP transport and uses XOP to let SOAP bindings speed up data transmission by selective encoding portions of the XML message. But MTOM uses a MIME package as opposed to XML and has the overhead of MIME processing to base-64 encoding.

Resource Representation SOAP Header Block (RRSHB) sends all the data needed to process the message. It can send a Web resource as a part of the SOAP message. This is specific to those cases where access to the resource is restricted to the body of the message and there is network overhead.

Conclusion

SOA infrastructure relies heavily on XML to be the lingua franca, and effective SOA performance management requires efficient ways of handling XML. XML compression techniques can go a long way in handling the SOA performance challenge. Needless to say, specific application needs are very decisive in choosing a compression technique from the myriad of techniques mentioned in this article. ■

References

1. *XMILL* http://sourceforge.net/projects/xmill
2. *gzip* www.gzip.org
3. *Datapower XML hardware* www.datapower.com/products/xa35.html
4. *Sarvega Hardware* www.sarvega.com/xml-security-products.html
5. www.w3.org/TR/soap12-mtom/
6. www.w3.org/TR/soap12-rep/
7. www.w3.org/TR/soap12-mtom/#XOP

About the Authors

Akash Saurav Das is a technical specialist with the Web Services Center of Excellence in the SETLabs of Infosys Bangalore. His research focuses on XML compression, Web Services security, semantic interoperability, UDDI, and Web Services for remote portlets. He has also been published at international conferences.

akash_das@infosys.com

Dr. Srinivas Padmanabhuni is a principal researcher with the SOA/Web Services Center of Excellence at SETLabs of Infosys Bangalore. His research interests include XML compression, the Semantic Web, SOA, and Web Services. He has published extensively in international conferences and journals and has also spoken on these topics at external forums.

srinivas_p@infosys.com

XML compression techniques can go a long way in handling the SOA performance challenge

OSS Nokalva Announces Fast Web Services Tools

(*Somerset, NJ*) – OSS Nokalva, Inc., a developer of standards based software toolkits for building data communications applications, announced the availability of the OSS Fast Web Services (FWS) Tools for Java, and C/C++.

The OSS Fast Web Services Tools are ideal for those who use Web Services in resource-constrained and/or bandwidth-constrained environments. The FWS Tools, which create and process efficient binary (PER-encoded) SOAP messages, are conformant to the W3C WSDL 1.1, SOAP 1.1 and 1.2, WS-I Basic Profile 1.0, and ITU-T X.892 | ISO/IEC 24824-2 Fast Web Services recommendations. XML SOAP messages are supported, ensuring your ability to exchange SOAP messages with XML based peers.

The Tools consist of three main components: an OSS FWS Compiler, an OSS FWS Runtime, and an optional OSS FWS Transport Layer Runtime.

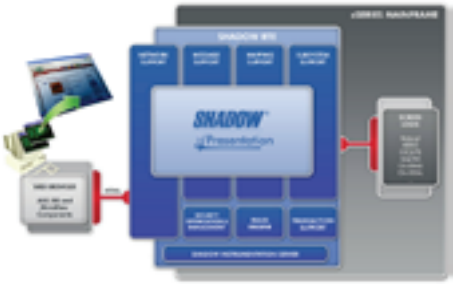
- The OSS FWS Compiler takes a WSDL 1.1 service description as input, and generates schema-derived files for inclusion in your application.
- The OSS FWS Runtime library is a collection of functions for use in writing efficient Web Services applications in Java, C, or C++.
- The Transport Layer Runtime can be used to send/receive SOAP messages.

www.oss.com

DataDirect Technologies Shadow z/Services Accelerates Mainframe SOA Deployment

(*Bedford, MA*) – DataDirect Technologies, a provider of data connectivity and mainframe integration and an operating company of Progress Software Corporation, has announced that the National Rural Electric Cooperative Association (NRECA) has deployed DataDirect Shadow z/Services to streamline the integration of its online customer transactions directly with mainframe data in an effort to improve service levels and the reusability of existing mainframe system resources.

The NRECA Web site currently hosts 13 benefits-related applications, ranging from registering new hires and terminations to employee disabilities. While the front-end system has significantly improved customer



service and operational efficiency, the back-end system remained static, unable to scale with ease or evolve to meet changing business conditions. Recognizing that greater efficiencies and faster service could be achieved by updating the IT infrastructure to support a more flexible, service-oriented architecture (SOA), the NRECA selected DataDirect Shadow z/Services technology for mainframe integration.

Shadow z/Services utilizes the Shadow RTE(TM) foundation architecture, a multi-threaded, native runtime that supports all of the major mainframe integration paradigms – Web services for SOA; real-time events for Event-Driven Architectures (EDA); SQL for direct SQL data access and transactional support; as well as automatic presentation-layer generation for extending screen-based applications to the Web.

www.datadirect.com

Percussion Software Introduces Rhythmyx 6

(*Woburn, MA*) – Percussion Software, a developer of Web Content Management (WCM) software for multi-channel, customer-centric applications, has introduced Rhythmyx 6, the newest version of its enterprise-ready WCM System. Rhythmyx 6 provides a robust platform for managing customer-centric content across multiple sites, multiple channels and multiple lines of business.

Among Rhythmyx 6's major new features are a Web Forms application, a Web Analytics application to be available in the fall and newly enhanced user interface customization capabilities. Other new Rhythmyx 6 features include support for the Velocity templating language, a new Developer's Workbench based on the popular Eclipse Interface Framework and a Web Services Software Developer's Kit (WSDK) with an enhanced Web services API that make it easier for Rhythmyx implementers to support a Web services and service-oriented architecture (SOA) approach to integration.

www.percussion.com

iTKO LISA 3 Supports SOAP 1.2 Protocols within a Complete SOA Testing Platform

(*Dallas*) – iTKO, Inc., a provider of complete testing solutions for service-oriented architecture software, has announced new features for complete Web services testing within its flagship iTKO LISA 3 Complete SOA Test Platform software.

The new features are being included as part of the General Availability release of LISA 3, giving the testing software automatic support for all the primary elements of the SOAP 1.2 protocol, whether the Web services are delivered on the Axis/Apache open source platform, on Microsoft .NET, or other commonly used platforms.

LISA now includes support for most Web services standards commonly encountered within SOA applications:

- WS-Addressing, WS-I, WS-Security and other protocols
- WSDL 2.0 discovery and interoperability
- SOAP 1.2 over HTTP, HTTPs and JMS
- Custom SOAP
- HTTP Headers

iTKO's LISA 3 release includes several significant component and system testing capabilities, including functional and load test suite management, native support for messaging protocols, and an improved workflow process so non-developers can easily learn and test every layer of an SOA application.

www.itko.com

Mindreef®

Quality and productivity
for the entire SOA lifecycle

Corporate Architect
Codify corporate standards, best practices, and governance policy around WSDL and schema

Project Architect
Leverage rules from corporate architects while creating shared workspaces that add documentation, example scripts, and simulation data to enhance the WSDL contract

Service Developer
Use shared workspaces from Project Architects to fully understand requirements. Create unit tests automatically during development

Client Developer
Enter the development process earlier with Mindreef's auto documentation and simulation features

Support Staff
Capture all associated problem artifacts into a Mindreef Shared Workspace*, allowing any team member to reproduce a problem at the push of a button

Testing
Leverage architect's example scripts and developer's unit tests to quickly and easily create robust regression tests

Mindreef
SOAPscope®

Industry-leading Web services testing and diagnostics

Mindreef
SOAPscope Server™

Design, develop, test, and support Web services as a team

- **Testing**
- **Diagnostics**
- **Governance**
- **Support**

Mindreef's lifecycle approach—from well-adopted best practices early in the process to efficient problem resolution post-launch—drives down the cost of developing and maintaining Web services, leading to more efficient, higher-quality SOA. Mindreef products are designed to leverage your most valuable assets—your service team members—to achieve sustainable business results.

From your first Web service project to large SOA initiatives. Download FREE trials at www.mindreef.com, or to speak with a Mindreef representative, call 603-465-2204 x581.

Mindreef®
QUALITY-DRIVEN SOA™

*patent pending

Ensure Interoperability.

Validate functionality.

Eliminate security vulnerabilities.

Test performance & scalability.

Confirm compliance.

Collaborate and reuse.

Ensure Secure, Reliable Compliant Web Services

PARASOFT®

SOAtest™

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

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

If you are building serious web services, you need SOAtest. For more information regarding Parasoft SOAtest, call 888-305-0041 (x-3501).

Download a copy of SOAtest for a free evaluation today at www.parasoft.com/WSJmagazine

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

PARASOFT®

We make software work.™

Automated Error Prevention™