SYS·CON

NOW INCORPORATING XML JOURNAL

Web Services

.NET J2EE XML JOURNAL

June 2004 Volume 4 Issue 6

POA

Building SOA from the Ground Up

Moving process to the forefront

pg 22

RETAILERS PLEASE DISPLAY
UNTIL JULY 31, 2004

$9.99US $9.99CAN

06

0 09281 03420 9

Introducing OpenLink Virtuoso

A CROSS-PLATFORM Universal Server that transparently integrates disparate application databases and disparate applications across your enterprise. It provides a Virtual Database, XML Database, SQL Database, Web Application Server, and Web Services Platform as part of a **single-server solution.**

Empower your organization today, download a FREE evaluation copy from www.openlinksw.com/virtuoso or call us on 1 800 495 6322

Database Support
Oracle, SQL Server, DB2, Sybase, Informix, CA-Ingres, Progress, MySQL, PostgreSQL, and other ODBC or JDBC accessible databases.

Protocol Support
SQL, ODBC, JDBC, XML, XSL-T, XQuery, XPath, HTTP, WebDAV, NNTP and POP3

Platform Support
Windows, UNIX (all major flavors), Linux, Free BSD, and Mac OS X

Programming Languages
Any .NET bound language, Java, C/C++ etc.

Web Application Hosting
ASP.NET, Java Server Pages, PHP, Perl, and Python

Runtime Environment Hosting
Microsoft .NET, Mono, Java, Perl, Python, PHP, and others

www.openlinksw.com/virtuoso

OPENLINK SOFTWARE

WebAppCabaret™

J2EE Web Hosting

http://www.webappcabaret.com/wsj.jsp

1.866.256.7973

Quality Web Hosting at a reasonable price...

<JAVA J2EE HOSTING AND OUTSOURCING>

JAVA Developers: Design and Architect with Struts. Develop with Eclipse.Build with Ant. Deploy to WebAppCabaret. It is that easy with a hosting company that understands the Java J2EE standard. Web Services - a buzz word or reality. How many web hosts provide the facilities to deploy and run applications written for Web Services? We Do. At WebAppCabaret, we lead the way in providing comprehensive Java J2EE and PHP/Perl Web Hosting solutions.

Easy Application Deployment is a reality at WebAppCabaret. Each Account's J2EE Application Server is installed with its own instance and default settings so you have complete control. Our Web Control Panel makes it a breeze for JVM restarts. Such is an example of our Advanced Web Hosting Infrastructure and Tools. If you are a consultant, do you have a complex web hosting requirement for your client? Web Host or Reseller, are you looking to provide services without the headaches of managing your own network infrastructure? Look no more.

For JAVA J2EE we offer the latest versions of Tomcat, JBoss, and Jetty Application Servers. For Scripting programming we offer the latest PHP and Perl. We also offer the latest MySql and PostgreSql Databases. Commercial software, including Resin, JRun, and Oracle, is also available for an extra charge. In addition we provide valuable tools such as Bugzilla Bug Tracking, CVS Version Control plus a whole lot more. Does your service provider offer ENCRYPTED ACCESS for Email (POP3S/SMTPS/IMAPS), Shell, and FTP? WE DO. All of this is backed by our Tier 1 Data Center with 100% Network Uptime Guarantee.

Below is a partial price list of our standard hosting plans. (Reseller accounts also available). For more details please log on to http://www.webappcabaret.com/wsj.jsp or give us a call at 1(866)-256-7973.

$39/mo Enterprise

Latest JBoss/Tomcat/Jetty
Latest JSP/Servlets/EJBs
Private JVM
Choice of latest JDKs
Dedicated IP Address
NGASI Control Panel
PHP and Perl
Web Stats
1GB Disk
200MB DB
MySql
PostgreSql
Dedicated Apache
Telnet . SSH . FTP
5 Domains
100 Emails
Web Mail . POP . IMAP
more...

Options

The following are some of the add-on options with the associated extra starting at costs:

Resin: $10/mo

300MB Oracle: $40/mo

JRun: $20/mo

ColdFusion: $30/mo

$191/mo Dedicated

Managed Dedicated Server starts at $191 per month for:
256MB RAM
Pentium 4
40GB RAID
Firewall
Unlimited Domains
Unlimited Web Site Hosting
NGASI Control Panel with your own Logo
Each dedicated server configured for standalone web application and web hosting (resellers) at no extra charge.
more...

$1500/mo TriBalance

TriBalance is our introductory Load Balancing service comprised of:
1 Database server
4GB RAM
Dual Xeon
2 Application Servers
2GB RAM
Single Xeon
more...

$99/mo Reseller

If you cannot afford a Dedicated server for reselling web hosting, for $99/mo the Shared Reseller plan gives you:

10 Professional Plans
Your Web Site
NGASI Control Panel with your own Logo
50% Discount
No System Admin
Easy Account Mgt

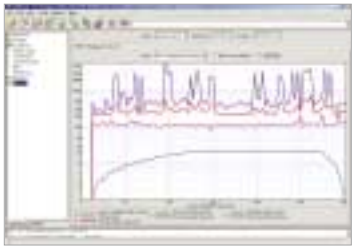
more...

Prices, Plans, and Terms subject to change without notice. Please log on to our website for the latest Prices and Terms.

Web Services Journal
.NET J2EE XML
XML Journal
JUNE 2004 Volume 4 Issue 6

Inside WSJ

WSJ: Features



Verifying Algorithms Through Unit Testing

Complicated algorithms are typically contained within one class or function so that developers can easily manage them. How then, can you test the functionality of your algorithms and verify that they correctly solve problems before you have a full application written? The answer lies in unit testing.

Unit testing is the perfect strategy for verifying and testing algorithms. Unit testing involves testing software code at its smallest functional point, which is typically a single class. Each individual class should be tested in isolation before it is tested with other units or as part of a module or application. By testing every unit individually, most of the errors that might be introduced into the code over the course of a project can be detected or prevented entirely.

I suggest that you investigate the benefits of performing unit testing to verify the functionality of your algorithms. Performing thorough unit testing reduces the amount of work you will need do at the application level, and drastically reduces the potential for errors.

By the way, unit testing is a fundamental part of Parasoft's Automated Error Prevention (AEP) Methodology. You can find more information about Automated Error Prevention at **www.parasoft.com**

– Adam Kolawa, *Ph.D.*
Chairman/CEO of Parasoft

LinuxWorld® Conference & Expo

CONFERENCE: August 2 – 5, 2004 EXPO: August 3 – 5, 2004

MOSCONE CENTER • SAN FRANCISCO, CA

Where OPEN MINDS Meet

LinuxWorld Conference & Expo is the #1 venue for decision-makers and influencers to discover real business solutions for real business problems, learn how Linux is accelerating as a total enterprise solution, and understand how it can be applied effectively to save their companies time and money.

Attend LinuxWorld and...

- Realize Linux as more than an operating system, but as **a world of applications**
- Evaluate the **latest applications and innovations** from leading open source companies
- Explore interoperability issues and opportunities in **open source and proprietary environments**
- Stay on the cusp of **emerging technologies** and the acceleration of open source adoption in enterprise computing
- Review the latest open source initiatives, their deployment and successes to help **make informed decisions** for your company

WWW.LINUXWORLDEXPO.COM

Register Online With Priority Code: D1401

CORNERSTONE SPONSOR

ORACLE®

PLATINUM SPONSORS

DELL™ IBM intel Novell®

IDG WORLD EXPO

Web Services Journal
.NET J2EE XML

INTERNATIONAL ADVISORY BOARD
Andrew Astor, David Chappell, Graham Glass, Tyson Hartman, Paul Lipton, Anne Thomas Manes, Norbert Mikula, George Paolini, James Phillips, Simon Phipps, Mark Potts, Martin Wolf

TECHNICAL ADVISORY BOARD
Bernhard Borges, JP Morgenthal, Andy Roberts, Michael A. Sick, Simeon Simeonov

EDITORIAL
EDITOR-IN-CHIEF
Sean Rhody sean@sys-con.com
XML EDITOR
Hitesh Seth
INDUSTRY EDITOR
Norbert Mikula norbert@sys-con.com
PRODUCT REVIEW EDITOR
Brian Barbash bbarbash@sys-con.com
.NET EDITOR
Dave Rader davidr@fusiontech.com
SECURITY EDITOR
Michael Mosher wsjsecurity@sys-con.com
RESEARCH EDITOR
Bahadir Karuv, Ph.D Bahadir@sys-con.com
TECHNICAL EDITORS
Andrew Astor aastor@webmethods.com
David Chappell chappell@sonicsoftware.com
Anne Thomas Manes anne@manes.net
Mike Sick msick@sys-con.com
EXECUTIVE EDITOR
Gail Schultz gail@sys-con.com
MANAGING EDITOR
Jennifer Van Winckel jennifer@sys-con.com
EDITOR
Nancy Valentine nancy@sys-con.com
ASSOCIATE EDITOR
Jamie Matusow jamie@sys-con.com
ASSISTANT EDITOR
Torrey Gaver torrey@sys-con.com

PRODUCTION
PRODUCTION CONSULTANT
Jim Morgan jim@sys-con.com
LEAD DESIGNER
Richard Silverberg richards@sys-con.com
ART DIRECTOR
Alex Botero alex@sys-con.com
ASSOCIATE ART DIRECTORS
Louis F. Cuffari louis@sys-con.com
Tami Beatty tami @sys-con.com

CONTRIBUTORS TO THIS ISSUE
Adam Blum, Suresh Damodaran, Jeannine Hall Gailey, Rick Hollar, Kerrie Holley, Paul Kaiser, David Linthicum, David Regan, Sean Rhody, Jonathan Rosenberg, Shone Sadler, Keith Swenson

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

Can I Be of Service?

WRITTEN BY
SEAN RHODY

When I started to think about writing this month's column I looked on the Internet for a good way to define service-oriented architecture (SOA). Some of the definitions were interesting, like "A Service Oriented Architecture is basically a Collection of Services" (www.service-architecture.com/). Others were a little bit more technical, such as "SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents" (www.xml.com). The definitions varied considerably, and often included notes that SOA is not a new concept, things like DCE and CORBA had implemented them years (or even decades) earlier.

I wasn't able to come up with one concise definition of service-oriented architecture, and that's probably okay. A single definition would likely be too narrow and restrict its usefulness. But both of the definitions I quoted above seem to hit significant aspects of this type of architecture. A service-oriented architecture is a collection of services, with a service fitting the very loose definition of "something that does a significant unit of work." That unit of work can be a business-related process, like adding a customer or taking an order, or it could be a support-related process, such as processing user login or keeping track of audit information. Services can rely on one another; they can be built on one another (that's the concept of interacting software agents from the second definition); and really the main requirement for a service is that its interface be easy to use (this is where Web services comes in) and that it be well defined (this is where architectural skill and best practices come in).

Service-oriented architecture is lauded as "the next big thing" in today's IT world – a second coming of a concept that should have freed applications from duplication of effort and lead to IT nirvana. While SOA is a worthy concept in many places, it does have some issues.

Implicit in SOA is the concept that services are shared. For example, rather than have each application build its own security module, a common, shared security service is built. In practice, this is easier said than done, especially when you are trying to abstract business processes rather than support processes. Often there are differences in processes between organizational groups, and of course that doesn't take into account the political hot potato of "who owns the service." Often, SOA requires not just a change in application design, but a change in who funds, manages, and owns services as well. The technical challenges are minor compared to getting buy-in from business and other organizations that have a stake in how technology is used. Even assuming consensus among organizations, differences in business processes can make it impossible to present a single service, even when conceptually the service is identical. Some of this may be mitigated by the adoption of a business process engine that can work behind the scenes of a service to allow customization of process. Or it can be hard-coded into the service implementation in some fashion, although such approaches typically become spaghetti code rather quickly.

Another challenge is the loose coupling that is implicit with an SOA. While this is not a bad thing, and in fact has many benefits, it also has certain drawbacks when creating an application. UI developers, especially those within a corporate firewall who work with thick-client tools, are accustomed to closer integration to data than services provide. If a VB developer is to work with a Web service instead of a database, he loses all his familiar methods of dealing with tabular data sets via the common ODBC mechanisms. This can definitely impact productivity, and even to a certain extent affect the way services are designed, as things like filtering drop-down lists based on screen choices (select a state for example, and the list of counties next to it is populated and filtered with those of that particular state), which used to be performed via database queries, now have to invoke a service. In practice, this makes service invocations much more chatty and fine grained than what is usually envisioned by proponents of SOA. In a thick client, some of this can be handled by more logic on the client (which is usually something that architects are trying to get away from), but in a thin client it has to be handled somewhere on a server.

This month's focus is service-oriented architecture. It's the next new thing, but it's not nirvana. Make sure you understand the drawbacks as well as advantages when you decide to go to this brave new world.

About the Author

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

sean@sys-con.com

Mindreef News
Volume 2 June 2004 www.mindreef.com

Retooling for a Service Oriented Application Lifecycle

Service Oriented Architecture

Service Oriented Architecture (SOA) promises to reduce costs, improve productivity, ease integration, and lead to platform and technology independent software. SOA-based applications are collections of services with well defined interfaces that are loosely-coupled and communicate with one another over network protocols.

Web services are the current technology of choice for developing and deploying applications following the principals of Service Oriented Architecture. Web services certainly changes how we develop applications, but what other impact does it have on the software development lifecycle?

Composite Applications

SOA introduces "composite applications" which deliver overall business functionality from a collection of cooperating services. The individual services making up the composite applications might be highly distributed across different physical machines and may even comprise services spanning different organizations.

The implications of an application spanning organizations are staggering. We no longer have control over deployment versions and timing. We don't have the luxury of attaching a source-level debugger to every component of our application. We also lose the opportunity for pre-deploy integration testing of the complete application. SOA and composite applications truly change the traditional software lifecycle. We now have to consider and accommodate continuous deployment and integration.

Continuous Integration

With loosely-coupled Web services we lose the traditional "integration" phase of application development. Services are discovered at runtime, integrating new components that we may have never encountered before.

For example, a manufacturer might require all parts providers to implement certain Web services to conform to their procurement standards. The manufacturer deploys their procurement system, but vendors come "online" one-by-one as they complete the implementation of their individual services.

The manufacturer may not have the luxury of integration testing with every individual vendor, much less integration testing of the entire composite application consisting of services from a variety of partners.

The challenges of continuous integration are compounded by versioning. Every partner delivering services for the overall composite application may update or upgrade their portions on different schedules. Service interfaces may need to be updated but not everyone will upgrade at the same pace.

The implications of an application spanning organizations are staggering

Lifecycle Changes

SOA suggests a dramatic impact on our traditional software development lifecycle. We will refer to this modified development lifecycle as the Service Oriented Application Lifecycle (SOAL). So, how does SOAL change the development lifecycle?

Individual services certainly may be developed with a traditional "design, develop, integrate, test" lifecycle. The lifecycle changes as we combine these services into composite applications, especially when those services are outside of our control. Now, how do we test this composite application? What is the deployment process for a service that has many dependent services and also itself depends on other services? How do we deal with versioning issues? How do we debug composite applications made of many distributed services?

We certainly don't have all the answers today, but the SOAL does suggest a need for a new breed of tools and approaches to testability of composite applications.

Service Oriented Lifecycle Tools

Traditionally we had great control of our application development lifecycle. Our tightly integrated stand-alone applications gave us complete control over development, testing, integration, deployment and versioning. The introduction of loosely-coupled applications with Service Oriented Architecture produces new lifecycle challenges. SOA has changed how we architect, design and develop software. SOAL also changes how we integrate, test, deploy and version our software systems. The Service Oriented Application Lifecycle will require some retooling to accommodate these changes. SOAPscope is the first in a line of products meeting the needs of the Service Oriented Application Lifecycle.

SOAPscope was designed to be used in environments that span organizations. We recognize that you can't attach a debugger at every node of your composite application. SOAPscope's approach is to debug at the message level, not the source level. Composite applications are non-invasively monitored for SOAP messaging on the wire.

SOAPscope also offers a suite of tools providing value to the developer and tester. These tools, applied to both SOAP and WSDL, allow detection of problems, recreating a bug, testing for boundary conditions, detecting errors due to versioning, and analysis for interoperability issues. SOAPscope has proven itself as a valuable tool for developer and testers, usually paying for itself in its first use.

But SOAPscope is not just a development tool, it is a Service Oriented Application Lifecycle tool. It goes beyond debugging and testing during development to provide value during the deployment and maintenance of a production system.

Imagine that your Web services management system detects a fault: what is the next step? Recreate the problem against a staging server to help to debug and fix the problem. SOAPscope allows you to capture the context of the problem, readily recreate it, and determine the root cause of the problem.

Another use for SOAPscope in production is to proactively detect potential problems.

For example, you might discover with SOAPscope that a particular service or client within the composite application is not truly WS-I Basic Profile compliant. That portion of the application can be dealt with before any real errors surface.

Think of SOAPscope as a tool for continuous quality throughout the new Service Oriented Application Lifecycle. Try SOAPscope for FREE today at www.mindreef.com.

untangle your integration woes with

Assande EAI Messaging Suite

Assande EAI Messaging Suite features

- XML Schema Repository
- Data Abstraction, Transformation, and Mapping
- Message Assembly and Generation from Components
- Messaging Integration Service Deployment
- Automated Naming Standards, Version Control, and DIFF Functions
- Infrastructure Utilization and Pipeline Reporting
- Conversation Deployment and Message Release Management
- State Management and Integrated Workflow
- Advanced Search Capabilities
- Integration Documentation (MS Word, Excel) Repository

ASSANDÉ™
INTEGRATION MANAGEMENT COMPANY

http://www.assande.com

WebServices JOURNAL
.NET J2EE XML

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

ADVERTISING
SENIOR VP, SALES & MARKETING
Carmen Gonzalez carmen@sys-con.com
VP, SALES & MARKETING
Miles Silverman miles@sys-con.com
ADVERTISING DIRECTOR
Robyn Forma robyn@sys-con.com
DIRECTOR, SALES & MARKETING
Megan Mussa megan@sys-con.com
ASSOCIATE SALES MANAGERS
Kristin Kuhnle kristin@sys-con.com
Beth Jones beth@sys-con.com

SYS-CON EVENTS
PRESIDENT, SYS-CON EVENTS
Grisha Davida grisha@sys-con.com
CONFERENCE MANAGER
Lin Goetz lin@sys-con.com

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

SYS-CON.COM
VP, INFORMATION SYSTEMS
Robert Diamond robert@sys-con.com
WEB DESIGNERS
Stephen Kilmurray stephen@sys-con.com
Christopher Croce chris@sys-con.com
ONLINE EDITOR
Lin Goetz lin@sys-con.com

ACCOUNTING
FINANCIAL ANALYST
Joan LaRose joan@sys-con.com
ACCOUNTS PAYABLE
Betty White betty@sys-con.com

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

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

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

SYS-CON MEDIA

Aligning IT with Business

WRITTEN BY
JONATHAN ROSENBERG

SOA Is Now a Business Imperative

Integration remains the number one IT priority; fully 60–70% of IT budgets are dedicated to it. Web services makes integrations simpler and cheaper. It makes B2B integrations practical. What businesses demand from a service-oriented architecture (SOA) is dynamic integration capability. They want rapid response to change, to competitors, to new business opportunities.

SOA built with Web services allows IT to build reusable business functions. Truly reusable business functions create efficiency, modularity, and easier integration. The functionality to be reused might be inventory, pricing, purchasing, shipping, and countless others. Most important, reusable business functions create flexibility. Flexibility and adaptability allow a business to respond to changing business conditions, to be agile.

Architectural Underpinnings of SOA

Instead of installing client code on your computer, SOA sends messages on your behalf to a remote computer. Message-based application integration is a key attribute of SOA: the messages are readable, self-describing XML business data.

The move to SOA is especially critical when a business has multiple channels of interaction. A large mutual funds company interacts via channels including the Web, automated phone systems, live customer service, and Web services to the employer's applications.

Today's SOA must be based on loosely coupled middleware like Web services to be flexible. A constant in business is change. Web services integrations are amenable to change. SOAs that rely on Web services do not place any requirements on the underlying technologies – because they are universally accepted standards and are built on the ubiquitous Web infrastructure.

Visibility Challenges SOA Creates

While SOA solves several critical problems for business, it creates new visibility challenges. Security is the first.

Traditional network security does not suffice when SOA is used for B2B integrations. Messages from B2B Web services are carefully designed to work over HTTP and come through firewalls. Incoming is a request for service to which your systems will provide an outgoing response from your company. That might be okay or it might be bad. How would you know? It is vital that someone monitors who is requesting which service, like the security camera watching a locked door.

Business visibility is another SOA challenge. In the past, since middleware messages have always been binary, proprietary, and opaque, there was no sense in asking them to monitor business information. Now the IT data flows are readable and contain actionable business information. If the goal of SOA is flexible, adaptable, reusable business functionality for integration, it stands to reason that we want to know the business impact fast adaptation actually had.

Deep Visibility Is Needed

For the first time we can map IT data flows to business context. To do so requires a new capability called *deep visibility*. It looks into the full business content of the XML messages and any of their attachments. Such deep visibility delivers the total business context of our SOA and the integrated applications built on it.

Deep visibility lets product managers see in real time which resellers are providing them with the best margins and indicates where to focus special offers or incentives.

Deep visibility lets the secure document repository managers understand who is accessing which critical business documents for what purpose and how often. This leads to improvements in the indexing of the most commonly used documents and metering of usage to create charge-backs.

Deep visibility lets the risk officers do a real-time drilldown on small credit card transactions that might otherwise fall below the radar. They focus on only the relevant portion of the XML message. Isolating first on the number of transactions at a certain suspicious dollar amount, then further on the number of transactions by card number at that amount, they can zero in on the fraudster's IP address.

SOA is a critical business imperative to remain agile and competitive. Be fully prepared for the visibility challenges it creates.

About the Author

Dr. Jothy Rosenberg is a serial entrepreneur and the founder, director, and CEO of Service Integrity, a company that helps businesses see, measure, understand and act in real time on business information flowing across their Web service networks. Prior to this venture, Jothy cofounded GeoTrust. He is also the author of *How Debuggers Work* and *Understanding Web Services Security* (Addison-Wesley) and *Securing Web Services with WS-Security* (Sams). Jothy holds patents on watchpoint debugging mechanisms, content certification, and site identity assurance, as well as a pending security compliance monitoring patent.

jothy@serviceintegrity.com

Extending Your SOA for Intercompany Integration

Find your own value

Service-oriented architecture, or SOA, is the modern notion of connecting systems together at both the information and service levels. Indeed, enterprises are racing to enable their existing applications to externalize services, as well as build the appropriate integration infrastructure around it.

WRITTEN BY
DAVID S. LINTHICUM

However, extending your SOA to automate your business means you must work and play well with other organizations. Managing services, orchestration layers, and connections between companies is not as easy as one might imagine.

Truth be told, while we've understood the value of SOA for some time now, the concept is still new to most enterprises. Not until the advent of Web services did we have a widely accepted standard and enabling technology that allows us to access all types of systems through a common services interface. In fact, we may be at a point in time where more is understood about the technology than the ways in which it fits into the enterprise or value chain. Organizations seem to adopt Web services without thought of strategic fit and function. Adoption is only half the battle.

To address strategic concerns, many enterprises are attempting to figure out how to best leverage SOA within their firewalls, as well as between organizations that are, or should be, part of their business processes. This notion of extending SOA to externalize and share processes and services is really the ultimate destination for SOA; certainly it was the vision for Web services.

Value of Services

Today most interorganization electronic business is conducted using traditional information-oriented mechanisms such as EDI or simple FTP exchanges. These exchanges deal with simple information, and they tend to occur within nightly and weekly batch transfers, meaning that latency is a consideration.

While traditional information-oriented exchanges are the way business gets done today, there are two basic needs for electronic business: the need to have access to information approaching zero latency, and the need to view external customer or supplier systems as sets of remote services as well as clusters of information sinks and sources.

The ability to leverage services will provide organizations with a clear advantage. In addition to the ability to see information in real time, they can abstract application behavior and leverage many remote services inside their enterprise systems as if they were local. This is the basic notion of Web services, so we won't get too deeply into it here.

Access to services implies that business processes existing in and between companies can be coupled at the services layer, meaning that services are shareable (if allowed) among the partner organizations. For instance, the service that defines how inventory is allocated supply chain–wide is shareable, and thus the service is not only consistent, but does not require reinvention within each organization. Moreover, since these services are always visible, information bound to these services is produced and consumed in real time. In essence, you're creating a virtual set of applications that exists between trading partners and allows those trading partners to function like a single entity, and thus service common business processes as if they existed in a single company.

By leveraging this type of architecture, businesses have the opportunity to reduce inventory costs a hundredfold. For example, all manufacturer systems would have service- and information-level visibility into all retail systems, and all of the parts suppliers have the same access, and perhaps their raw materi-

Everything OLD Is NEW Again!

Imagine **XML-enabling** your existing **PowerBuilder** and **Visual Basic** applications

QUICK In hours, without changing a line of code

CHEAP Inexpensively, using your current skills

SAFE Safely, preserving all logic and security

NOW YOU CAN

www.active-endpoints.com

als providers as well. With everyone sharing both information and services, business processes are fully automated and inefficiencies, such as overstocking, understocking, or manufacturing delays, drop dramatically. What's more, and perhaps more important, customer satisfaction goes up since items they demand are available, and at the best possible price.

Functional Components

So, how do you begin sharing your SOA? You must first break the SOAs down into several basic components before attempting integration. These include:

- Private and public services
- Public and private processes
- Data and abstract data
- Monitoring and event management
- Points of integration
- Directory services
- Identity management and security services
- Semantic management

Private and public services refer to services that you create for use within your organization (private) and services that you create to share with your partner organizations (public). These concepts are simple enough, although understanding which services to make public and which to make private requires a bit of analysis.

Public services are those that are redundant within your trading community, such as logistics, inventory, or billing. By exposing these services to outside organizations, you allow them to share the service and thus avoid their own development cost, and also allow them to leverage a shared service as a point of integration and a binding point for common processes.

There are a few key criteria for selecting services that are public, or, exposed to trading partners. First, the service should be redundant to two or more entities. In other words, you solve the same problem for several partners. Second, the service should be unique to the trading community; otherwise it makes sense to look for other public services to solve the problem. Finally, the service should offer ease of integration, including the ability to discover semantics as well as interfaces.

In order to make services public, you must create or leverage an existing shared directory service that allows those outside of the organization to locate, discover, and leverage the service you deem to be public. Directories may be proprietary, LDAP-based, or UDDI-based (or mixed and matched). Typically these directories are public, but support the notion of public and private services, processes, and semantics.

Public and private processes provide orchestration of services, binding them together into a business process to drive information movement and invocation of services. You may consider processes or orchestrations as a group of services gathered together to solve a particular business problem, an overriding control mechanism, if you will (see Figure 1).

There are three types of processes to visualize enterprise and cross-enterprise processes: private, public, and specialized processes.

- Private processes exist at the intra-company level, allowing the business user to define common processes that span only systems that are within the enterprise and not visible to the trading partners or to community-wide processes. For example, the process of hiring an employee may span several systems within the enterprise, but should not be visible to processes that span an enterprise or trading community or other organizations.
- Public processes exist between companies and consist of a set of agreed-upon procedures for exchanging information and automating business processes within a community. This is the core notion of intercompany SOA, since it's really the concept where we create intercompany orchestrations.
- Specialized processes are created for a special requirement, such as collaboration on a common product development effort that only exists between two companies and has a limited life span.

Of course there are standards. BPEL (Business Process Execution Language) focuses on the creation of complex processes by joining together local and remote services, thus leveraging the notion of process integration as well as service-oriented Web services.

In the world of BPEL, process is one of two things:

- Executable business processes that model actual behavior of a participant in a business interaction.
- Business protocols that use process descriptions specifying the mutually visible message exchange's behavior for each of the parties leveraging the protocol (does not reveal internal behavior).

Process descriptions for business protocols are known as abstract processes, and BPEL models behavior for both abstract and executable processes.

> "The ability to leverage services will provide organizations with a clear advantage"

To this end, BPEL leverages a well-defined language to define and execute business processes and business interaction protocols, thus extending the Web services interaction model by providing a mechanism to create meta-applications – process models, really – above the services that exist inside or outside the company. What's both different and compelling about BPEL is the use of a common syntax that is designed to be transferable from process engine to process engine. This is in contrast to other process integration standards, such as BPMI or WFMC, which are more about approaches than a common language. There is more momentum beyond BPEL, and all technology vendors are declaring support for BPEL.

The notion of *data and data abstraction*, in terms of intercompany SOA, lets us think about collections of data or services as abstract entities, thus represented in a form that is most useful to the integration server or the application integration architect. It's this notion that provides for the grouping of related pieces of information, independent of their physical location and structure, as well as defining and understanding what meaningful operations can be performed on the data or services. Thus, we can create any representation needed for data that exists anywhere, and bound to any service.

What's more, we need to separate the implementation from the abstraction itself. This

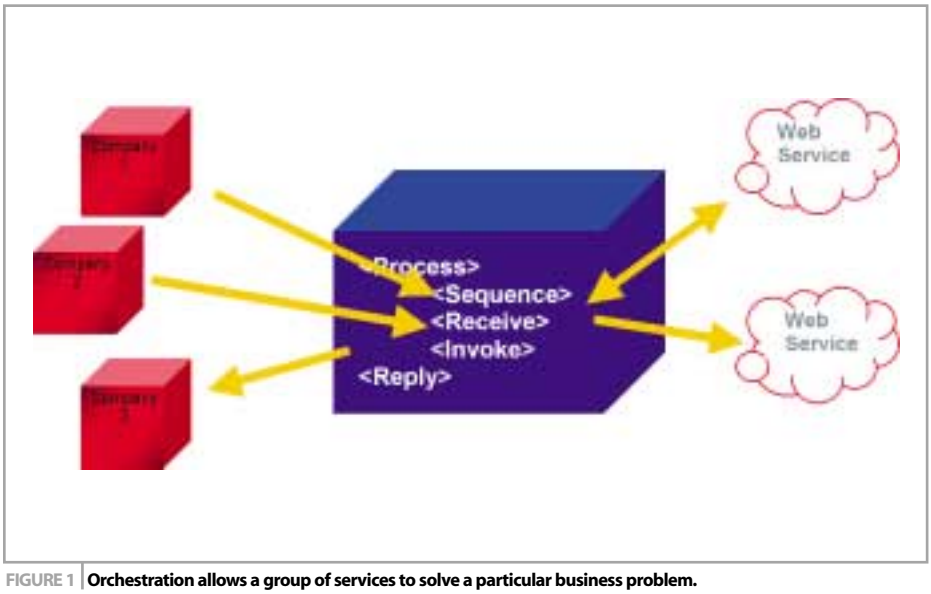


FIGURE 1 **Orchestration allows a group of services to solve a particular business problem.**

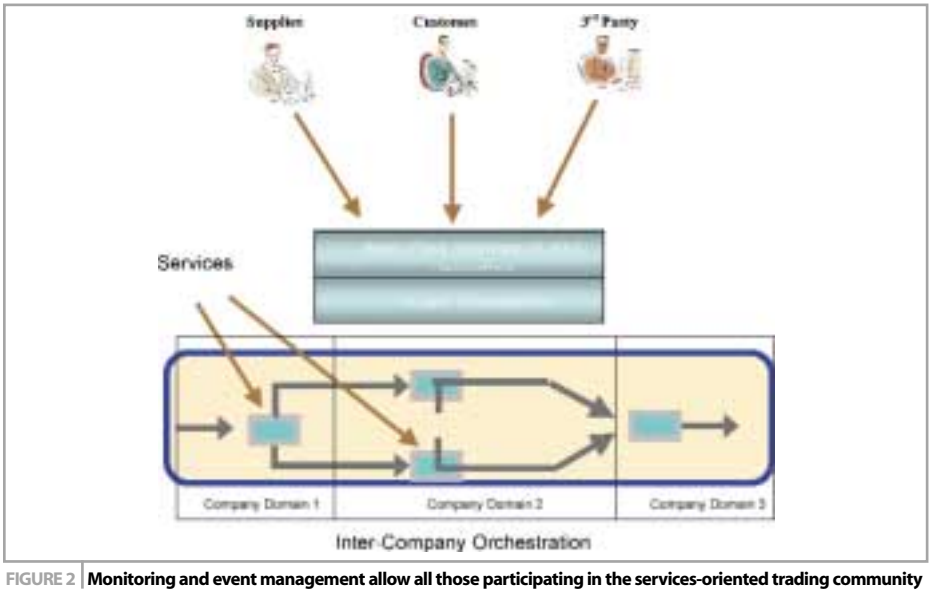


FIGURE 2 **Monitoring and event management allow all those participating in the services-oriented trading community to monitor processes/services as they happen, and setup automatic mechanisms to handle events. For instance, if supplier is more than 4 days late, then use another supplier.**

allows us to change the internal representation and/or implementation without changing the abstract behavior, and allows people to use the abstraction in terms of intercompany SOA without needing to understand the internal implementation.

Monitoring and event management encompass the ability to analyze all aspects of the business and enterprise or trading community to determine the current state of the process in real time, and adjust those processes as needed and in real time. Optimization, or the ability to redefine the process at any given time in support of the business and thus make the process more efficient, is an aspect of event management (see Figure 2).

Points of integration allow services to interact with other services, or perhaps an orchestration layer. Services, especially those build for intercompany SOA, need to be designed to interact with other systems. For instance, they should provide more robust discovery of metadata and management of connections. Thus, the service developer needs to architect a service as a point of integration, not simply a point of abstracted functional behavior.

Identity management and security services seem to go without saying when you think of intercompany SOA, due to the naturally occurring exposures, and a detailed discussion is out of the scope of this article. However, it's worth a mention that there are three A's of security which you need to consider: authentication, authorization and audit.

In identity management, especially in the work of inter-company SOA, you must pay a lot of attention to authentication. However, rights and permissions are identity-based attributes and should play a very important role in the identity management.

Finally, we need to manage semantics between any number of systems that have very different application semantics and ontologies. We typically do this through a semantic repository, where the semantics are understood and persisted, and a semantic mapping layer that understands the semantics of the source or target systems and can account for the differences during runtime.

Share and Share Alike

It's not a matter of when intercompany SOA will become a reality. The evolution is under way today, as organizations attempt to extend their SOAs to other partner organizations, and want service-level visibility in return. The key words here are real-time automation and visibility. You can either get good at it now and create a more competitive and responsive business, or play catch-up later.

However, like any other new way of doing old things, you have to consider the architectures and how they mesh together over time. The new dynamic here is that you're dealing with many IT departments and many approaches to building applications The best approach is to understand your own value, and work directly with your partners to ensure that everyone is adjusting their way of thinking. The goal is to build sharable services and processes that automate your business, as well as processes between businesses.

About the Author

Dave Linthicum is the CTO of Grand Central Communications (www.grandcentral.com) and has held key technology management rules with a number of organizations including CTO of both Mercator and SAGA Software. David has authored or co-authored ten books including the ground breaking and best selling *Enterprise Application Integration* released in 1998. His latest book, *Next Generation Application Integration: From Simple Information to Web Services,* was just released.

dlinthicum@grandcentral.com

Who's Master of Your Domain?

Web services security in an unfriendly world

W.C. Fields once said, "The practice of keyhole-listening is usually confined to hotels and boarding houses. It is absolutely indefensible to stoop so low. If the transom is not ajar, remember there are plenty of other rooms in the building." Hackers on the Web can take a similarly cavalier attitude – surfing from site to site until they find one whose "transoms are ajar." The question for you is whether yours are among them.

WRITTEN BY
RICKLAND HOLLAR

Information is an increasingly valuable asset in most organizations. Information security is about protecting that asset. Computer security deals with protecting data on corporate systems, network security with protecting data in transit across corporate networks. William Stallings introduced the term *internetwork security*, combining elements of both computer and network security, to cover tools and techniques for protecting data in today's internetworked world.

Internetwork security is most relevant to Web services because it brings together the mechanisms for protecting clients and servers operating across interconnected networks such as the Internet. Internetwork security involves recognizing and countering attacks that may occur as part of transmitting information between parties across such open networks. The Web services recommendations and standards I cover in this article address three specific types of internetwork attacks:

- ***Fabrication:*** Unauthorized parties masquerading their identities or inserting unauthorized messages.
- ***Interception:*** Unauthorized parties "listening in" on conversations.
- ***Modification:*** Unauthorized parties changing message content.

Web services counter these attacks by weaving together security services specifically designed to detect, prevent, or counter elements of the attacks to create "virtual trust domains" like that shown in Figure 1. The National Institute of Standards and Technology (NIST) defines a trust domain as "a group that operates under the supervision of a Domain Policy Management Authority, uses consistent policies, and has similar Certification Practice Statements." Domain members enter into mutual trust relationships, either directly or through third parties, as part of any sensitive information exchange. Security policies governing these relationships may be broad, saying that parties can openly "talk" to anyone, or extremely narrow, saying parties must verify each and every aspect of every conversation. The sensitivity of the information, the number of unknown and untrusted parties in the network, and the price of compromise must all enter into the decision about how much security is enough.

When you make services available on the Internet, your security focus must shift from protecting systems and private networks to protecting individual machines, users, and applications (or services). The lack of a single policy management authority and agreed-upon certification practice statements makes creating trust domains in this environment extremely challenging. The Internet is a collection of cooperating internetworks, each potentially with its own security policies and certification authority, therefore Web services crossing domain boundaries must create "virtual trust domains" that are inclusive (contain all the parties to the transaction: the Web services client and server, potentially the UDDI service, and any third-party brokers) and bridge any policy differences. Web services integrate four basic security services to create such virtual trust domains:

- ***Authentication:*** Verifying parties are in fact who they say they are.
- ***Authorization:*** Ensuring parties only have permissions, accesses, and authorities they warrant.

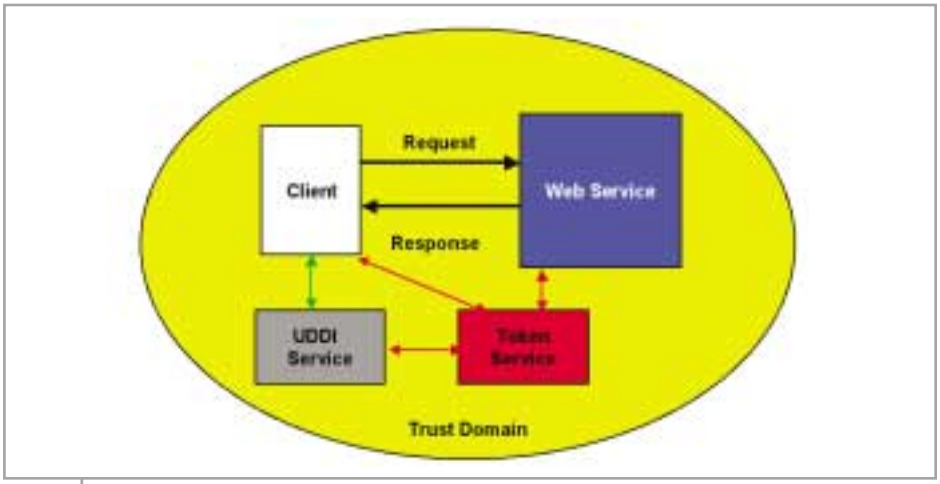


FIGURE 1 | **Virtual trust domain**

June 28–
July 1, 2004

Moscone Center
San Francisco, CA

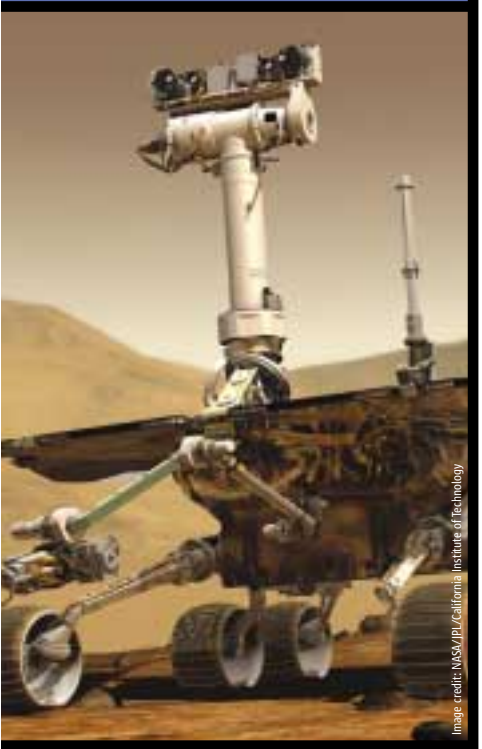


Image credit: NASA/JPL/California Institute of Technology

Benefit from four days of in-depth training and networking with the foremost leaders and creators of Java™ technology in:

- Enterprise
- Mobility
- Web Services
- Desktop

Java™

Go Anywhere You Want with Java™ Technology

It all starts at the JavaOne℠ conference.

Java™ technology is everywhere, improving the digital experience for everyone. It all starts at the JavaOne℠ conference, your source for cutting-edge knowledge and proven solutions. You'll learn how to tap into the full power of the Java platform: platform independence; compatibility with and between editions; open toolsets and technologies supported by a global developer community.

The JavaOne conference offers hundreds of in-depth technical sessions in:

Topic 1—The Foundations: Core J2SE™ Technologies
Topic 2—Core Enterprise Technologies
Topic 3—Java™ Technology on the Desktop
Topic 4—Java Technology for the Web
Topic 5—Java Technology for Mobility
Topic 6—Dissecting the Implementation: Solutions
Topic 7—Intriguing and Unexpected: "New and Cool"

JavaOne℠

Sun's 2004 Worldwide Java Developer Conference℠

Save $100! Register by June 27, 2004, and receive $100 off the full Conference package. Priority code: ADBEZKND

Register at java.sun.com/javaone/sf

Sponsored by Sun microsystems

Produced by MediaLive INTERNATIONAL

- ***Confidentiality:*** Ensuring intermediaries cannot see data exchanged between parties.
- ***Integrity:*** Verifying intermediaries did not modify data in transit between parties.

Figure 2 shows how three key Web services standards (WS-Policy, WS-Security, and WS-Trust), in conjunction with a number of existing and derivative standards, provide the underlying framework for integrating these security services together. While I'll primarily talk about using security services to create relationships between Web services clients and servers, you can use the same techniques for establishing trust relationships among any of the actors.

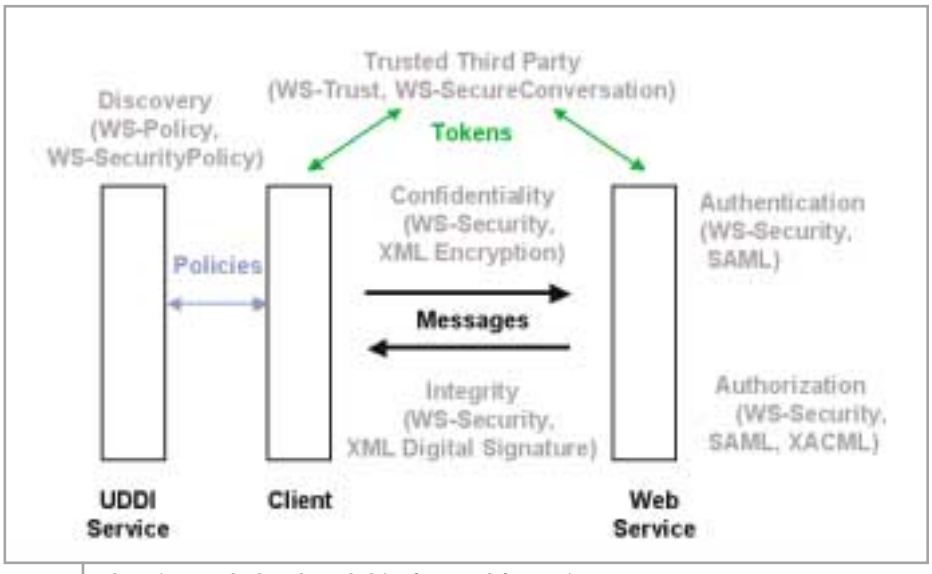


FIGURE 2 | **Web services standards as the underlying framework for security**

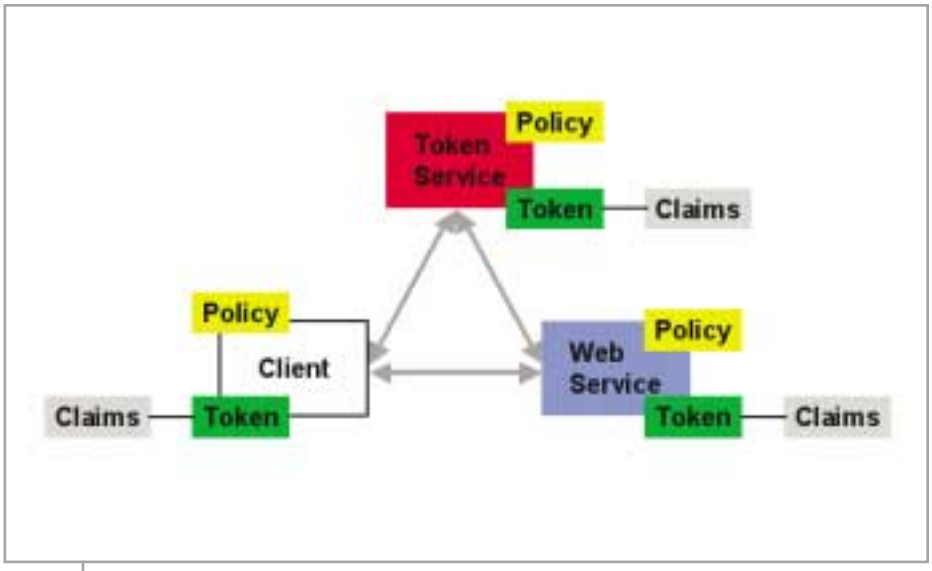


FIGURE 3 | **WS-Trust token model**

WS-Policy

Security begins with a set of security policies. Security policies are rules marrying security tokens, which are representations of sets of security information or claims, to messages (the lingua franca for Web services exchanges). In an exchange, the client is responsible for including the right tokens in the message; the Web services server is responsible for using those tokens to enforce the security policy in place. Enforcement involves verifying the client's claims against the policy in determining whether to honor or deny the request. Verification may be in-band, part of the message flow, or out-of-band, based on a pre-set decision.

WS-Policy and WS-SecurityPolicy make policies discoverable by providing grammars for describing policies as sets of conditions on actions (policy assertions) and for identifying specific elements of a security policy such as:

- The security tokens a Web service will accept (user-id/password, Kerberos tickets, X.509 certificates, Security Assertion Markup Language [SAML] assertions, etc.)
- The message digest algorithms a Web service supports
- The encryption algorithms a Web service supports
- The message elements or tags that must remain unencrypted

Using these grammars, servers can unambiguously publish their security policies and clients can discover them at runtime, if necessary. A server can publish its policies through either its UDDI entries or some other mechanism known and accessible to its clients.

WS-Security

The policy tells clients "which" security tokens to use; WS-Security tells them "how" to encode those tokens as XML. The standard includes elements for representing user IDs, passwords, Kerberos tickets, and X.509 certificates as part of the SOAP message header, and builds on the XML Encryption and XML Signature standards for providing message confidentiality and integrity. WS-Security also supports SAML because SAML assertions are simply wrappable XML.

WS-Trust

Some security tokens require third-party brokers. WS-Trust provides the request/response model, and syntax for requesting, obtaining, and verifying security tokens from such brokers. WS-SecureConversation extends WS-Trust for establishing security contexts for exchanging multiple messages by defining three types of contexts: those created by a token service, those created by one of the communicating parties, and those created through negotiation. Once parties establish a security context they can exchange multiple messages without having to reestablish claims.

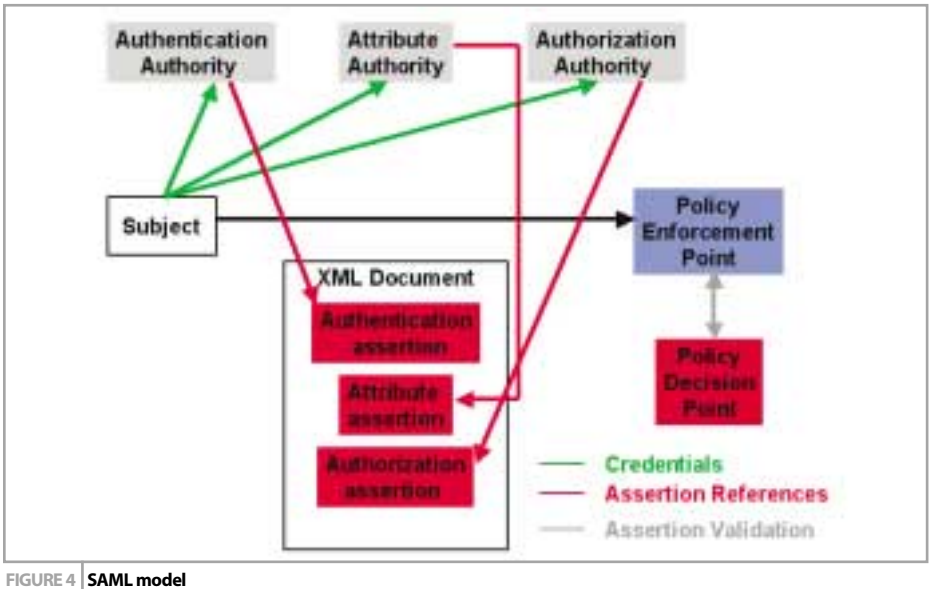


FIGURE 4 | **SAML model**

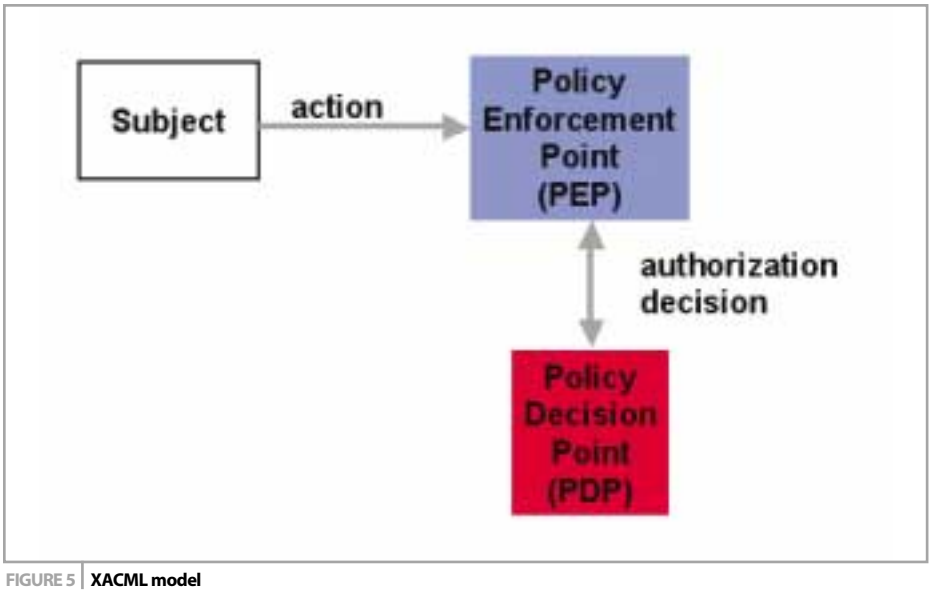


FIGURE 5 | **XACML model**

Authentication

The authentication service validates identities as part of creating trust relationships. In direct trust relationships, the client presents unendorsed claims, such as a user ID and password. The server trusts these claims solely on the basis of the client's knowledge or possession of the necessary token. In brokered trust relationships, the server accepts claims that a trusted third party either endorses or presents on the client's behalf. The third-party is a security token service, such as a Certificate Authority (CA), Kerberos Key Distribution Center (KDC), or SAML service. Most consider third-party tokens more secure than simple user ID/password schemes because of the increased sophistication of the attack necessary to compromise them.

WS-Security does not specify a particular authentication method, so you have several options available for implementing authentication services. One option is to build authentication directly around traditional security tokens such as user IDs and passwords, Kerberos tickets, or X.509 certificates. WS-Trust describes how to build trust relationships using these tokens following the model shown in Figure 3. IBM and Microsoft first presented this as a general model in their Security Roadmap subsuming more specific models such as identity-based security, access control lists, and capabilities-based security. In this model, if a component does not have the tokens necessary to substantiate claims, it contacts a Token Service it trusts to obtain those tokens.

SAML offers an option combining one or more external authorities and assertions. It defines three types of assertions: authentication for proving user identity, attribute for specifying details about the user, and authorization for describing what the user can do. Figure 4 illustrates the SAML model. In this model, the subject (client) sends its credentials to up to three different authorities to obtain assertion references it can include in the message it sends to the Policy Enforcement Point (PEP) (Web services server) that controls the resources. The PEP uses the references to request the actual assertions (authentication decision) from the issuing authority or Policy Decision Point (PDP).

Regardless of which authentication method you use, or how parties establish context, the parties must agree for it to succeed. From the authentication service's standpoint, it's all about who you trust. A brokered token from an untrusted broker is no better than a direct token from an untrusted client.

Authorization

The authorization, or access control, service maps the user's identity against their permissions for the resources being used or requested. It assumes authentication. Permissions are actions the user can perform. Primitive permissions include atomic activities such as create, read, update, and delete; advanced permissions include more abstract activities such as debit, credit, buy, and sell. In Web services applications, permissions often map to Web services transactions.

The two most prevalent approaches to access control on the Web are:

- ***User-based access control (UBAC):*** User's identity determines their accesses.
- ***Role-based access control (RBAC):*** Roles bring together collections of users and permissions.

The Extensible Access Control Markup Language (XACML) is the rapidly emerging authorization standard. XACML provides a grammar for expressing access control poli-

cies as access matrices of authorization rules stated as subjects, actions, objects, and conditions. It includes elements for expressing access control information within XML documents and a request/response language for determining specific access permissions. Figure 5 illustrates the XACML model where the subject (client) wants to take an action (request) on a resource belonging to a PEP (a Web services server). The PEP uses XACML to request the access authorization decision from a PDP that makes a determination based on XACML access policies and returns the decision to the PEP.

SAML and XACML are complementary standards sharing basic concepts and definitions, hence I foresee them becoming the predominant mechanisms once implementing products are widely available.

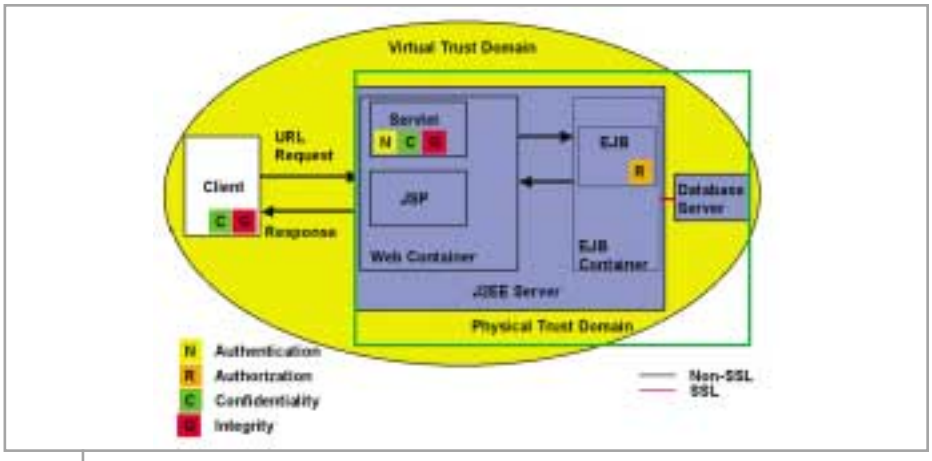


FIGURE 6 **J2EE rendition of this standard**

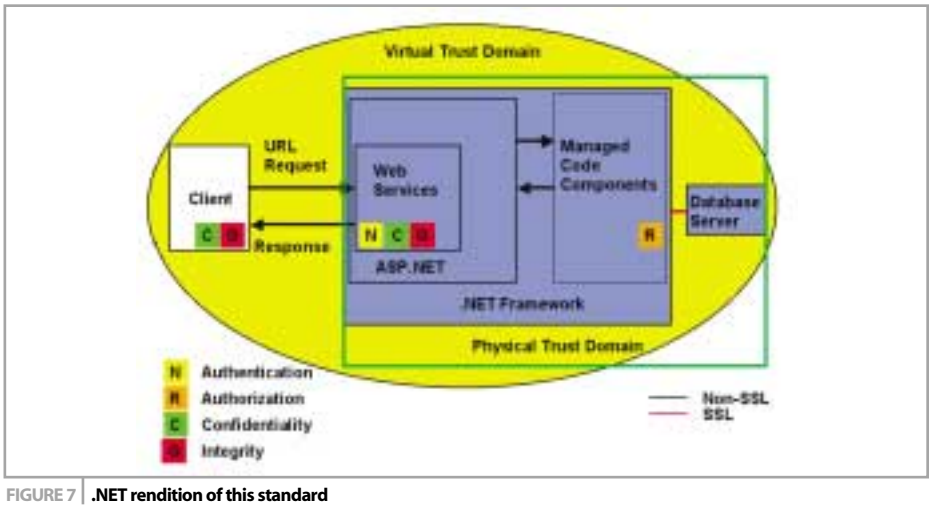


FIGURE 7 **.NET rendition of this standard**

Confidentiality

The confidentiality service generally uses an encryption function to ensure privacy. Encryption changes the data being protected from plain text into a cipher text that only the sender and receiver can understand. Encryption programs apply an algorithm in combination with an encryption key in order to make that transformation. The key may be a shared secret (a symmetric key that both parties share) or different halves of a public, private key pair (asymmetric algorithms use one of the keys to encrypt the data, the other to decrypt it). The strength of the encryption, how difficult it is to break, is a function of the algorithm and the key length. Encryption ensures confidentiality because while someone may still be able to steal the traffic, they are stealing meaningless bits unless they also know the algorithm and a key.

WS-Security leverages XML Encryption for confidentiality. XML Encryption provides for encrypting individual elements (including the tags), only their content, and complete documents. This flexibility enables end-to-end encryption in a Web services environment while supporting some processing by intermediaries – an advantage over point-to-point solutions such as SSL.

XML Encryption defines elements for carrying cipher text and information about underlying encryption algorithms and encryption keys. XML Encryption supports: Data Encryption Standard (DES) and Advanced Encryption Standard (AES) for block encryption and key wrapping, Rivest-Shamir-Adelman (RSA) for key transport, Diffie-Hellman for key agreement, and Secure Hash Algorithm (SHA) and RIPEMD for message digest. While XML Encryption does not include stream encryption algorithms, it is extensible given that parties agree on the algorithm and its usage. XML Encryption also supports super encryption, i.e., encrypting data multiple times to strengthen the encryption.

Integrity

The integrity service generally uses a digital signature capability to ensure nonrepudiation. WS-Security leverages XML Signature in this area. XML Signature provides for signing both individual elements and entire documents. The standard defines elements for describing: what is signed, the signature itself, the canonicalization method used to deal with different data streams within elements, any transformations applied prior to generating the message digest, and the digest method used to generate the signature.

Digital signature algorithms create signatures in a four-step process. The first step converts the document (or element) into a standard form through the application of a canonicalization, or standard serialization, algorithm. The Exclusive XML Canonicalization recommendation specifies Canonical XML (XML-C14N) as the standard serialization technique for Web services. The second step is to apply one or more transformations to the canonicalized content. The document (or element) in standard form becomes the input to the first transformation. The output of each transformation becomes the input to the subsequent transformation procedure. The output from the final transformation procedure becomes the input to the digest function. Transformation operations include canonicalization, encoding/decoding, compression/inflation, XSLT, XPath, XML Schema validation, and XInclude. The third step is to process the content through a secure hash function to create a message digest (message digest algorithms, such as

Message Digest 5 [MD5] and SHA, use a hash function to create a unique code that becomes a digital fingerprint for the document). The final step encrypts the message digest, using the client's private encryption key, to create a digital signature. The digital signature is then appended to the document. The recipient recalculates the message digest and compares it to the sender's to ensure the document was not changed. The recipient uses the sender's public encryption key to verify the digital signature.

Communicating the order of the encryption and signing operations is an important consideration. WS-Security modifies XML Signature to add a decryption transformation. This allows the sender to include decryption order as part of the transformation information for the message. The recipient simply reverses the order to decrypt the correct message parts before recomputing the message digest.

Virtual Trust Domains

Figures 6 and 7 illustrate how these concepts come together in J2EE and .NET environments. In both cases, a client requests services from a Web services server hosted within a trust domain formed by one or more systems in physical trust relationships. The Web services server establishes a dynamic trust relationship with the client thereby creating a virtual trust domain. The client and Web services server then exchange whatever number of messages is relevant to the transaction including the necessary security tokens and signing and encrypting message parts, as appropriate. The security policies in place, in combination with the strength of the security tokens, encryption, and digest algorithms in use, establish the strength of the trust relationship.

Conclusion

The Web services and XML standards I've discussed provide the framework necessary to create security strategies in these environments. In creating your security strategy, you must also consider computer and network security of your systems and the vulnerabilities others' clients may introduce. You may want to leverage additional mechanisms, such as SSL and IPSec, where appropriate to bolster the techniques I've discussed. One size does not fit all. The important thing is that you take a comprehensive approach to ensure you are doing everything necessary in your Web services to protect sensitive information and resources they control. Remember, your services are only as secure as their weakest link. Are you master of your domain, or are your transoms ajar? Don't wait until after an attack to find out.

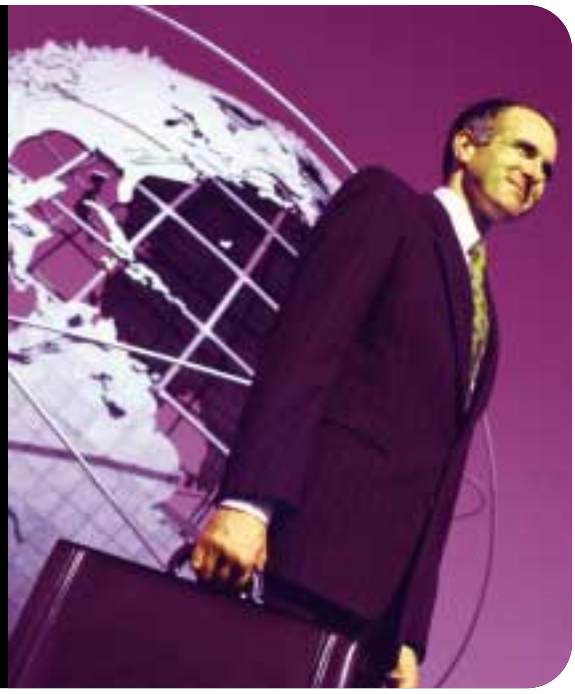
About the Author

Rickland Hollar is a senior applications architect with the Central Intelligence Agency with over 30 years of experience in the industry. The views expressed in this article are his own and not necessarily those of the Agency. Prior to joining the CIA, he was president of a Virginia-based software development firm.

rick_hollar@yahoo.com

POA: Building SOA from the Ground Up

Moving process to the forefront



As technologists today, we face a uniquely challenging paradox. On the one hand, Web services have created renewed excitement for service-oriented architectures (SOA) as the answer to common integration problems; on the other hand, today's application platforms (J2EE & .NET), used to implement SOA applications, are often deemed overly complex, and a barrier to widespread SOA adoption. While the primary goal of the industry is to make the concepts and benefits of SOA mainstream, we must recognize that SOA itself must evolve to facilitate such widespread adoption. Fundamentally, we must move from focusing on the integration aspects of SOA (the "band-aid") to focusing on how to build applications based on the principles of SOA (the solution); that is, applications that are inherently interoperable. In this article we discuss how the convergence of several key technologies is essential to taking SOA mainstream, and how process-oriented architectures (POA) are evolving to meet this need by providing a more detailed version of SOA that derives its fundamental purpose from a process-oriented view of application development.

SOA Defined

SOA describes an approach to distributed computing based upon three primary entities: a provider, a consumer, and a registry; and seeks to maximize certain desirable characteristics for networked resources called services. Services are the single construct used to represent a given system and are expected to be loosely coupled, modular, interoperable, compose-able, coarse-grained, network-addressable, and discoverable.

WRITTEN BY
SHONE SADLER

While the importance of these concepts and characteristics cannot be overstated, they are by no means new. RPC, CORBA, DCOM, and RMI are all examples of past SOA-based implementations that, though successful to some degree, were not and are not adequate for making SOA a prevailing practice. Very few IT shops had (or could maintain) the resources and skill sets required to develop enterprise applications using such technologies. The technologies were simply too complex for the majority. Or rather, resource constraints often dictated a more rapid (UI-centric) approach.

Learning from the Past

While Web services has indeed generated much excitement and debate around SOA, can it succeed where its predecessors did not? Some would argue that Web services were an inferior technology to alternatives such as CORBA or DCOM because those technologies already provided many of the enterprise-level features that are currently being developed for Web services. Yet much of that debate has subsided, as Web service specifications have been created to address those concerns. Furthermore, Web services has the distinct advantage of support by all of the dominant vendors in the software industry to an unprecedented extent.

Still, we are plagued with the problems of the past, in that Web services and SOA alone do not overcome the complexities and challenges of developing applications on today's platforms. Moreover, the industry is still focused on "band-aiding" stovepipe applications, rather than providing the solution

How to Make Money with Web Services

By Bob Brauer
bob.brauer@strikeiron.com

A new way to work with Web services - build them and they will come?

The benefits of Web services are well understood – lower integration costs, maximum reusability, faster deployment, more automation, easier to work with new partners, and so on. You can build them but that does not necessarily mean that people will use them or that you can make money on them. Why? And more importantly, how?

What drives Web services utilization and return on investment?

Here are some key points to consider:

- Accessibility and ease-of-use
- Billing and accounting
- Subscriptions and trials
- Availability, reliability, and security
- Promotions and pricing

What tools and services do you need to commercialize your Web services?

- Tools to improve understanding and use.
- Commerce capabilities to manage subscriptions, accounting, billing, payments, account management, etc.
- Ability to manage free trials and convert trial users to subscribers.
- Service levels that ensure availability, reliability, and security.
- Knowledge about acceptable pricing structures based on value of the data, process, accessibility, and performance.
- A way to deliver and promote them to the appropriate target audience.

Creating a new revenue channel

Your Web services provide value and you need to be reimbursed for that value. However, before that can happen, you need a distribution channel with the infrastructure to publish and sell your Web services. This channel must take care of issues such as delivery, account set up, billing and collections, marketing, and customer support, to free you of having to make that investment.

The answer is Premium Web Services

Premium Web Services provided by StrikeIron and available through the StrikeIron Web Services Business Network™ (WSBizNet™) are the answer.

Whether you are a Web service provider wanting to commercialize your Web services or a company wanting to find a way to manage your Web services without the infrastructure investment, StrikeIron will do the work for you, and you make money in the process.

StrikeIron Premium Web Services are subscription-based Web services that deliver "live" data and functionality over the Internet via XML using a dynamic SOAP interface. Premium Web Services are supported by the commercial services and easier-to-use tools of the StrikeIron WSBizNet to remove barriers to purchase.

Premium Web Services benefits

- An alternate sales channel to reach more customers for minimal cost.
- Web services hosting and delivery services in a secure environment.
- Pricing, billing, collections, and support.
- Reporting for easier maintenance and tracking.

The StrikeIron Web Services Business Network

The StrikeIron WSBizNet is a subscription-based online network of services and tools for easier and faster utilization of Web services by business users and developers.

The WSBizNet allows anyone, with minimal experience, to quickly find, understand, and utilize Web services for accessing information, integrating with existing applications, and assembling new applications.

StrikeIron WSBizNet benefits

- Service levels to meet your business requirements.
- Integrated online StrikeIron Web Services Analyzer for faster understanding of Web services.
- Integrated online Knowledge Base makes it easier to understand and work with your Web services.
- Prime positioning in the WSBizNet.
- Increased visibility in one of the largest Web services business directories.

Delivering Web Services to the World!

For Free Trials and more about how to make money with the StrikeIron Web Services Business Network and Premium Web Services, please visit our Web site at www.strikeiron.com.

StrikeIron, Inc.
+1.919.405.7010
sales@strikeiron.com
www.strikeiron.com

for not building them initially. Just having a standard distributed computing technology does not make it substantively less complex for developers to design, build, deploy, and maintain enterprise applications that are inherently interoperable. We still need to implement the business processes behind the services. Today, that means writing code on either the J2EE or .NET platform.

Convergence: Process, Forms, and Data

Another distinct advantage that Web services have is timing…meaning that we are now at a point in time where several key technologies– including Web services – are converging, with XML serving as the common basis. Integrated into a single platform these combined technologies can significantly drive the adoption of SOA concepts into mainstream application development.

Back-End Service Orchestration (Process)

The most notable of these technologies is business process management (BPM), which is concerned with the design, deployment, execution, maintenance, and monitoring of business processes. Business Process Management Systems (BPMS) are design-driven environments for adding functionality to the organization. They execute business processes that consist of one or more activities which may span across multiple systems, with the purpose of achieving a particular goal for the organization. Web services are a natural fit with BPM. Processes designed in a BPMS can be automatically published to a registry and made available as a Web service, making the BPMS a Web service container. Alternatively, processes designed in a BPMS can consume coarse-grained Web services as a step in the process.

Several XML standards are being worked on concurrently that focus on the orchestration of Web services. The front runner is the Business Process Execution Language (BPEL), submitted jointly by Microsoft and IBM to the OASIS standards committee. It is said to have combined the best of both Microsoft's and IBM's prior technologies (XLANG and WSFL, respectively).

It is important to note that BPM has not traditionally been used within the scope of application development. Instead, BPM technology has focused on a subset of business processes – "integration" processes.

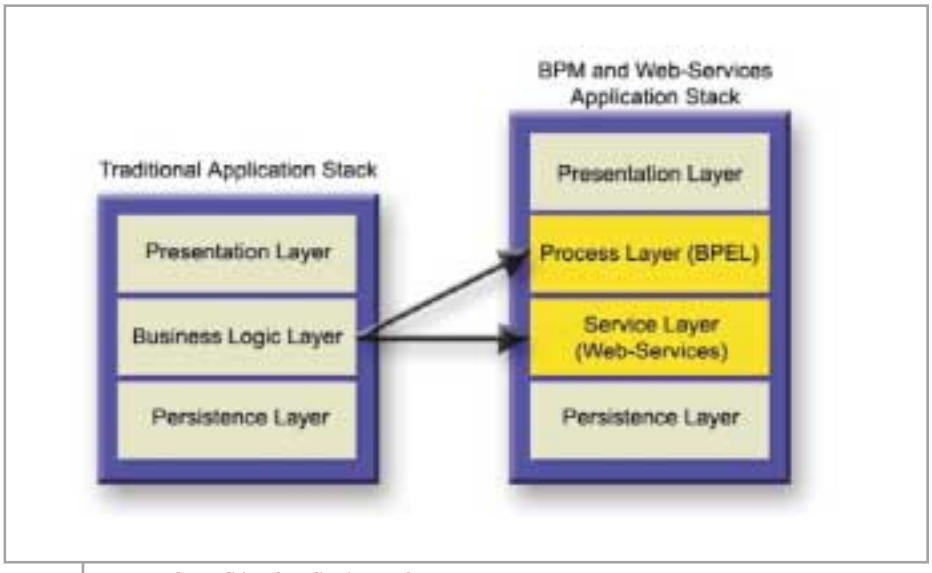


FIGURE 1 **Impact on the traditional application stack**

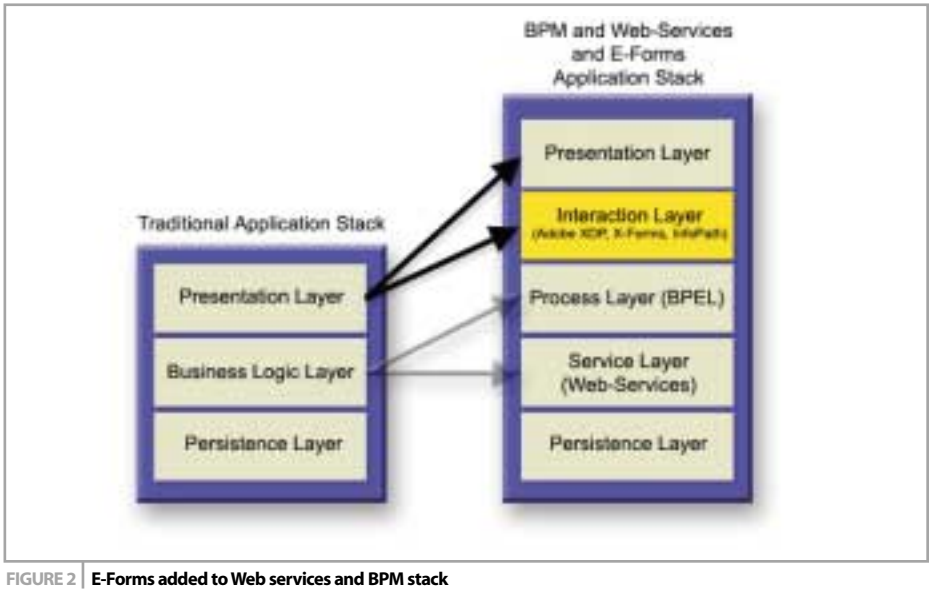


FIGURE 2 **E-Forms added to Web services and BPM stack**

The core processes that exist at the application level have been left behind, hard-wired into stovepipe applications.

Figure 1 shows how the combination of BPM and Web services can affect the traditional application stack.

The business logic layer as we know it is split into two layers: the process layer and the service layer. The latter still manages the majority of business logic and may contain both coarse-grained Web services that initiate long-lived processes, and fine-grained services that execute within the container. The process layer, however, provides a new layer that contains the logic used to orchestrate services in the lower layer. This logic was previously scattered within both the business logic and presentation layers.

Front-End Information Gathering (Forms)

The Web services and BPM combination provides a perfect solution for back-end service orchestration, but falls short in a key area of application development – gathering information from front-end users. Electronic forms (E-Forms) technology fills that gap by treating user interactions as componentized, reusable assets that collect information from end users to be validated against business rules, utilized for routing and controlling the flow of work, parameterized and marshaled into service calls, and

then rendered in yet another form. E-Forms typically offer:

- Design-driven environment
- Platform and device independence
- Offline capabilities
- Richer data structures than name-value pairs offered in HTML
- Security features, including encryption and digital signatures
- A predictable on-screen and printable appearance
- Component-oriented capabilities

E-Forms will be essential in driving widespread adoption of SOA in application development for the following reasons:

- ***Interoperability:*** Development tools have historically relied upon various presentation technologies, whether a host programming language such as Visual Basic, or a tag-based framework such as JavaServer Pages, to create forms for user interactions. For developers this reliance on a particular presentation technology introduced the problem of constructing multiple versions of a form – one for each presentation technology/layer used in a single application. Creating different versions of a form for different technologies and devices that in the end interact with the same underlying business processes engenders extra development and expensive maintenance costs related to cross-platform consistency, and complicates the problems of configuration management.
- ***Orchestration:*** Development tools currently lack an effective way of managing (other than hard-wiring) the relationship between forms and back-end services. Similar to Web services, forms should be treated as componentized assets that can be reused across various contexts. A BPMS provides an effective solution for utilizing these assets to model complex flows of interactions that involve one or many users, either sequentially and/or in parallel. Leveraging both Web services and E-Form technologies, a BPMS could weave together various front-end user interactions with back-end services to form one or more business processes.

There are three primary competing XML-based technologies in this area, including Adobe's PDF/XML Data Package (XDP) technologies, the W3C's X-Forms specification, and Microsoft's InfoPath product. Adobe's experience and breadth of technology in this area gives it a clear advantage; various vendors of all sizes are looking towards and implementing X-Forms–based solutions, while Microsoft can definitely be expected to integrate its InfoPath product deeper into its office and back-end solutions.

In Figure 2 we add E-Forms to the Web services and BPM mix to further extend the application stack. The key distinction here is that front-end user interactions used to gather information are separated out of the presentation layer as componentized assets in the interaction layer to be orchestrated by the process layer.

Business Objects (Data)

If you are developing enterprise applications, you inevitably have to store the data related to the objects that your underlying processes are utilizing and/or affecting. These objects, often termed Business Objects, are typically complex data structures used to store data that is needed by the process, part of the process, or used to notify management about the status of a process. The design and management of these objects was, in the past, left to the sole discretion of the database administrator. However, as database technology has advanced in both speed and ease of maintainability the lines between database design and application development have blurred. This has largely been driven by the need to develop and adapt applications more rapidly, but also due to the more complex data structures and constraints needed in today's environment.

Two technologies are of keen interest in this area. The first, business object modeling tools, has evolved from pure relational modeling tools used by database administrators into design-driven environments supporting newer object-oriented methodologies used in application development. The second, object/relational mapping frameworks, provides a runtime environment for managing and storing objects used in applications by mapping their properties and relationships back to relational (and sometimes nonrelational) data stores. Together, these technologies can be used to facilitate both the design and runtime aspects needed to support the growing complexity of data structures and constraints needed in today's enterprise applications.

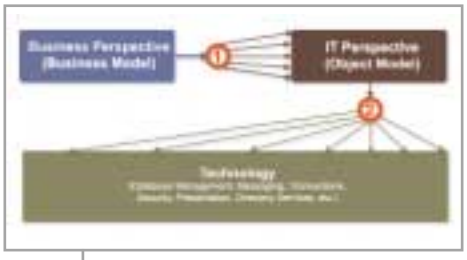


FIGURE 3 **Primary points of translation in enterprise application development**

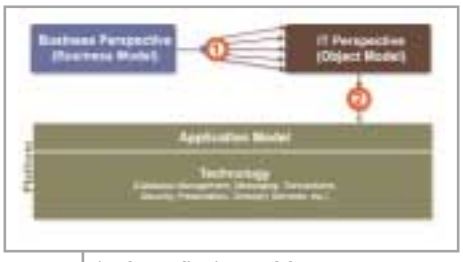


FIGURE 4 **Simpler application model**

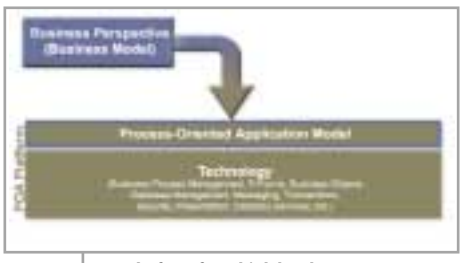


FIGURE 5 **POA platform from high level**

Several XML standards, including XML Schema, XPath, and XQuery will play an essential role with respect to business objects. While XML Schema is used for defining complex data structures, XPath and XQuery provide a standard means of querying against those data structures.

POA: The Next Step for SOA

Together these technologies are representative of the three dimensions of Model-View-Controller (MVC), a popular software design pattern. MVC has typically been limited solely to the presentation aspect of applications, but when applied within the context of these technologies it produces a powerful design-driven platform for orchestrating both services and user interactions. The end result provides the blueprint for a more detailed version of SOA that derives its fundamental purpose from a process-oriented view of application development. This evolved architecture has been termed process-oriented architecture (POA) and is the key to taking SOA concepts mainstream. POA

> What if developing interoperable, scalable, and maintainable applications did not have to be complex?

differs from SOA in the following respects:

- Focuses on both the "what" and the "how" with respect to services
- Is an application architecture not an integration architecture
- Focuses on the constructs that make up an application, not only the connection points
- Affects our methodologies used for application development, not just the underlying technologies

POA picks up where SOA leaves off by facilitating the rapid application development of complex enterprise applications that inherently exhibit SOA characteristics such as interoperability, modularity, reusability, and scalability. While these have always been desired characteristics of enterprise applications, past application architectures have simply been too complex for them to become a reality. As a result, organizations have purchased hard-wired packaged applications, developed their own stovepipe applications, or simply chosen to do nothing because an implementation may not have been economically feasible.

A fundamental notion of POA is to address this complexity, the root of which stems from translating business requirements into the underlying technology of the day. There are at least two primary points of translation in enterprise application development (see Figure 3).

The first represents an impedance mismatch, a semantic difference in how a business domain expert and an application developer view an application. The second represents the translation between an application developer's view and the underlying (but often disjointed) technologies required to support enterprise applications.

J2EE and .NET have been somewhat successful in making enterprise application development less complex. With these platforms, more people today can participate in building enterprise applications that leverage what were once very specialized technologies. They succeeded in bringing cohesiveness to many of the underlying technologies required for enterprise applications by providing an application model that abstracted away many of the complexities and better correlated the underlying technology with the IT perspective (see Figure 4).

Still, a problem exists in that these platforms are still too complex for mainstream developers. Many applications are being developed with process and business logic scattered throughout the presentation tier and with little forethought into the basic tenets of SOA: interoperability, modularity, reusability, and scalability.

POA implementations build upon and leverage the success achieved by current platforms to make those basic principles of SOA available and feasible for the majority rather than the few by:

- Defining a process-oriented application model that formalizes and integrates the relationships between processes, forms, and data.
- Providing additional run-time and design-time services to support that application model.

Figure 5 depicts a POA platform from a high level.

The important difference when compared to mainstream platforms is the direct correlation between the business perspective and the underlying technology. In reality, the IT perspective is still there, but the two perspectives are merged along with both of their respective techniques and methodologies. The end result is an architecture that enables organizations to build applications in a design-driven environment using business – rather than software – constructs.

Looking at the various levels of convergence occurring in the market today, one can see that movement toward this vision is already underway.

Conclusion

While the convergence of BPM, E-Forms, and business object technologies into a single design-driven platform for developing and executing enterprise applications has many inherent benefits, such as reduced application development time, decreased application TCO, and overall increased business automation, we will limit this discussion to a topic particularly relevant to today's environment – bringing control back within the organization.

It is no secret that organizations across the globe have dramatically stepped up outsourcing the development of enterprise applications. And, based on the inherent complexities in application development we highlighted here and the economic realties they cause, it is no wonder why this phenomenon is occurring. However, outsourcing does not come without a price. As companies become highly dependent and coupled with external development organizations that may not have the same interests or goals in mind, they will lose a significant amount of control and knowledge with respect to the processes that are being outsourced.

Based on the material presented here, we should ask ourselves several questions: What if developing interoperable, scalable, and maintainable applications did not have to be complex? What if you could actually see the "best practices" in your packaged application? What if these "best practices" were no longer hard-wired processes, but instead templates for businesses to customize? What if you could see and manipulate the processes within any application at any point in time? And finally, what if the business value gained by organizations having real-time control of their enterprise processes exceeded the cost of developing and deploying the related applications? These are the questions we should be trying to answer and that form the objectives of POAs. No one debates the importance of SOA; however, it does not begin and end with integration. The beauty of Web services is that it answers the integration question for us, allowing us to instead focus on how we implement our business processes and bringing control back within the organization.

About the Author

Shone Sadler is chief architect for Q-Link Technologies, a leading business process management platform provider. With more than 17 years of experience in emerging technologies, Shone is considered an authority on application architectures and frequently speaks and writes on the topic.

ssadler@qlinktech.com

Lowering the cost and complexity of developing secure, manageable enterprise-class Web services with BEA WebLogic Workshop™ and Confluent for BEA

With Confluent for BEA, Web services developed in BEA WebLogic Workshop™ 8.1 are automatically monitored, managed and secured at development time. Confluent's Control is bundled with WebLogic Workshop 8.1.

Real-time monitoring
No additional coding required, turn on instrumentation for a Web service, view all conversations as they execute, and drill down to views of individual operations and controls

Policy specification and enforcement
Define operational policies including security, logging and monitoring for Web services within BEA WebLogic Workshop 8.1

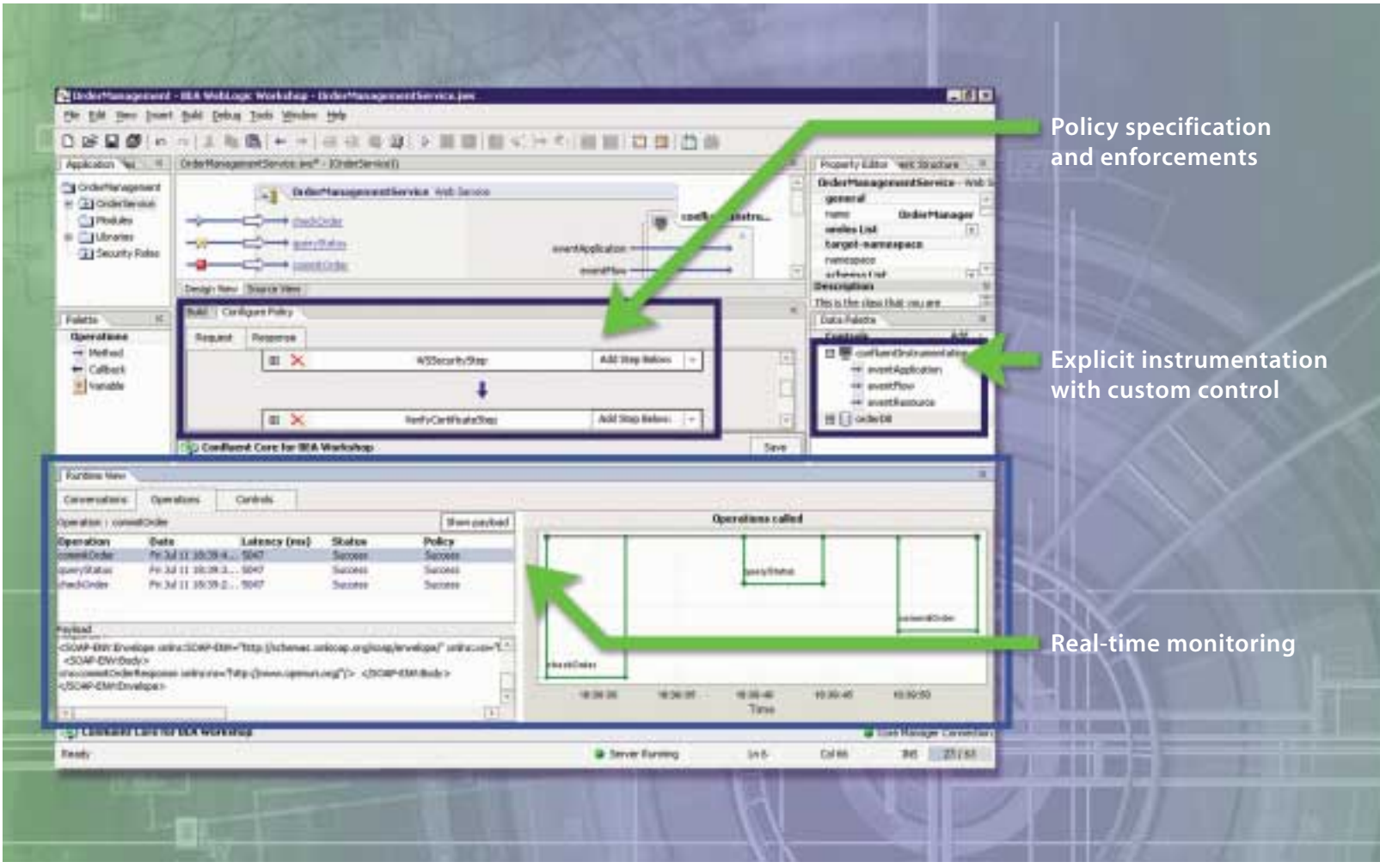
Explicit instrumentation with custom controls
Use Confluent Control to explicitly call operations such as monitoring and security from Web services

BENEFITS:

If you're a Developer—focus on application logic not infrastructure services

If you're an Architect—consistently implement security, change and other operational policies

If you're an Operations Manager—efficiently monitor, manage and evolve increasingly distributed applications



Confluent Software

Download Confluent for BEA at:
http://www.confluentsoftware.com/solutions/download/beacib.html

For more information on additional Confluent Web services management solutions, visit: **http://www.confluentsoftware.com**

SOAPtest 2.5 from Parasoft

Testing made simple – and productive

SOAPtest is a comprehensive testing tool geared to testing Web services. It provides unit, system, and load testing features and support for WSDL, WS-Security, and JMS, including asynchronous messaging. It also supports attachments, access to performance monitors, comparative reporting, and more.

WRITTEN BY
PAUL KAISER

Unit Testing

The goal of unit testing is to exercise each method individually to make sure it is working correctly. SOAPtest leverages WSDL to create a test suite for a Web service. Simply enter the URL to the WSDL resource and SOAPtest does the rest. It creates a new test project with one test suite containing a unit test case per method described in the WSDL.

In addition to the unit test cases, SOAPtest offers you the option to create test cases to validate the WSDL itself. With this, you can check for correctness by verifying that the WSDL conforms to the XML Schema Definition via XML validation, check the validity of URL references it contains, and make sure it conforms to the WS-Interoperability Basic Profile 1.0. In addition, you can take a snapshot of the WSDL and compare it in future test runs to make sure it has not changed. By default, it compares all elements. But if you want, you can choose to ignore certain elements.

Each test case will contain an automatically generated value of the correct type for each input argument described. If that isn't appropriate, you may specify your own either as fixed or parameterized values (see Figure 1). Parameterized values come from one of several forms of data stores. The data store may be one of six types of data sources supported (simple text files, CSV files, databases, Excel spreadsheets, internal tables, aggregate), a built-in tool called an XML Data Bank, or a custom method that you write.

By default, SOAPtest adds an output tool to each test case so you can see the HTTP request and response. There are also a variety of output tools that you can add to a test, in one or more tool chains to produce the output that is most meaningful to you.

Tools

SOAPtest comes with many tools that can process the test case request or response. A test case may have multiple tools applied and tools can be chained to further refine the output. They include tools that check, compare, transform, and extract data from XML documents; write files; handle MIME attachments; and are custom methods written in Java, JavaScript, or Python. Refer to the user guide on the Parasoft Web site for complete details.

Regression Testing

Initially, the test is considered a pass or fail based on the presence of SOAP fault elements in the response. You can change this to have SOAPtest compare the response to one previously captured.

Once the service is responding as expected, you can easily create regression tests using the Create Regression Control feature. With this you can have SOAPtest record the correct responses from the service it is testing. This is particularly helpful for negative tests, where a SOAP fault is the correct response. Once created, the test passes or fails based on a comparison with the recorded response.

System Testing

System testing is a higher-level functional test that exercises a set of features, in a particular sequence, to achieve the desired outcome. They are usually based on use case scenarios. A system test case is best defined as its own test suite that contains the sequence of interactions required as part of the user scenario. This will let you reuse your system test cases as part of load test user profiles.

System tests typically need to pass the output of one interaction as an input to a subsequent interaction. SOAPtest supports this requirement a couple of ways, most notably with the XML Data Bank tool. This tool lets you identify, via XPath expressions, what elements and/or attributes will be available as input to subsequent tests.

Load Testing

Load testing verifies that your system can scale to meet its expected volume demands. Load tests can be pinpoint, targeting individual service methods or, more likely, based on user interaction scenarios. Either way, SOAPtest makes it easy to create load tests.

You can run the load test from the local machine or you can choose to spread the virtual user load around to other machines that have SOAPtest installed.

User profiles let you tailor the load to

PARASOFT

Company Info

Parasoft Corporation
101 E. Huntington Ave.
Monrovia, CA 91016
Phone: (888) 305-0041
E-mail: info@parasoft.com
URL: www.parasoft.com

Platforms/Requirements

2000/XP, Linux, Solaris
At least 256MB RAM (per processor); 512MB recommended, JRE 1.4.1

Price

Starts at $3,995

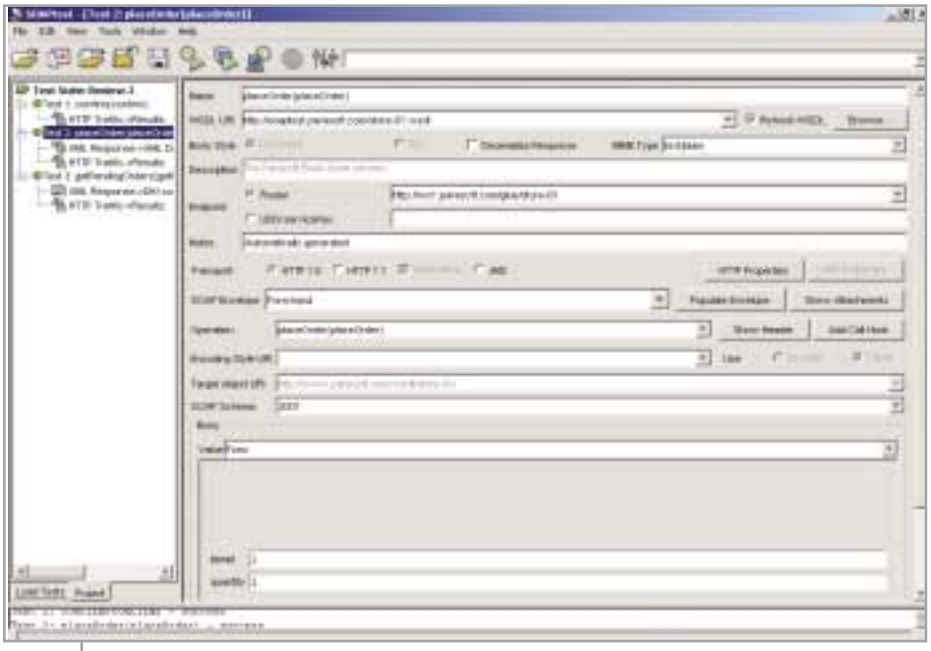


FIGURE 1 **Test project showing the details of the placeOrder test case.**

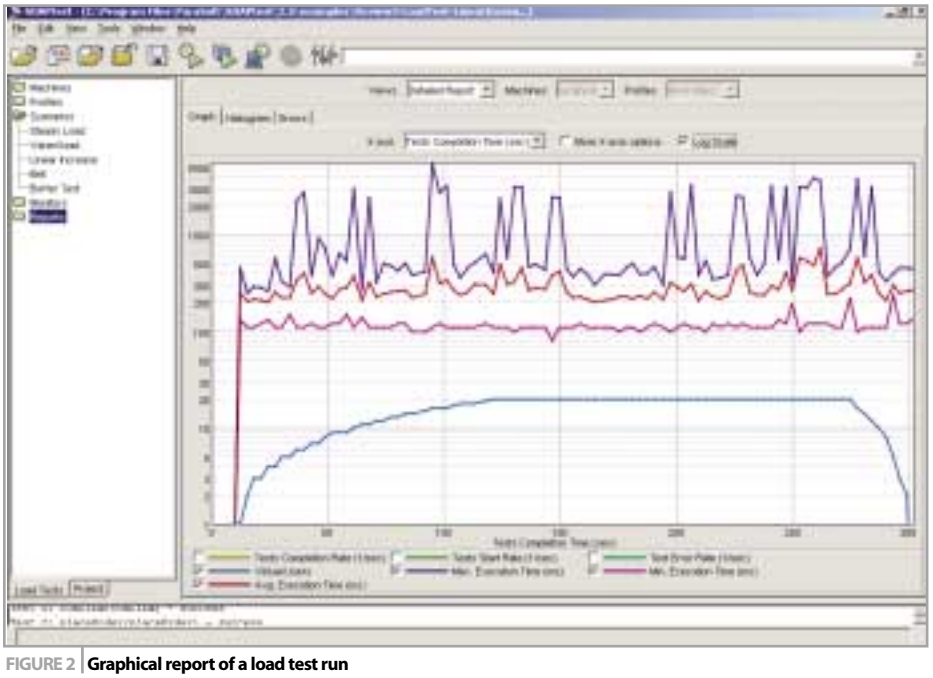


FIGURE 2 **Graphical report of a load test run**

realistic scenarios using one or more test suites. User "think time" can be modeled at the profile, test case, or test suite level.

Load scenarios define the load to produce over the duration of the test. It is specified by a graph of load (in terms of users or hit rate) versus time. You can create any kind of load you want within the constraints of your license and available resources. SOAPtest includes four general curves to start you off. Once created, you can fully customize the load curve by adding or removing points along the curve and move those points around.

You can monitor the performance of the service hosts during the load test using the Monitors feature. This feature lets you capture performance metrics from a Windows or SNMP-enabled host as part of the load test report base. This lets you correlate load with resource usage on the service host(s).

You can capture and report on the results of the load testing both while the test is executing and after (see Figure 2). SOAPtest lets you save the results to a report file separate from the test project. The report information includes the parameters of the load test such as the load scenario, test project/suite, and user profiles.

Other Features

SOAPtest can execute test suites and scripts from the command line, letting you integrate the tests as part of your nightly build process. It can act as a proxy between Web service clients and service hosts. This can provide consistent, centrally managed security across many services. The proxy can check for certain patterns in a request and return SOAP fault if requirements are not met. The proxy can also act as a translator if different hosts implement the same service but require a different message structure.

SOAPtest can also create service implementation stubs from a WSDL and expose the implementation as a Web service using a built-in version of Tomcat. This lets you test Web service clients without requiring access to a real implementation of the service.

Security Support

SOAPtest supports client authentication using certificates or HTTP Basic Authentication. If the Web service requires access over HTTPS, you simply register the server's SSL certificate with SOAPtest. SOAPtest includes features to support WS-Security including configurable SOAP headers, X509, SAML, and username tokens, and support for XML Digital Signature and Encryption.

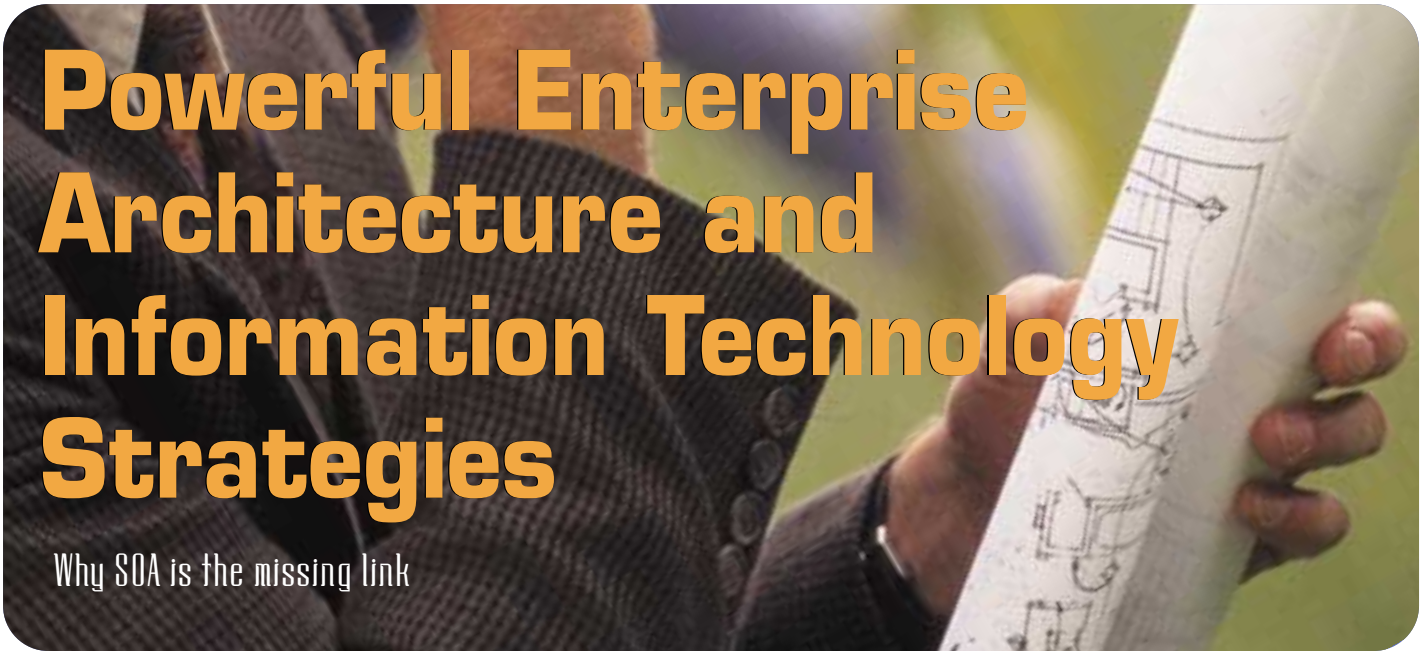
Conclusion

SOAPtest 2.5 makes testing Web services a simple and productive activity. It helps hide the complexities of the Web service standards and technology and lets the tester focus on the task at hand: making sure the service meets all of the functional and technical requirements in scope. If you are serious about what you develop, then automated testing should be important to you. If you develop Web services, then you should check out this tool.

About the Author

Paul Kaiser is a technical architect for InfoTechnologies, Inc., in New Jersey, where he leads the development of Java-based Web applications.

paulkaiser@yahoo.com



Powerful Enterprise Architecture and Information Technology Strategies

Why SOA is the missing link

The promises of services-oriented architecture, enterprise architecture (EA), and information technology (IT) strategies – to change IT from inhibitor to enabler of business flexibility and align IT with business strategies – are remarkably similar. However, for many companies, their EA and IT strategies have not become the catalyst for the intelligence enterprise, the sense-and-respond organization, the agile enterprise, or the adaptive enterprise.

WRITTEN BY
KERRIE HOLLEY

It is the failed promises that tightly couple SOA, EA, and IT strategies. Effective EA and usable IT strategies are necessary to realize the vision of an SOA. And SOA is the thread that connects this trio and provides the perfect opportunity to realize business value through EA and IT strategies. SOA is the future state for EA and IT strategies. SOA is the end game.

The IT industry is littered with definitions of EA, IT strategies, and SOA, and their definitions morph with each passing day. Everyone agrees that creating and implementing EA or IT strategies or SOA is a good thing for an enterprise. However, understanding and reaching agreement on the utility of an EA, IT strategy, and SOA is more useful than agreeing on the definition.

Utility of Enterprise Architecture

Effective EA is a complete model of the enterprise, a master plan and integrating force that couples business goals with IT assets such as services, applications, databases, platforms, and other technologies to guide IT assets in areas of acquisition, development, deployment, and management. And EA must record both a current view of the business as well as the desired future state to provide for future flexibility.

Successful enterprise architecture maps the design of the enterprise within which systems design, technology infrastructure, or business processes should be considered. Well-designed enterprise architecture should have as one of its primary design goals the flexibility to deal with foreseeable change as well as the capacity to cater to enforceable change. This can only be accomplished when the end state envisioned for the enterprise architecture includes service-oriented computing, that is, SOA as the end state.

The 21st-century enterprise needs to do more than merely reflect IT assets on the balance sheet; it must develop structures and frameworks, strategies and processes for leveraging these assets for competitive advantage. Hence, the enterprise architecture must operate as an ordinance for enterprises that aim to maximize flexibility of their IT assets. SOA is an integral aspect of making this a reality for enterprise architectures.

Vital to the success of software development, enterprise architecture is a key communication tool in the enterprise, and is especially valuable in facilitating communication between the IT function and business unit executives. Enterprise architecture must be a joint initiative between business and IT that enhances their alignment.

In *Alice in Wonderland,* Alice asks the Cheshire Cat, "Would you tell me, please, which way I ought to go from here?" "That depends a good deal on where you want to get to," said the Cat. Too many IT architects, IT managers, and business unit executives find themselves asking their enterprise architecture board the same question, and, alas, receive the same response. Enterprise architecture must have greater utility and this is facilitated by making the vision of service-oriented computing, SOA, integral in both the vision and governance aspects of the enterprise architecture.

FREE*CD! ($198.00 VALUE!)

Secrets of the Web services Masters

Every WSJArticle on One CD!

— The Complete Works —

CD is edited by WSJEditor-in-Chief Sean Rhody and organized into 40 chapters containing more than 400 exclusive WSJarticles!

All in an easy-to-navigate HTML format! **BONUS: Full source code included!**

ORDER AT WWW.SYS-CON.COM/FREECD

**PLUS $9.95 SHIPPING AND PROCESSING (U.S. ONLY)*

SYS-CON MEDIA

Only from the World's Leading i-Technology Publisher

Utility of IT Strategies

It has long been obvious that business strategy must drive IT strategy. IT strategies are useful in identifying business and IT requirements currently and in the future by examining the forces shaping the external environment. This entails using information and related technologies to provide advantage, such as increasing information content in products or increasing information content in value networks, or leveraging technologies to create advantage, new products, or enhancements made possible by emerging technologies or sharing knowledge, technology, and capabilities across business units and partners.

> "The 21st-century enterprise needs to do more than merely reflect IT assets on the balance sheet"

IT strategies are useful in implementing IT capabilities that greatly improve the execution of a business strategy. This is made possible by business unit executives sharing their vision with the IT executives.

Second, IT strategies are not a process, but a journey. The French poet, Antoine de Saint-Exupery, wrote in *The Wisdom of the Sands* that "as for the future, your task is not to foresee it, but to enable it." And the discovery of ways to make a difference in the business strategy using IT capabilities is instrumental to the utility of IT strategies.

The Missing Link

SOA must become part of the fabric of EA and IT strategies. SOA is the missing link that, when widely understood and enabled, will make the next era of IT computing possible. And that next era is service-oriented computing, an emerging approach for how software applications are designed, delivered, and consumed where collaborating business applications distributed within and across organizational boundaries emerge. At the core of this emerging paradigm of service-oriented computing is SOA.

By design, service-oriented architecture expresses computing features as a collection of services. A *service* is a capability or a business function that can be dynamically discovered and interfaced. They are described externally from the service implementations and are consumable. These services are part of a layer where services are realized through components. Services must support real-world activity. Many organizations that have implemented Web services solutions have realized that the proliferation of these does not an SOA make. Such point-to-point Web services may not be able to participate in a managed-services environment, be composed, or have business utility to the enterprise as they may not represent good business services. SOA allows enterprises to provide their software and information resources as commercially-available and revenue-generating business services.

SOA also addresses modularity, loose coupling, separation of concerns, and composable and single implementation. It is an approach for business-process orchestration of enterprise-level business services using a distributed model comprised of disparate organizational, customer, supplier, and partner systems. SOA is a programming model complete with tools, technologies and guidelines for creating solutions. It is a natural evolutionary step from the object-oriented (OO), procedural, and data-centric approaches adopted for solution implementation until now. In fact, when creating an SOA system, individual services are typically implemented using one or more of these technologies. SOA in essence, is a way of designing software systems to provide services to either end-user applications or other services.

Conclusion

We are at a tipping point in the IT industry. SOA can become a major catalyst for IT effectiveness or be relegated to the IT ghetto where over-hyped technologies, failed promises, and ineffective IT groups reside.

For SOA to be effective we must raise it beyond the narrow view of simply being a style or architectural pattern. Although useful, it is not enough to meet the promises of service-oriented computing that SOA enables or to create the adaptive enterprise, agile organization or intelligent enterprise. We must look ahead to closing the gaps so that IT is ever-more responsive to relentless financial pressures, unpredictable threats, continuous discontinuities, and competitive Darwinism. SOA is the blueprint for service-oriented computing. And service-oriented computing has the potential to bring about faster time-to-market, new revenue sources, lower integration costs, and reduced complexity for organizations.

IT strategies, EA, and SOA are all about the next stage of IT effectiveness. It is an opportunity for IT departments to finally get it right and truly bring about the fusion of business and IT. The challenge ahead is to integrate them into a comprehensive framework which represents a holistic model of the organization. This trio is necessary to bring about the promises of service oriented computing. Because the enterprise which does not have a useful enterprise architecture is like a machine that coasts downhill at the mercy of any bump in the road; the enterprise which does not discover and execute IT strategies is a stalled machine slowly rusting; and the enterprise which has no plans to move toward SOA is already a relic.

The enterprise architecture is often documented using business, system, and technology models. The system and technology models describe various architecture views such as the data or application architecture. These models and views must be revised to represent both a future state and services portfolio view (i.e. SOA service layer) if the enterprise architectures are to have a measurable effect on how the enterprise develops business solutions. The enterprise architecture must do more than define technology standards or document the business as it is today; it must provide a vision of the end state, one that addresses SOA. Getting to this end state requires that enterprise architecture take a prescriptive approach to how business solutions are created. The vision of SOA can only be realized if the enterprise architecture governance board or architecture working group takes an active role in this prescriptive approach.

The role of the enterprise architecture governance body must expand to acknowledge new services as they mature and to classify them as business services or enabling services for the enterprise. SOA provides boundaries in which services that have value for the enterprise can be realized. The enterprise architecture governance body must take an active role in the cataloging of services and maintaining the integrity of this catalog to prevent redundancy and to make sure published services are available to others.

Whether it is lines of business or the enterprise itself that creates business strategy, having corresponding IT strategies is necessary to optimize the business strategy. Strategy is all about implementation. In his article "Strategy as a Revolution" for the Harvard Business Review, Gary Hamel writes that organizations today fail to distinguish planning from strategizing. "Planning is about programming, not discovery. Planning is for technocrats, not dreamers. Giving planners responsibility for creating strategy is like asking a bricklayer to create Michelangelo's Pieta." SOA and service-oriented computing must be a part of IT strategies; it requires dreamers, business executives, IT executives, and architects who see IT not how it is, but how it ought to be. It requires understanding, as Hamel writes, that this is a quest, a journey. While EA, IT strategies, and SOA are powerless against political infighting and the feudal kingdoms that besiege so many companies; uniting this trio lowers the risk of paralysis that comes into play when no one force dominates an organization. The vision of SOA will take some time to be broadly understood, requires strategic thinking, and will take time before it becomes mainstream.

The promises for companies that transform their IT assets to SOA are enormous. Similarly, the promises for companies that adopt and execute Enterprise Architecture (EA) and IT strategies are equally enormous.

Companies who have effective enterprise architectures have seen the promises of EA come to fruition. Companies with effective IT strategies see alignment of business and IT and the capabilities of IT being applied for business advantage. Similarly, companies adopting SOA today see remarkable improvement in the flexibility of their IT assets. The vision and promises of SOA, EA, and IT strategies are remarkably similar. When this trio is combined to create a holistic view, reengineering the business, creating new revenue sources or reducing costs becomes largely a matter of combining existing components in a different fashion, with the creation of new components being an exception. As an industry, this is the journey on which we embark. This is what SOA and service-oriented computing is all about. The time to start this journey – the time to launch your SOA plans – is now!

About the Author

Kerrie Holley received a bachelor of arts degree in mathematics and a JD in law degree from DePaul University. He is currently a Distinguished Engineer in IBM Global Services and a chief architect in IBM's Application Innovation Services, where he provides technology and thought leadership for the SOA and Web services Center of Excellence. His current focus is in software engineering best practices, end-to-end advanced Web development, adaptive enterprise architecture, conducting architecture reviews, Web services, and service-oriented architecture.

kholley@us.ibm.com

BizTalk Server 2004 in an Investment Bank

The agile service-oriented architecture becomes a reality

This article describes recent work in a leading investment bank using Microsoft's BizTalk Server 2004 (BizTalk) as an integral component of a service-oriented architecture. I'll describe how BizTalk is used to implement lightweight workflow that builds new services from existing services and ties in tactical solutions to enable straight-through processing (STP) of service requests.

WRITTEN BY
DAVID REGAN

I'll look at the architectural vision for a service-oriented architecture and how BizTalk fits into this architecture as one possible implementation technology. I will also outline the progress that has been made to date in creating the base services that will enable this vision to flourish within the bank.

But first, some background on the unique IT requirements of investment banking.

Background

Investment banking has unique information technology requirements. The profits to be made from a competitive advantage often dwarf the IT costs in gaining that advantage. For example, a new derivative product that fits with the mood of the market can generate huge turn over – if it's brought to the market in a timely fashion.

The natural consequence is that the business drives the IT more directly than in most other industries. Consequently, investment banks are often organized with IT functions aligned with business areas, i.e., an IT area supporting foreign exchange (FX) and an IT function supporting fixed income (FI), etc.

The advantage of this organizational structure is obvious – the IT functions respond to the business requirements in a timely and focused fashion. But the disadvantage is equally obvious – the endless development of tactical solutions, the continual patching of out-of-date systems, and the growth of inter-system complexity.

Of course, the banks recognize the drawbacks of their organizations and mitigate these problems by forming cross-functional IT areas with architecture boards at all levels. There is always a tension between the architectural desire for well-structured, loosely coupled strategic solutions and the business demand for functionality now. The architecture that offers most hope for squaring this particular circle is the service-oriented architecture (SOA).

An international investment bank has to respond to different regulatory and best-practice requirements in each country in which it operates. Moreover, an international bank is often organized as a set of separate business entities divided on a regional or country basis. One of the challenges in implementing an SOA is enabling agility in the tailoring/enhancement of services to cope with the differing international requirements and local processes.

This article discusses one of the architecture initiatives being driven by a leading investment bank; namely the "agile" service-oriented architecture. That is, an SOA that has rapid application development and enterprise application integration facilities, and facilitates straight-through processing with a capability to orchestrate disparate sub-services.

An Agile Service-Oriented Architecture

The traditional SOA (traditions are quickly established in Internet time!) replaces conventional layered applications with processes that make use of services that span business areas. The SOA makes services available on a network using widely accepted standards (i.e., XML, WSDL, UDDI, etc.) and focuses on defining contracts for consuming and producing business documents. One common misconception is that SOA is synonymous with Web services, whereas Web services are one, albeit attractive, option for implementing an SOA.

Many industries and investment banking in particular, have a need for a variation on the traditional SOA – the agile SOA. This can be thought of as a set of tools and implementation techniques for building an SOA that can quickly change to meet shifting business requirements and can handle lightweight workflow requirements in a heterogeneous enterprise environment.

Consider the simple example of reference data within a bank. There are myriad sources of reference data that need to be cross referenced. For example, financial instruments can be identified in a variety of ways (e.g,. exchange ticker, ISIN code, Reuters RIC code). Many trading, quotation, or back-office reconciliation applications may well refer to the same financial instrument using different identifiers. These facts give a clear requirement for a reference data service that can map one identifier to another. Such a service would fit well into the scope of a traditional SOA.

Now consider a more complex scenario where a number of business areas want to register new counter party (i.e., trading partner) details with the bank. Again, an obvious candidate for a service in the SOA but this time there are more complex factors to remember:

- Checks of identity, address, etc. with reputable agencies
- Credit checks

- Fulfilling regional money laundering requirements
- Data enrichment with credit ratings

The seemingly simple "Add Counter Party" service has become a complex, long-lived process that needs to correlate responses from sub-services (whether internal or external) and, in addition, may well need to interact with legacy systems.

In the complex and rapidly changing world of investment banking, the requirement to be able to fulfill the straight-through processing (STP) demands of services, such as the "Add Counter Party" service, are commonplace. These processes are sometimes referred to as edge processes as they appear in myriad forms around the core of the bank's business.

> "The traditional SOA... replaces conventional layered applications with processes that make use of services that span business areas"

An implementation technology that can enable the creation of agile SOAs must not only have rapid application development (RAD) aspects, and orchestration technology for long-lived correlated processes, but must also enable enterprise application integration (EAI) to build services from existing infrastructure.

A Framework Implementation

To realize the promise of the agile SOA, the bank has been working on a framework in which to implement this architecture. The objective has been to create a secure, integrated, heterogeneous, and extensible framework that supports the aim of building an agile, service-oriented architecture.

The framework can be thought of as consisting of three components:

- ***Infrastructure:*** Enterprise-level diagnostics, security components, directory services, etc.
- ***Orchestration technology:*** To enable long-running business processes that interact with legacy data sources and applications. This facilitates the incremental delivery of the SOA rather than an all-or-nothing approach – something that would be unacceptable to the lines of business.
- ***Base services:*** E.g., a noteworthy business event logging service

Microsoft's BizTalk Server 2004 is the choice for the orchestration technology for the .NET platform. BizTalk provides excellent facilities for the orchestration of processes that integrate legacy enterprise application and data sources. Moreover, its message-based subscription mechanism enables sets of loosely coupled services to interact in an efficient and scalable way. Furthermore, BizTalk's RAD tools (e.g., the orchestration designer and message mapping tool) allow new services to be added to the SOA in a timely fashion.

The infrastructure of the framework has been designed to provide a stable base for the development of services. Important components of the infrastructure are:

- ***Security:*** The approach has been to leverage the emerging WS standards and to work closely with vendors to make sure that cross-platform Web service technology is a reality. In particular, the bank has developed standards and components that work with WS-Security and WS-Policy to enable end-to-end security of SOAP envelopes using a variety of authentication and encryption techniques: X509, Kerberos, etc. For the .NET platform, we use Microsoft's Web Services Enhancements to implement the Web service security stack and significant effort has been put into integrating this with BizTalk 2004.
- ***Diagnostics:*** On the .NET platform, services make use of Microsoft's Enterprise Instrumentation application block. This provides a set of facilities for tracing requests through distributed systems

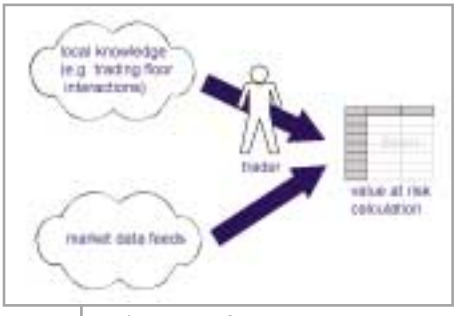


FIGURE 1 | **Logging noteworthy events**

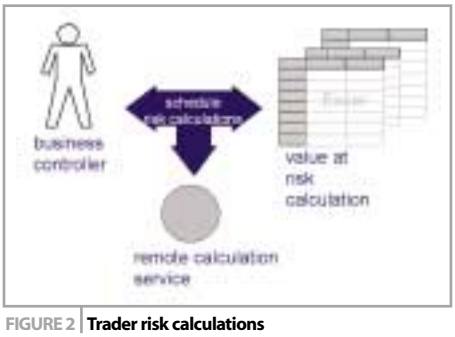


FIGURE 2 | **Trader risk calculations**

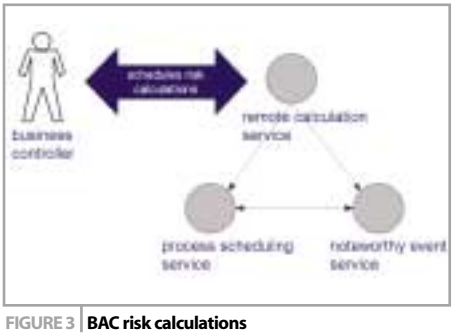


FIGURE 3 | **BAC risk calculations**

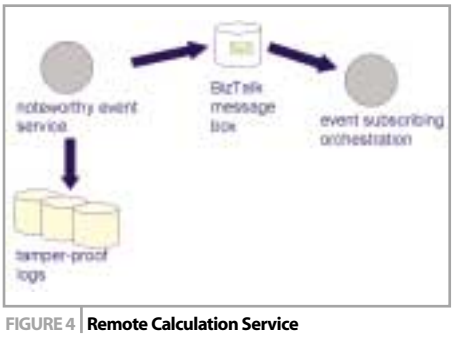


FIGURE 4 | **Remote Calculation Service**

and raising exceptions in a number of formats including Windows Management Instrumentation (WMI) events that can be monitored by Tivoli or similar systems.

- ***UDDI:*** The bank is planning a global directory of the services that make up the SOA.

For the rest of this article I will briefly discuss two base services implemented in the framework: the business event logging service and the remote calculation service.

Tamper-Proof Logging

An investment bank works with a large number of counter parties and semi-autonomous business units. One of the attractions of a secure SOA is that it enables the business to interact with its complex web of partners in a more streamlined fashion. An example of a service that promotes this sort of business activity is a tamper-proof logging service.

The framework has an implementation of an event logging service that uses hashing and encryption algorithms to make the log tamper-proof. The service exposes a Web service interface and is hosted by BizTalk orchestration that not only logs the business event but also publishes the event as a BizTalk message. This enables subscribing orchestrations to be triggered by business events of interest to them (see Figure 1).

As an example, consider the scenario where a bank client is using a bank-supplied application to obtain financial information. If the tamper-proof logging service exists, then both the sending and the receipt of the information can be recorded in the log. Each record will contain an encrypted hash of the message payload and the tamper-proof nature of the log guarantees that disputes can be settled with unambiguous information. Moreover, it would be possible to create an auditing service (written as a BizTalk orchestration) that subscribed to either or both of the send and receive events. In this way, BizTalk's efficient messaging subscription design allows services to be loosely coupled with each other.

It is easy to see that when a service such as tamper-proof logging exists within the bank, it can quickly become a component of many higher order services. One such service is the remote calculation service.

Remote Calculation Service

Spreadsheets are used extensively in an investment bank. Traders and back-office staff are very familiar with this sort of technology and build complicated models of trades, risk calculations, and so on. Spreadsheets have plug-ins for market data feeds from providers such as Reuters or Telerate and often make use of add-in analytic libraries that implement algorithms for calculating risk, curves, and prices.

Some of the algorithms run by traders, back office reconciliation, or business controlling staff take a surprisingly long time to run. Value-at-risk calculations tend to fall into this category and it is not unusual for a risk calculation to take a couple of hours to complete.

> "One of the challenges in implementing an SOA is enabling agility in the tailoring/enhancement of services"

To see why spreadsheets are used so widely for these sort of calculations, consider the trader shown in Figure 2. Here, the trader has their risk model, which they have honed over the years. The spreadsheet model is fed with live data from the market data feeds and the trader uses their expert knowledge to input candidate prices, rates, or spreads into the model. At the end of the trading day the trader will run their value-at-risk calculation to inform their trading decisions for the next day.

Now consider the position of the business controller who is responsible for assessing the value at risk for a whole trading floor. They are in the invidious position of taking each of the trader's spreadsheets in turn, validating that the market data has been approved, and running each calculation. This sort of situation was one reason for developing the remote calculation service (see Figures 3 and 4).

The remote calculation service is one of the base services of the framework and can run a computation remotely on behalf of its client – one computation that is supported is spreadsheet calculations (including the driving of macros and add-ins, etc). The service allows its client to schedule a computation immediately or for some time in the future on a recurring basis. The computation can also be scheduled to run on receipt of a business event logged via the logging service described earlier.

The remote calculation service is implemented as a BizTalk orchestration and has support to return the calculation results using a custom output orchestration. This feature, again, relies on the message subscription design of BizTalk. In this way, the results of calculations might be transformed and routed back to requesting users using their preferred transport mechanism (i.e., SMTP, FTP, etc.)

Figure 4 illustrates, in a simple way, how the SOA can be built using the lower level services and how the BizTalk's flexibility can be utilized to add application, regional, or even user-specific processing into the architecture.

Conclusion

Investment banks have complicated businesses that spread a round the world with many different trading partners. They operate in an environment of constant change and need to react very quickly to keep their competitive edge. This article has shown how one investment bank has reacted to these challenges and how BizTalk Server can help make the agile service-oriented architecture a reality.

About the Author

David Regan is a consultant specializing in XML technologies and service oriented architecture

david.regan@cassium.com

DEPLOY SOA. NOW.

Service-Oriented Architecture. It's the buzz in the world of enterprise software development and integration: Web services and service-oriented architecture (SOA). And at BEA, we're all over it.

SOA—and the growing use of discrete, self-contained business services—has become widely recognized as the most relevant strategic direction for enterprise computing. And increasingly, the BEA WebLogic™ development environment is recognized as the premier platform on which to implement it. Right now.

As a ground shaking new architecture, SOA is making waves throughout many organizations. Come to BEA eWorld 2004 to see how easily you can deliver SOA in your organization NOW!

BEA eworld 2004

REGISTER NOW!

www.bea.com/eworld

MOSCONE WEST CONVENTION CENTER **SAN FRANCISCO, CA** **MAY 24-27, 2004**

Thanks to our generous sponsors

Diamond Sponsors

hp invent

intel

Platinum Sponsors

accenture
High performance. Delivered.

COMPUWARE.

bea®

UDDI as an Extended Web Services Registry

Versioning, quality of service, and more...

As enterprises build a critical mass of Web services, they need some way of keeping track of those services. UDDI is an ideal store for such information.

WRITTEN BY
ADAM BLUM

Using UDDI's built-in abstractions of business services, binding templates, and tModels referring to interface specifications, UDDI can be used to manage all of the addresses and protocols and formats of those services. This information can be used for several purposes, including providing location independence and identification of common supported interfaces among those services.

But the amount of information tracked on each Web service is a small subset of the information that companies truly need to manage for each of these services. Developers want to access the actual formats used by those Web services within the registry. This includes objects from the WSDL descriptions of Web services, such as portTypes, and also includes other artifacts associated with Web services such as XML Schemas to represent types used by those WSDLs. They also want to be able to keep track of the versions of those Web services and their associated objects. IT administrators typically want to know a variety of information on each Web services' quality of service. They also want to know who has access to those services, and what applications and Web services may be dependent on those services.

This article discusses how UDDI can be used as this more robust registry of extensive information on each Web service. It describes UDDI tModels for categorization and shows how they are used in the "Using WSDL in a UDDI Registry, Version 2.0" Technical Note [WSDL-UDDI] to represent all of the information present in WSDL in UDDI. It then describes how the techniques used by the WSDL mapping technical note can be used to represent new classes and attributes of information generally.

We then show how to use these techniques to represent versioning information of Web service objects and quality of service information for a Web service. My company is working with the UDDI Technical Committee as well as customers and partner companies to create Technical Notes in each of these areas: versioning and quality of service metrics. Our hope is that the techniques developed in these areas will be useful models to extending UDDI to cover other areas of describing Web services mentioned above as well.

UDDI tModels

tModels, short for technical models, can register two types of things: specifications and categorizations. Specifications describe the Web service and can be expressed in Web Service Description Language (WSDL) files, XML Schema definitions, policy expressions, or just a reference to a Web page for a spec. Categorization tModels provide an extensible mechanism for adding property information to a UDDI entity. The uddi-org:types category indicates which type of tModel is at hand. The following example below demonstrates how to define a categorization tModel, and how to use a categorization tModel to categorize a UDDI entity. This tModel entity represents the WSDL Transport Categorization tModel as defined in the WSDL mapping technical note. It's used to specify what type of transport is supported by a Web service. This tModel includes a categoryBag that defines the tModel as a checked categorization tModel (see Listing 1).

Entities are categorized with the keyedReference structure on a categoryBag. A keyedReference element has a tModelKey attribute that refers to the unique ID of the categorization tModel. In this example, the tModelKeys in the two keyedReference structures refer to the uddi-org:types tModel. The keyName attribute is purely for human readability. The keyValue attribute contains the value for the category. The first keyedReference specifies that the tModel represents a taxonomy. The second keyedReference specifies that the values specified for the taxonomy are checked against a defined value set.

Taxonomies used for categorization may have validated value sets or may support arbitrary values.

WSDL Mapping

WSDL files describe a Web service's interfaces (portTypes), their encoding formats and transport protocols (bindings), specific endpoint locations (ports) that implement a binding, and services that contain a collection of named ports. Some of this information already exists in UDDI in some form. Specifically, WSDL ports are mapped to UDDI bindingTemplates. The bindingTemplate's tModelInstanceDetails collection points to the tModels generated for the WSDL portType and binding (described below). WSDL services are represented as UDDI businessService entities. More interesting than these mappings is how to represent WSDL artifacts that are not already present in UDDI.

WSDL PortTypes

The v2.0 WSDL mapping generates a UDDI tModel for each WSDL portType. The name of the tModel is the same as the local name of the portType in the WSDL specification. The overviewURL of the tModel is the URL of the WSDL specification.

Each portType tModel contains a categoryBag with at least two keyedReference structures. The first keyedReference indicates the type of WSDL artifact, and the second keyedReference indicates the namespace of the portType. The first keyedReference specifies the tModelKey referring to the WSDL Entity Type tModel (uddi-org:wsdl:types), which is a checked taxonomy of WSDL entity types. The taxonomy defines four allowed values: portType, binding, port, and service. The keyValue in this keyedReference will be "portType". The second keyedReference specifies the tModelKey referring to the XML Namespace tModel (uddi-org:xml:namespace), which is an unchecked taxonomy. The keyValue in this keyedReference will contain the target namespace of the WSDL document that defines the portType. These taxonomies are defined in the WSDL mapping technical note.

WSDL Bindings

In similar fashion, the v2.0 WSDL mapping generates a UDDI tModel for each WSDL binding, with the name of the tModel gathered from the local name of the WSDL binding, and the overviewURL being the URL of the WSDL definition file. Using categoryBags with keyedReferences:

- The tModel is categorized as a WSDL entity of type "binding".
- The namespace category specifies the target namespace of the WSDL document that defines the binding.
- A portType category is used to refer to the portType tModel that was created for the WSDL portType (as described above).
- Protocol and transport categories indicate what XML and transport protocols (e.g., SOAP 1.1 and HTTP) the binding supports.

Techniques for Information Extension

Looking at how WSDL bindings and portTypes were represented in UDDI, we can see some generally useful approaches to putting new types of information associated with Web services into the registry.

tModels as Classes

To represent new classes of information associated with Web services, we can generate additional specification tModels that we can then categorize (using taxonomy tModels) to represent the attributes of those classes. An example of a new class of information might be "quality of service information for a given binding template" – a QoS Information tModel. The specific QoS information that applies to the businessService would be specified using keyedReferences in the QoS Information tModel. The UDDI businessService entity would include a reference to its associated QoS Information tModel by including the tModel in the tModelInstanceDetails structure in its bindingTemplate. (The tModelInstanceDetails structure points to a set of specification tModels that collectively define the "technical fingerprint" of the service. According to the WSDL mapping technical note, the tModelInstanceDetails structure points to (at least) the portType and binding tModels. We are suggesting that an additional tModel be included in this list to represent QoS information.)

A QoS Information tModel could also be referenced directly from the businessService entity or from a UDDI version 3 bindingTemplate entity using a keyedReference structure in the entity's categoryBag.

Categories as Attributes

To annotate a UDDI entity with attributes, create new categorization tModels to represent the property types. The categorization tModels can be included in the categoryBag structures on UDDI businessService, bindingTemplate, or the new "class tModels" that we just described. UDDI business services and UDDI v3 bindingTemplates can link to these tModels by adding a keyedReference structure to the categoryBag on the businessService or bindingTemplate that references these new categories. Examples of this technique from the mapping spec are the WSDL Entity Type and XML Namespace tModel that are placed as categories on the UDDI business services themselves as well as the tModels which represent WSDL portTypes and bindings. If there is a

set of related categories, the new UDDI v3 keyedReferenceGroup can be used to associate related categories together.

Versioning Information

Let's try to put these techniques to work to represent some of the other information about Web services discussed above. One particular area of interest we have heard from many customers is a standard way of representing the versions of various Web service objects: business services, binding templates, portType tModels, and binding tModels. It would be helpful to categorize each of these objects with:

- Major and minor version numbers
- Revision or build numbers
- Named milestones or phases
- A label to allow associating related objects together

Let's create some new taxonomies to represent this information. Let's assume that the taxonomy names are "uddi:uddi-org:versioning:major-version", "uddi:uddi-org:versioning:minor-version", "uddi:uddi-org:versioning:revision-number", "uddi:uddi-org:versioning:milestone", and "uddi:uddi-org:versioning:versioning-component", although of course these actual names (prefixed as they are by "uddi-org") would not be used unless this proposal actually did become a Technical Note from the UDDI TC.

Listing 2 shows you how to use all of these taxonomies to categorize a portType tModel as version 2.1, build 1241, and the latest revision. It also uses the "versioning-component" tModel with a value of "StockQuoteSampleComponent" to tie together the portType with other UDDI objects (business services, binding templates, binding tModels, other portType tModels, and other versions of the portType tModel). The versioning-component taxonomy is used to establish an association between two versions of the same object. That is, it can be used to establish that two portType tModels are versions 1 and 2 of the same interface (relying on the name is not a reliable way of doing that). It can also be used to associate various UDDI objects as part of the same component. As portType tModels get replaced, there may not be a bindingTemplate that points to the old versioned portType tModels. So using this "versioning-component" is a reliable way to be able to ask "give me all of the UDDI objects associated with version n of this component" (rather than relying on business services containing binding templates which point to all supported tModels.

Quality of Service Information

For another example of extending UDDI with more information, let's also provide a method of representing quality of service metrics. Systinet is working with AmberPoint, a leading Web services management vendor, to author a Technical Note in this area. The example below represents an approximation of the currently envisioned approach, but this may change by the time the TN is released.

In this example, we represent quality of service information for each Web service deployment by generating a tModel for quality of service information for the binding template that represents the deployment. We then assign properties for each QoS metric on the generated tModel using keyedReferences. This QoS information allows companies to track the actual delivered quality of service of a particular physical implementation of a Web service according to many criteria. Examples of this might be classifications of average response time (fast, medium, slow), categories of average throughput (>10 Mbps, 1-10Mbps, <1Mbps), and overall reliability (99.99%, 99.9%, 99%).

Listing 3 is an example of the bindingTemplate reference to the tModel that will have the QoS attribute categories attached. It starts with the Stock Quote Service example used in the WSDL mapping technical note, but for simplicity does not include the WSDL artifacts that the example uses. In order to retrieve more detailed management information, we store the location of a WSDL in a keyed reference with tModelKey "uddi:uddi.org:QoS-Detail". This tModel has the reference to a binding template of a "management Web service" endpoint which provides operations for more detailed analysis of the given Web service's performance.

Listing 4 is an example of the QoS Information tModel, which contains a categoryBag specifying QoS metrics.

Additional Standards for Enhanced Web Services Information

We've just described how a given company can use the extensibility methods in UDDI to track more extensive information about Web services than just their locations, access protocols, and supported formats. Specifically, we have shown how we can use categories on UDDI objects to represent both versioning information and quality of service information. As mentioned, we expect to finalize Technical Notes working with the OASIS UDDI TC, in these areas as well.

In addition, we are helping to create ways of representing other classes of Web services information, as additional technical notes under the aegis of OASIS' UDDI Technical Committee. Some of the areas that we think are of most immediate value to track for each Web service include:

- Extending the versioning information to "lifecycle information" describing attributes of the milestones referred to
- More detailed WSDL mapping, storing individual operations and types in the registry
- Additional resources related to Web services such as XML schemas and individual XML documents and mappings between interface formats
- Dependency information between Web services and on Web services
- Access permissions (what developers have access to what services)

Summary

UDDI is an excellent basis for an enterprise Web services registry. Incremental standards such as the "Using WSDL in a UDDI Registry" technical note demonstrate methods of storing additional information in UDDI. Companies can start leveraging these approaches to add more information on additional attributes of Web services, such as quality of service and versioning information, to their enterprise UDDI registries. Incremental standards in the form of UDDI Technical Notes are emerging for these attributes of Web services and others to allow tracking of this information.

Acknowledgements

Thanks to Anne Thomas Manes, Zdenek Svoboda, Mirek Novotny, Fred Carter, and Luc Clement for helpful feedback on topics related to this article.

References

- *"Using WSDL in a UDDI Registry, Version 2.0"*: www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm

About the Author

Adam Blum is an evangelist for Systinet Corporation. He spent five years with Microsoft, most recently as a manager in Microsoft's SQL Server product group. Adam has also worked as vice president of Engineering for CommerceOne, where he managed development of all XML-related tools and platforms, including MarketSite, the world's first XML-based document integration server.

adam.blum@systinet.com

Listing 1

```
<tModel tModelKey="uuid:e5c43936-86e4-37bf-8196-1d04b35c0099">
  <name>uddi-org:wsdl:categorization:transport</name>
  <overviewDoc>
      <overviewURL>
        http://www.oasis-open.org/committees/uddi-
spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm#transport
      </overviewURL>
  </overviewDoc>
  <categoryBag>
      <keyedReference
         tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
         keyName="types"
         keyValue="categorization" />
      <keyedReference
         tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"
         keyName="types"
         keyValue="checked" />
  </categoryBag>
</tModel>
```

Listing 2

```
<tModel tModelKey="uddi:mycompany.com:stockquoteporttype" >
     <name>
          StockQuotePortType
     </name>
     <overviewDoc>
          <overviewURL>
              http://location/sample.wsdl
          <overviewURL>
     <overviewDoc>
     <categoryBag>
       <keyedReference
         tModelKey="uddi:uddi.org:versioning:major-version"
         keyName="Major Version"
         keyValue="2"/>
       <keyedReference
 tModelKey="uddi:uddi.org:versioning:minor-version"
 keyName="Minor Version"
 keyValue="1"/>
       <keyedReference
         tModelKey="uddi:uddi.org:versioning:revision-number"
         keyName="Revision Number"
         keyValue="Build 1241"/>
       <keyedReference
         tModelKey="uddi:uddi.org:versioning:milestone"
         keyName="Milestone"
         keyValue="latest-revision"/>
       <keyedReference
         tModelKey="uddi:uddi.org:versioning:versioning-
           component"
         keyName="Versioning Component"
         keyValue="StockQuoteSampleComponent"/>
       <keyedReference
         tModelKey="uddi:uddi-org:xml:namespace"
         keyName="portType namespace"
         keyValue="http://example.com/stockquote/"/>
       <keyedReference
         tModelKey="uddi:uddi-org:wsdl:types"
         keyName="WSDL type"
         keyValue="portType"/>
     </categoryBag>
</tModel>
```

Listing 3

```
<businessService
      serviceKey="uddi:mycompany.com:StockQuoteService"
      businessKey="uddi:mycompany.com:business>
     <name>Stock Quote Service</name>
     <bindingTemplates>
          <bindingTemplate

bindingKey="uddi:mycompany.com:StockQuoteService:primaryBinding"

serviceKey="uddi:mycompany.com:StockQuoteService">
               <accessPoint URLType="http">
                   http://location/sample
               </accessPoint>
               <tModelInstanceDetails>
                   <tModelInstanceInfo

tModelKey="uddi:mycompany.com:StockQuoteService:Primary
   Binding:QoSInformation">
                        <description xml:lang="en">
                        This is the reference to the tModel
that will have all of the QOS related categories attached.
                        </description>
                   </tModelInstanceInfo>
                   <tModelInstanceInfo

tModelKey="uddi:mycompany.com:StockQuoteService:Primary
   Binding:QoSDetail">

                        <description xml:lang="en">
                            This points to the tModel that
has the reference to the web service endpoint that allows
detailed retrieval of information
                        </description>
                   </tModelInstanceInfo>
               </tModelInstanceDetails>
          </bindingTemplate>
     </bindingTemplates>
</businessService
```

Listing 4

```
<tModel
tModelKey="mycompany.com:StockQuoteService:PrimaryBinding:QoSI
nformation"" >
     <name>
          QoS Information for Stock Quote Service
     </name>
     <overviewDoc>
          <overviewURL>
              http://<URL describing schema of QoS
                attributes>
          <overviewURL>
     <overviewDoc>
     <categoryBag>
          <keyedReference
               tModelKey="uddi:uddi.org:QoS:ResponseTime"
                keyName="Average ResponseTime"
               keyValue="fast" />
          <keyedReference
               tModelKey="uddi:uddi.org:QoS:Throughput"
                keyName="Average Throughput"
               keyValue=">10Mbps" />
          <keyedReference
               tModelKey="uddi:uddi.org:QoS:Reliability"
                keyName="Average Reliability"
               keyValue="99.9%" />
     </categoryBag>
</tModel>
```

Advanced Web Services Policies & Microsoft WSE

Use WS-Policy support in Microsoft WSE to secure your .NET Web services

While widely adopted or standardized security protocols are great for interoperability, a set of SOAP message header elements as well as a few new elements that belong in the message body are outside the scope of the existing mechanism for publishing service descriptions, which is WSDL.

WRITTEN BY
JEANNINE HALL GAILEY

In my previous article (***WSJ***, Vol. 4, issue 3), I covered the basic tenets of the WS-Security (recently renamed by OASIS as SOAP Message Security). This specification leverages existing XML security functionality to enable a uniform way to implement security for SOAP messages. WS-Security implements a set of SOAP message header elements as well as a few new elements that belong in the message body. While WSDL is designed to describe components of a Web service, such as exposed Web methods, data types, ports, and SOAP message elements, it has proven less useful when it comes to describing choices, preferences, and other requirements not tied directly to the messaging interface itself. It's all well and good if our Web service rejects all incoming SOAP requests that don't contain a certain type of security token or that don't have the required signatures, but how do we communicate these requirements to clients that want to access the service? What is needed is a more comprehensive way of describing a Web service – not just the interface, but requirements and expectations as well.

Introducing WS-Policy

Enter the WS-Policy family of specifications. Proposed by IBM, BEA, SAP, Microsoft, and others, WS-Policy simply defines a framework and mechanisms for constructing policy expressions that describe the requirements for accessing a Web service. In this policy-based model for Web services, individual requirements are declared using XML policy assertion elements. An assertion might declare something like "if you send me a request, you need to digitally sign the message" or "I only accept X.509-based security tokens" or even "I only accept messages that conform to version X of a given specification." A policy expression can be comprised of one or more policy assertions assembled using logical policy operators, and this expression can also be associated with a Web service resource (such as a service or endpoint) using WSDL or other mechanisms defined in WS-PolicyAttachment.

Declaring Policy Assertions

Policy assertions are the building blocks of policies. Each assertion describes an atomic aspect of a service's requirements. WSE 2.0 supports the following policy assertions, which are defined in the WS-SecurityPolicy specification:

- ***SecurityToken:*** Declares that a security token be contained in or referenced from the message and defines which type of security token can be used to secure a message as well as attributes of the token.
- ***Integrity:*** Declares that one or more digital signatures are required and defines which parts of the message are signed, the signing algorithm, and information on the token used for signing.
- ***Confidentiality:*** Declares that part of the message is encrypted and defines which message parts are encrypted, the encryption algorithm, and information on the token used for encryption.

Other policy assertions, which are used to specify encoding, language, and versioning requirements, are described in WS-PolicyAssertion.

Building Complex Policy Expressions

Consider the simple policy expression shown in Listing 1. The All element is a policy operator that defines how the policy attributes, child elements of All that can be either assertions or other operators, are

applied. Since the use of All indicates that the requirements of all child elements must be met, the net result for this policy expression is that any request message must contain an X.509-based security token, a digital signature of the message created using an X.509-based token, and a body element encrypted using an X.509-based token. WS-Policy defines a set of policy operators that enables us to construct policy assertions into logical blocks to better describe complex policy requirements. These policy operators include:

- ***All:*** Message must conform to all of the assertion and operator elements that are direct children of this element.
- ***ExactlyOne:*** Message must conform to exactly one of the assertion or operator elements that are direct children of this element, and the assertion or operator with the highest value for the Preference attribute indicates the preferred child element.
- ***OneOrMore:*** Message may conform to one or more of the assertion or operator elements that are direct children of this element, and the assertion or operator with the highest value for the Preference attribute indicates the preferred child element.

You can use these operators to construct policy expressions that can accurately describe the complex requirements of your Web services. For example, the policy expression in Listing 2 describes a Web service that accepts three kinds of security tokens: Username, X.509-based, and security context tokens (SCT), where use of each type of security token has a unique set of requirements. In this example, the top-level ExactlyOne operator declares that only one type of authentication be implemented in the request, and the Preference attribute is used to indicate that security context tokens are most desired (Preferrence="90") while username tokens are accepted but least desired (Preferrence="30"). Each child of ExactlyOne is treated as a logical policy block, where All child elements are used to indicate that each authentication mechanism "block" requires a security token as well as either a digital signature, encryption, or both.

While a service that accepts three types of security tokens might not be common in the real world, it does demonstrate the versatility of the policy framework.

Introducing WSE 2.0

Since I described Web Services Enhancements (WSE) version 2.0 for Microsoft .NET in my first article, I will provide only a brief recap before discussing policy support in WSE 2.0. WSE is a .NET assembly that supports both TCP and HTTP transports and implements a set of input and output filters that read incoming SOAP messages and translate known SOAP header elements into WSE programming objects. In a similar fashion, WSE output filters construct SOAP headers based on the properties of the SoapContext object for the outgoing message. WSE also implements a policy engine that parses policy cache documents to ensure that incoming and outgoing messages conform to the service's defined policy. WSE provides a rich API to enable programming WSE from your applications.

Support for Policy in WSE 2.0

While policies were initially conceived as a way to extend WSDL to better communicate the requirements for clients to successfully interact with a Web service, WSE 2.0 puts an interesting spin on this original intent. While it does not directly support the attachment of policies to WSDL (as defined in WS-PolicyAttachment), it does use policy expressions to better automate the implementation of Web service security. When you create and store one or more policy expressions in the Web service's receive-policy cache, WSE will use this policy document when it evaluates incoming request messages, rejecting all requests that do not conform to the effective policy. Similarly, WSE will consult the send-policy cache document to ensure that outgoing messages conform to it as well. The really cool thing about the send-policy cache is that WSE will attempt to automatically generate the required security headers to bring the message into conformance with the policy. This enables you to secure outbound messages based only on a well-constructed policy document and without writing a line of WSE code.

A single policy cache document can contain multiple policy expressions that each map to a unique resource hosted by the Web service. For example, you might define a default policy for your Web service and a separate policy for a security token service hosted by your service. WSE uses the Policy element's ID attribute to map a Web service resource to a specific policy. It uses the policyDocument element to denote a policy cache document. Listing 3 shows a receive-side policy cache document that contains two policy expressions, one that is mapped to default, which is used for all requests to the Web service's virtual directory, and a second mapped to http://myservice/secureConversation.ashx, which is used for specific requests to the WSE-provided security context token service. (*Note:* These policy expressions have been truncated for readability.) In order for WSE to know to use a given policy document as the receive-side or send-side policy cache, the web.config file for the service is modified to point to the policy documents. Listing 4 shows how the policy element in the micrsoft.web.services configuration block is used to specify send- and receive-side policy cache documents.

Defining Policies Using WSE Tools

Since WSE 2.0 supports WS-Policy, WS-PolicyAttachment, and WS-SecurityPolicy, you could manually construct a policy document based on these specifications. However, hand-coding XML can easily result in errors that will prevent WSE from loading parsing the policy document. The same applies when editing the web.config file to tell WSE to apply send- or receive-side policies. The easiest way to use policies with any WSE-enabled application is to use the WSE Settings tool, which is a Visual Studio add-in that enables you to easily configure WSE. Note that this tool is essentially the same for both client and service endpoints, except that changes are made in either web.config for an ASP.NET Web service or in app.config for a WSE-enabled client application. When defining policies, it is important that they represent the behavior of the WSE at a given endpoint. For example, don't specify the use of username tokens if you haven't written a custom username token handler, and don't specify encryption with a username token because WSE doesn't support it.

To define policies for your project:

1. Right-click the project in the Visual Studio Solution Explorer window and select WSE Settings 2.0. This displays the WSE Settings Tool.
2. In the General tab, check both checkboxes to enable WSE for the Web service.

(When enabling WSE for a client application, the ASP.NET checkbox will be grayed out.)

3. In the Security tab specify the security behaviors of WSE.
4. In the TokenIssuing tab, specify if a security context token service or other custom token service is hosted by WSE (Web service endpoints only).
5. In the Policy tab (see Figure 1), check the Enable Policy checkbox and click Add. This starts the WSE Security Settings Wizard.
6. On the Choose the Type of Application screen of the wizard, select either a client or service application and if Secure Conversation (using SCTs) is required. When requiring Secure Conversation, incoming messages must have an SCT, be signed and encrypted.
7. On the Message Settings page of the Wizard, select the security policy requirements for incoming and outgoing messages.
8. On the Client Authentication Token screen of the Wizard, select the type of token (Username, Kerberos-based, or X.509-based) is required for authentication and securing messages.
9. For a Web service and depending on the type of security token selected in Step 8, the next screen of the wizard enables you to restrict who can access the service, but this level of restriction is optional.
10. If there are requirements for encryption on incoming messages or signatures on outgoing messages, you must specify an X.509-based certificate for this on the Server Certificate screen of the wizard (see Figure 2).
11. Create Security Settings, the final screen of the wizard shown in Figure 3, displays a recap of the security policy settings that are created when you click OK to complete the wizard.
12. Click OK to close the WSE Settings tool. WSE automatically modifies the appropriate configuration file for the application so that it references the newly created policy cache document.

Beyond the WSE Policy Tools

As you have seen, WSE tools are able to generate very simple policy documents, containing only one Integrity assertion and one Confidentiality assertion, and it is not even able to create an independent SecurityToken assertion. However, the policy parser implemented by the WSE runtime fully supports the WS-Policy specification. This means that WSE can handle policy expressions as complex as the one shown in Listing 2, which implements the full range of security policies as well as all of the operators defined in WS-Policy. The best way to create a policy document as complex as the one in Listing 2 is to define a policy using the WSE Setting tool to generate your assertions, which can be arranged into logical

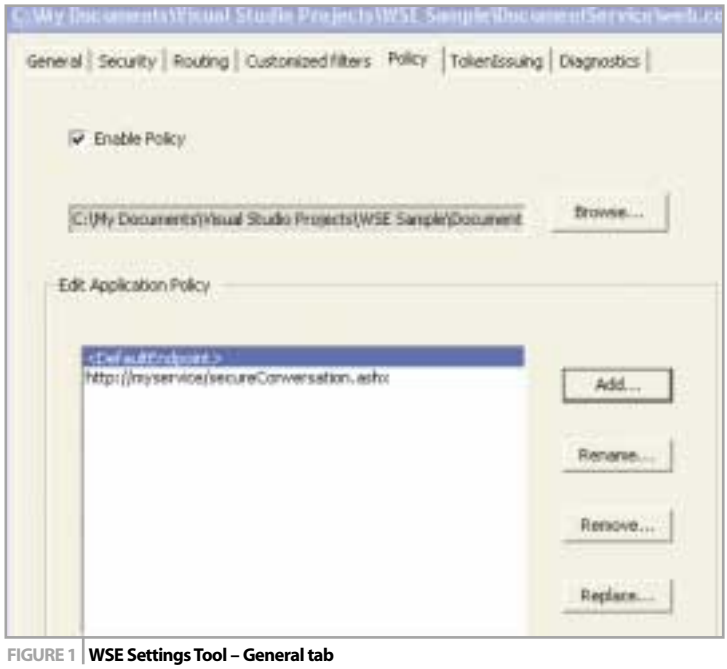


FIGURE 1 | **WSE Settings Tool – General tab**

FIGURE 2 | **WSE Settings Tool – Policy tab**

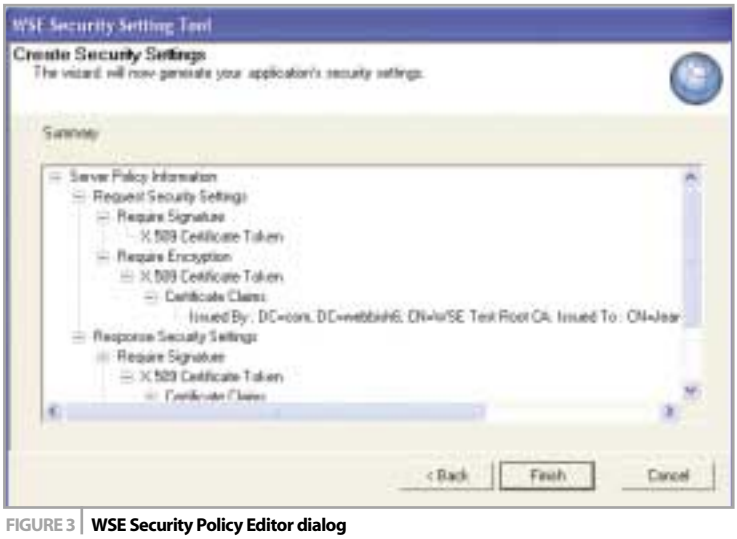


FIGURE 3 | **WSE Security Policy Editor dialog**

groups using the three policy operators, All, ExactlyOne, and OneOrMore – following the rules in WS-Policy.

Limitations of Policy in WSE

Since WSE 2.0 does not modify WSDL as specified in WS-PolicyAttachment, you have to communicate policies to your clients either by manually editing the WSDL document to reference the location of your Web service's policy document or by sending the policy document to the client using some other out-of-band mechanism. It would be convenient to have the Web service implement a method that returns the policy document. However, WSE must positively authenticate every request using a supplied security token, or how will a client know which tokens are accepted without access to the policy.

Conclusion

In the emerging world of well-secured Web services, WS-Policy provides a much needed mechanism for describing the security requirements for accessing Web services. While WSE 2.0 does not enable the communication of policies, it does help you to easily create policy expressions and leverages them to ensure that incoming and outgoing messages conform to these policies. One of the most useful features of WSE is the ability to use the send-side policy cache to automatically secure outgoing messages without writing a line of code. While WSE supports the policy assertions defined in WS-SecurityPolicy and the policy operators defined in WS-Policy, its tools are only able to define rudimentary polices. In future releases, I look forward to improved tool support and support for additional policy assertions out of the box.

About the Author

Jeannine Hall Gailey is a former Microsoft manager whose book *Understanding Web Services Specifications and the WSE* is available now from major bookstores. She has published numerous technical articles on Web services.

author@jeanninegailey.com

Listing 1

```
<wsp:Policy
xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"

xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/06/secext">
  <wsp:All>
    <wsse:SecurityToken wsp:Usage="wsp:Required">
      <wsse:TokenType>wsse:X509</wsse:TokenType>
    </wsse:SecurityToken>
    <wsse:Integrity wsp:Usage="wsp:Required">
      <wsse:Algorithm Type="wsse:AlgSignature"
        URI="http://www.w3.org/2000/09/xmlenc#aes" />
    </wsse:Integrity>
    <wsse:Confidentiality wsp:Usage="wsp:Required">
        <wsse:Algorithm Type="wsse:AlgEncryption"
          URI="http://www.w3.org/2001/04/xmlenc#3des-
cbc"/>
        <MessageParts>
            wsp:GetNodesetForNode(wsp:GetBody(.))
        </MessageParts>
    </wsse:Confidentiality>
  </wsp:All>
</wsp:Policy>
```

Listing 2

```
<wsp:Policy wsu:Id="default-receive-policy"
  xmlns:wsp="...">
  <wsp:ExactlyOne>
    <wsp:All Preference="30">
      <wsse:SecurityToken wsp:Usage="wsp:Required"
        xmlns:wsse="...">
        <wsse:TokenType>http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-username-token-pro-
file-1.0#UsernameToken
        </wsse:TokenType>
      </wsse:SecurityToken>
      <wsse:Integrity wsp:Usage="wsp:Required"
        xmlns:wsse="...">
        <wsse:TokenInfo>
          <SecurityToken>
            <wsse:TokenType>http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-username-token-pro-
file-1.0#UsernameToken
            </wsse:TokenType>
          </SecurityToken>
        </wsse:TokenInfo>
        <wssp:MessageParts  Dialect="..."
          >wsp:Body() wsp:Header(wsa:Action)
          wsp:Header(wsa:FaultTo) wsp:Header(wsa:From)
          wsp:Header(wsa:MessageID)
          wsp:Header(wsa:RelatesTo)
          wsp:Header(wsa:ReplyTo) wsp:Header(wsa:To)
          wse:Timestamp()
        </wssp:MessageParts>
      </wsse:Integrity>
    </wsp:All>
    <wsp:All Preference="60">
      <wsse:SecurityToken wsp:Usage="wsp:Required"
        xmlns:wsse="...">
      <wssp:TokenType
        >http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509v3
      </wssp:TokenType>
    </wsse:SecurityToken>
    <wsp:OneOrMore>
      <wsse:Integrity wsp:Usage="wsp:Required"
        xmlns:wsse="...">
        <SecurityToken xmlns="...">
          <wssp:TokenType
            >http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3</wssp:TokenType>
        </SecurityToken>
      </wsse:TokenInfo>
      <wssp:MessageParts Dialect="..."
        >wsp:Body() wsp:Header(wsa:Action)
        wsp:Header(wsa:FaultTo) wsp:Header(wsa:From)
        wsp:Header(wsa:MessageID)
        wsp:Header(wsa:RelatesTo)
        wsp:Header(wsa:ReplyTo) wsp:Header(wsa:To)
        wse:Timestamp()</wssp:MessageParts>
    </wsse:Integrity>
    <wsse:Confidentiality wsp:Usage="wsp:Required"
      xmlns:wsse="...">
      <wsse:KeyInfo>
        <SecurityToken xmlns="...">
          <wssp:TokenType
            >http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3</wssp:TokenType>
        </SecurityToken>
        <wsse:MessageParts Dialect="..."
```

```
          >wsp:Body()</wsse:MessageParts>
        </wsse:Confidentiality>
      </wsp:OneOrMore>
    </wsp:All>
    <wsp:All Preference="90">
      <wsse:SecurityToken wsp:Usage="wsp:Required"
        xmlns:wsse="...">
        <wssp:TokenType
          >http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509v3</wssp:TokenType>
        <wsse:TokenIssuer
>http://myservice/documentservice/secureConversation.ashx</
wsse:TokenIssuer>
      </wsse:SecurityToken>
      <wsp:OneOrMore>
        <wsse:Integrity wsp:Preference="100"
          wsp:Usage="wsp:Required" xmlns:wsse="...">
          <wsse:TokenInfo>
            <wsse:SecurityToken wsp:Usage="wsp:Required"
              xmlns:wsse="...">
              <wssp:TokenType
                >http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3</wssp:TokenType>
              <wsse:TokenIssuer

>http://myservice/documentservice/secureConversation.ashx</
wsse:TokenIssuer>
            </wsse:SecurityToken>
          </wsse:TokenInfo>
          <wssp:MessageParts Dialect="..."
            >wsp:Body() wsp:Header(wsa:Action)
              wsp:Header(wsa:FaultTo)
wsp:Header(wsa:From)
              wsp:Header(wsa:MessageID)
              wsp:Header(wsa:RelatesTo)
              wsp:Header(wsa:ReplyTo) wsp:Header(wsa:To)
              wse:Timestamp()</wssp:MessageParts>
          </wsse:Integrity>
          <wsse:Confidentiality wsp:Preference="50"
            wsp:Usage="wsp:Required"
            xmlns:wsse="...">
            <wsse:KeyInfo>
            <wsse:SecurityToken wsp:Usage="wsp:Required"
              xmlns:wsse="...">
              <wssp:TokenType
                >http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-
1.0#X509v3</wssp:TokenType>
              <wsse:TokenIssuer

>http://myservice/documentservice/secureConversation.ashx</
wsse:TokenIssuer>
            </wsse:SecurityToken>
          </wsse:KeyInfo>
        <wsse:MessageParts Dialect="..."
          >wsp:Body()</wsse:MessageParts>
        </wsse:Confidentiality>
      </wsp:OneOrMore>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Listing 3

```
<?xml version="1.0" encoding="utf-8" ?>
<policyDocument xmlns="...">
  <mappings xmlns:wse="...">
    <!-- Mapping for all non-SCT requests to the service.
-->
    <defaultEndpoint>
      <defaultOperation>
        <request policy="#default-request-policy" />
        <response policy="#default-response-policy" />
        <fault policy="" />
      </defaultOperation>
    </defaultEndpoint>
    <!-- Mapping for all SCT requests to the service. -->
    <endpoint

uri="http://localhost/documentservice/secureConversation.as
hx">
      <defaultOperation>
        <request policy="#sct-request-policy" />
        <response policy="#sct-response-policy" />
        <fault policy="" />
      </defaultOperation>
    </endpoint>
  </mappings>
  <policies xmlns:wsu="..." xmlns:wsp="..."
xmlns:wssp="..."
     xmlns:wse="..." xmlns:wsse="..." xmlns:wsa="..."
     >
     <wsp:Policy wsu:Id="default-request-policy">
       ...
     </wsp:Policy>
     <wsp:Policy wsu:Id="default-response-policy">
       ...
     </wsp:Policy>
     <wsp:Policy wsu:Id="sct-request-policy">
       ...
     </wsp:Policy>
     <wsp:Policy wsu:Id="sct-response-policy">
       ...
     </wsp:Policy>
   </policies>
</policyDocument>
```

Listing 4

```
<configuration>
  ...
  <microsoft.web.services>
    <security>
      ...
    </security>
    <tokenIssuer>
      ...
    </tokenIssuer>
    <diagnostics>
      ...
    </diagnostics>
    <policy>
      <cache
name="C:\LocalPolicies\DocumentService\policyCache.config"
/>
    </policy>
</microsoft.web.services>
</configuration>
```

XML JOURNAL SUPPLEMENT

This Month

Automating B2B Integration with XML
BY SURESH DAMODARAN & NEELAKANTAN KARTHA

This article will explain how XML is used to enable businesses to work together via the Internet, in the context of the RosettaNet B2B framework.

Streamline Your XML Searches
BY KEITH SWENSON

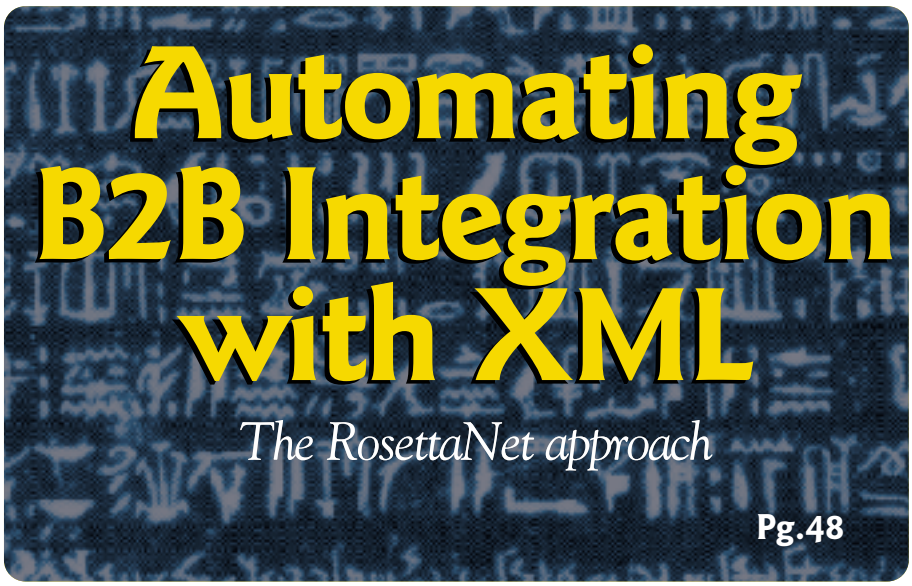
Imagine a customer has hired you to put together a solution for managing a huge quantity of XML information. How do you manage and search for records? This article introduces an alternative to a traditional database for storing and searching huge quantities of XML encoded data.

Using XML to Deliver Critical Messaging Services
BY MIKE SEGURA

One of the biggest risks in communication around natural disasters and city-wide emergencies has traditionally been the reliability and accessibility of wired and wireless systems. One solution may be using XML to deliver critical messaging services that, when layered on top of traditional applications, can for the first time spread risk across both wire and wireline services.

Understanding Information Transformation
BY DAVID S. LINTHICUM

The transformation layer understands the format of all information being transmitted among the applications and translates that information on the fly, restructuring data from one message so that it makes sense to the receiving application or applications. This article covers several important aspects of transformation.



XML-Based Interop, Close Up

In addition to the strategy side of Web services, there is also the protocol-oriented side of things, the XML side. Embracing not only XML itself but also the full range of mainstream XML-based technologies like XPath, XSLT, XML Schema, and SOAP, *XML-Journal* has been delivering insightful articles to the world of developers and development managers since the year 2000.

It is our privilege to bring *XML-Journal* directly to readers of *Web Services Journal*, and vice versa. Anyone already familiar with the Web services world of SOAP, UDDI, and WSDL will find here articles and features each month that will interest them – about the cutting-edge technologies and latest products that are changing not only our industry, but the way the world exchanges information. To make it easy for you to find your way around, we have four distinct sections:

Content Management:
Organization, dissemination, and presentation of information

Data Management:
Storage, transformation, representation, and general use of structured and unstructured data

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

Labs:
Product reviews, book reviews, tutorials, and standards analysis

Automating B2B Integration with XML

The RosettaNet approach

WRITTEN BY
SURESH DAMODARAN & NEELAKANTAN KARTHA

This article will explain how XML is used to enable businesses to work together via the Internet, in the context of the RosettaNet B2B framework. Looking at proven frameworks such as RosettaNet is important as it provides insight into what works today, and what will become important tomorrow. The article also offers a snapshot of how pervasive XML is in business integration frameworks, and outlines how XML technology can reduce the implementation costs of B2B integration solutions.

RosettaNet

RosettaNet, founded in 1998, is a nonprofit consortium of more than 500 of the world's leading information technology, electronic components, logistics, semiconductor manufacturing, solution provider, and telecommunications companies working to create, implement, and promote open e-business process standards. RosettaNet is named after the Rosetta stone, which is inscribed with the same message in three languages. The Rosetta stone enabled scholars to decipher Egyptian hieroglyphics – something they'd previously been unable to do. In a similar way, the mission of RosettaNet is to establish a common language and standard processes for business-to-business (B2B) transactions. In 2002, RosettaNet merged with Uniform Code Council (UCC), the organization that administers UPC symbols, in an effort to apply the RosettaNet approach to other vertical industries (such as consumer goods).

RosettaNet was founded to improve supply-chain efficiency, to increase supply-chain visibility, and to enhance collaboration in a secure and reliable fashion across trading networks separated geographically. RosettaNet has achieved tremendous adoption of its standards in the high-tech manufacturing supply chain over the past three years, with more than 3,000 documented production implementations and a growth rate of approximately 500% from 2001 to 2003.

As background for discussing the technical components of RosettaNet, it is necessary to define business processes and to distinguish between two kinds of business processes – public and private.

Business Processes

A business process consists of a set of steps that, when executed, accomplish a certain business goal. For instance, a customer issues a request for a product from a supplier by sending a purchase order containing the descriptions of the items to be purchased. The supplier checks for the availability of the items in its inventory before accepting or rejecting the customer's purchase of the specific items. These steps constitute a purchase order business process.

To the customer in the purchasing scenario, some business process steps are neither visible nor particularly relevant. For instance, checking the availability of the items in the supplier's inventory is one such step that is particularly relevant to the supplier but not visible to the customer. The difference in relevance and visibility of the business process steps to the participants is the basis for distinguishing between two kinds of business processes, public and private business processes. Public business processes consist of steps that are visible to both parties conducting the business, whereas private business processes consist of steps that are visible only to one party. As this example illustrates, executing a public business process often requires executing private business processes as well.

Rationale for Specifying Business Processes

In the past, the steps of both public and private business processes were typically undocumented and considered part of a company's "way of doing business." In other words, while each company had business processes that it used to accomplish select business transactions, such as satisfying a customer order, ordering supplies, or responding to requests for support, the actual steps and the software needed to accomplish these steps were internal to the company and its employees. This was not a very satisfactory state of affairs for a number of reasons:

1. When business processes are undocumented, they are not amenable to automation. Automation of business components (such as standardizing business documents using EDI and XML) increases the incentive to automate business processes and increase the effectiveness and efficiency of the business organization.
2. With the advent of the Internet, new ways of doing business are becoming more and more common. Along with these opportunities comes a need to embrace new business processes and to change existing ones. Assessing the impact of modifying an existing business process is difficult when the business process is undocumented.
3. Documenting business processes in a rigorous fashion helps in dividing the labor between business analysts who are familiar with the business processes and technical experts who are familiar with implementing the various steps of the business process in software.

Several languages for specifying business processes have emerged in recent years: BPSS (Business Process Specification Schema) from ebXML (www.ebxML.org); BPEL (Business

Process Execution Language) from OASIS (www.oasis-open.org); and BPML (Business Process Modeling Language) from BPMI (www.bpmi.org).

Business Processes in RosettaNet

RosettaNet focuses on standardizing and automating public processes – it does not attempt to standardize private processes. RosettaNet enhances public processes by standardizing the sequence of operations in a business process, the business documents themselves, and the messaging system that sends and receives these documents (see Figure 1). RosettaNet differs from the previous industry standards efforts such as ANSI EDI in that RosettaNet developed its standards using XML for the Internet-based business transactions. The RosettaNet approach also differs from those of other standards initiatives in two distinct ways:

1. RosettaNet provides its members with considerable support while they implement the standards
2. RosettaNet undertakes standardization of a new business process only if the member entities involved commit to implementing the business process.

Standardizing business processes and documents in RosettaNet – PIP

A Partner Interface Process (PIP) specifies the business documents that are exchanged within a business process, the sequence in which the messages are exchanged, and the physical attributes of the messages that define the quality of service. The business documents and the exchange sequence are specified in XML. Today, more than 50 PIPs have been validated by actual implementation between two trading partners, and are freely available for download by the public at www.rosettanet.org. A PIP defines the following:

- Roles for the trading partners that use the business process are defined. For example, PIP3A4, Manage Purchase Order, defines the roles "Buyer" and "Seller."
- The business process is defined in terms of the business activities it requires. For example, in PIP3A4, the business activities are Request Purchase Order and Confirm Purchase Order.
- The messages (business documents) exchanged between the roles are called "action" messages. The Request Purchase Order business activity sends a Purchase Order Request Action Message from the Buyer to the Seller. The Seller activates the Confirm Purchase Order Business Activity and sends a Purchase Order Confirmation Action Message to the Buyer, who acknowledges, at the line level, if the purchase order is accepted, rejected, or pending. The PIP defines the content and format of the action messages in XML.
- The sequence in which these messages is sent and the quality of service attributes for the message exchanges, such as time to respond, authentication, and number of retries in case of an unsuccessful message transmission, are defined. The sequence is currently specified using the business process specification language, ebXML BPSS.

Action messages are acknowledged by positive or negative signals: a Receipt-Acknowledgement signal acknowledges that a message has been received and is syntactically validated, whereas an exception signal indicates the opposite. Error codes identify the types of errors.

Currently, PIPs are available for the business areas described in the honeycomb diagram (see Figure 2). For instance, the area named Forecast can be thought of as standing for collaborative forecasting across the supply chain. Clearly, order management is an integral part of such collaborative forecasting. The area named Manufacture stands for material composition and the exchange of semiconductor test data and so on. From the figure, it is clear that Order Management impinges on a number of areas of the supply chain.

RosettaNet divides the entire supply-chain domain into clusters and segments. Each PIP is categorized according to the cluster and segment to which it belongs. For example, PIP3A4 is the fourth PIP in Segment A of Cluster 3 of the RosettaNet classification system.

RosettaNet-Based B2B Integration Solutions

As the discussion in the previous section shows, a key element of RosettaNet-based solutions is the PIP. To deploy a RosettaNet standard or PIP, several activities need to be executed (not necessarily in this order):

1. Choose the PIPs that will be used in transactions. Design private business processes that work in tandem with the public business process defined by the RosettaNet PIPs.
2. Map the PIP messages to the back-end data structures based on the private process. In particular:
   -Develop the data maps from legacy/packaged application transactions into RosettaNet PIP transactions.
   -Change data from one system or application format to another.
   -Edit data to ensure consistency and remove errors.

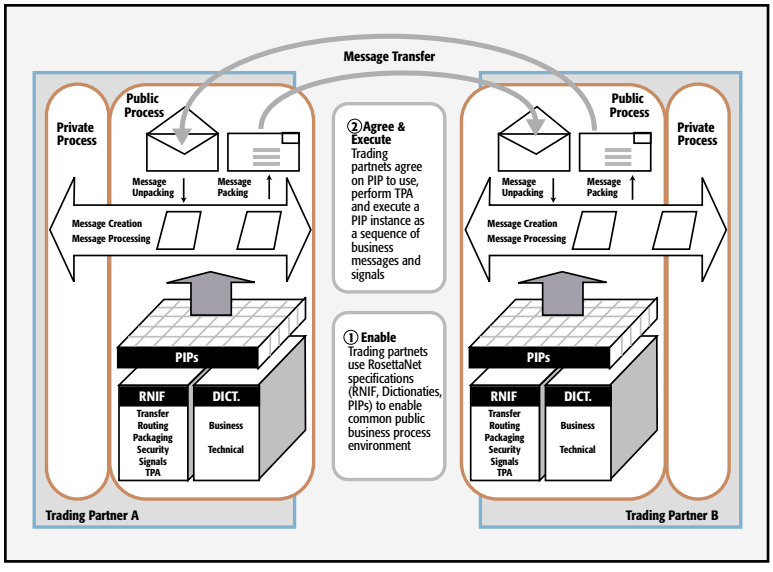


Figure 1 • RosettaNet B2B integration architecture today

Figure 2 • Business processes domains

3. Develop custom applications or modify existing/packaged applications to support the RosettaNet PIP implementation.
4. Test the private and public business process implementation.
5. Manage and train internal staff and all trading partners affected by the RosettaNet PIP implementation.

Data collected by RosettaNet shows that 20% of the integration effort is spent in the design and implementation of the internal processes (step 1). Roughly, 30% of the effort is spent in managing and training the partners and internal staff involved (step 5), while 50% of the effort is associated with the technical implementation the solution requires (steps 2, 3, and 4).

In a typical RosettaNet B2B integration transaction additional components exist. The discussion of each component contains how using XML in a standardized way can reduce cost. Many commercial systems make use of XML extensively in providing these components.

1. ***RNIF connections:*** Implement and establish RosettaNet Implementation Framework (RNIF) connections with the trading partners, and set up appropriate secure network connections across the organizational firewalls.
2. ***Trading partner profile/contract manager:*** An important part of the activity in managing the trading partners is managing their technical profiles on connectivity. This profile includes the contact information, digital certificates, and RosettaNet PIP message controls. The effort in managing the profiles is substantial prior to the beginning of the business transactions with a trading partner, and lessens over time. For example, expiration of a certificate requires changing a profile. XML can be used for defining profiles of trading partners. One available format is ebXML Collaboration Profile and Agreement (CPPA). The use of a commonly accepted format increases portability and reduces errors.
3. ***Execution management interface:*** This interface helps monitor the execution of PIPs, and also provides tracking and tracing functionality of PIP messages from a specific trading partner. For execution monitoring and for reacting when certain conditions are reached, XML-based languages such as RuleML can play a role. The ebXML BPSS may be used in the verification of the PIP choreography described in a PIP specification.
4. ***RosettaNet solution administrator:*** Manageability and configurability of the RosettaNet solution depends on the quality of the facilities available for these activities. This interface can reduce training effort and cost. Currently, there are no XML standards for specifying the management and configurability of RosettaNet.
5. ***Internal application integration:*** Often, private processes are integrated with back-end applications such as ERP and CRM. A good solution for this type of integration considerably reduces the effort required to design and implement the private process (steps 1, 2, and 3). For example, XML-based business process languages, such as Business Process Markup Language (BPML) or Business Process 4 Web Services (BPEL4WS), can be used for defining private processes. The use of such languages increases the flexibility of the implemented business process.

Compliance, Cost Reduction, and Automation

For the successful adoption of RosettaNet across a wide spectrum of businesses, a key requirement is that the solution is capable of implementing and exchanging business information using RosettaNet PIPs in a way that complies with RosettaNet specifications. As more components of public processes are standardized in machine-readable language, implementing and executing RosettaNet e-business standards becomes less costly and accessible to more types of businesses (i.e., SMEs).

RosettaNet took the first step in facilitating compliant automation by creating PIPs based on Document Type Definition (DTD). This step allowed trading partners to implement PIPs in a manner that complies with the business document structure specified in the DTD and additional constraints described in message guidelines. This step also defined the choreography of sending and receiving business documents and their responses. In addition to these PIPs, RosettaNet also created the RosettaNet Implementation Framework (RNIF), which defines the infrastructure for transporting PIP messages. The RNIF standard considerably helped solution providers in supplying compliant solutions that are simple to integrate in many environments, leading to widespread adoption of PIPs today.

The recent move from DTD-based PIPs to XML Schema–based PIPs is the second step to increasing automation and to driving down implementation costs. XML Schema–based PIPs can better express constraints that previously required DTDs and message guidelines in free-formatted text. PIPs are now created with reusable components. The PIP choreography described earlier in a UML activity diagram is now described in XML that conforms to ebXML BPSS. The quality of service attributes of PIP messages are also described in ebXML BPSS instead of a table. All of these improvements result in further automation of the implementation of these PIPs.

The RosettaNet industry consortium has undertaken these efforts to reduce the implementation, execution, and administration costs of B2B integration across the global trading network. To reiterate, RosettaNet-based integration solutions employ XML-based standards to enhance portability and interoperability. Thus, the use of XML-based standards reduce costs without compromising integrity and the interoperability of the solutions. There are arguably other means to reduce the cost of solutions; however, each comes at the expense of interoperability.

Conclusion

When standardized e-business processes are applied, they deliver considerable value, as evidenced by the significant number of RosettaNet production implementations in the marketplace. Over the past several years, RosettaNet has been aggressive in applying XML to more areas than representing business documents to increase machine interpretation and use of the specifications. Increased automation in various steps of the implementation process, and the removal of custom code in execution, results in greater reductions in costs, and thus, promotes greater use and applicability of the RosettaNet standard.

Author Bios

Suresh Damodaran, PhD, is chief technologist of RosettaNet (www.rosettanet.org), on loan from Sterling Commerce. Establishing the architectural strategy of RosettaNet, leading the implementation of the next generation of RosettaNet architecture, and being the final arbiter of all technical issues with RosettaNet standards are among Suresh's responsibilities with RosettaNet.

Neelakantan Kartha, PhD, is a senior software architect at Sterling Commerce. He has a background in knowledge representation and the application of Artificial Intelligence techniques to solve large optimization problems. He has participated in several technical committees within OASIS, representing Sterling Commerce. Kartha holds a PhD in computer science from the University of Texas at Austin.

SURESH_DAMODARAN@STERCOMM.COM
N_KARTHA@STERCOMM.COM

FREE*CD! ($198.00 VALUE!)

Secrets of the XML Masters

Every *XML-J* Article on One CD!

— The Complete Works —

CD is organized into 33 chapters containing more than 450 exclusive *XML-J* articles!

All in an easy-to-navigate HTML format! **BONUS: Full source code included!**

ORDER AT WWW.SYS-CON.COM/FREECD

**PLUS $9.95 SHIPPING AND PROCESSING (U.S. ONLY)*

SYS-CON MEDIA

Only from the World's Leading i-Technology Publisher

WRITTEN BY **KEITH SWENSON**

Streamline Your XML Searches

An index-free approach to managing data

Imagine a customer has hired you to put together a solution for managing a huge quantity of XML information. The firm's team is using XML because it gives them flexibility in how the data is structured. They like the fact that they do not need to specify a given record structure up front, and they can change the XML structure of records whenever they need to. Still, the question remains, "How do you manage and search for records?"

One choice, of course, would be to put the data into a relational database. This approach is not always convenient because you need to decide ahead of time what the schema of the tables will be. Such a database will need to be managed, and any changes in requirements will demand changes in the schema as well as migration of the data. A second option is to put the data into the XML column of a relational database. This will work for moderate to large quantities of data, but what if you have truly huge amounts of data?

A New Approach

Now there is an alternative to a traditional database for storing and searching huge quantities of XML encoded data, and it is based on the use of a grid paradigm that eliminates the need for indices and provides the ability to guarantee the search response time.

The concept of searching XML data without using an index may well sound counter-intuitive. After all, we learned in Computer Science 101 that an index is a way of making searches go faster. The accepted practice is that if your query performance times are taking too long, reexamine your schema and figure out how to construct the right index. In practice, judicious index construction can often make certain queries run much faster. However, several challenges arise when the number of records extends into the terabyte range.

Consider first that, no matter how carefully a database is coded, there will be an n^2 (or higher) order component to the time needed to maintain the index. Whenever a record is added to the database, the proper place in the index must be found, and an entry must be added. Whenever a record is removed, the index must be similarly modified to remove the entry. Of course, the ability to search on any part of an XML record then requires that every part must be indexed. The amount of extra processing involved to keep the indices up to date can be very significant if the index is large.

The second effect of an index is one of partial results. If the query involves several different fields, and if one of the values being searched for is commonly found in the data, the result set for that part of a query could be huge as well. The "and" operation in a search query effectively can cause the database to generate two huge partial result sets and then find the intersection of the large set – all in order to end up with what might be a relatively small final result set. The partial result sets must be managed in and out of memory, which can demand a great deal of processing, and the index itself can take up a lot of disk space requiring even more I/O operations.

It doesn't stop there. Once the result set is found from the indices, the database must go back to the disk and read those records into memory, causing yet more random I/O. It turns out that this random access nature of retrieving data contributes to slowing result times. A disk can be read sequentially about 10 times faster than reading random sectors because when reading sequentially, there is no need to seek and reposition the head across the disk.

The last effect of the index is to make the data coupled. An index by its nature must be an ordered set of references to all the records. An index over part of the records is just not as useful for searching as a complete index.

The solution is remarkably simple: eliminate the challenges of using indices by starting with the idea that data can be searched without an index. One benefit of this is that nothing has to be set up in advance. The XML is written to files on a disk, so it can be searched immediately.

At the same time, offer the ability to add and search XML data of any record structure, thereby eliminating the need to convert or prepare the XML records. If over time the application is modified to handle additional data, the new records can be added to the database without changing any of the existing records.

Performance

Simplicity and flexibility are nice, but how do you actually get the required performance? Two techniques are being employed today.

The first is permitting data to be spread across any number of low-cost Linux blade servers by having a controlling service as well as search services deployed on any number of Linux blade servers. Since there is no index, there is no need for any of these servers to know anything about the other servers; each blade server is given its part of the data, and it searches that part alone. In this scenario, the controlling service is able to collect the search queries and submit them in parallel to each of the search

AUTHOR BIO

Keith Swenson, chief architect and director of development, began his tenure at Fujitsu in 1991, developing TeamWARE Flow. He returned to Fujitsu Software Corporation in 2002 to direct the development of the Interstage family of products. Keith is currently working on standards such as WS-CAF and ASAP. He holds both a master's degree in computer science and a bachelor's degree in physics from the University of California, San Diego.

servers. It then collects all of the results from the search servers, merges them together, and returns them to the requester.

One of the great advantages of this approach is that new search servers can be added at any time. If the response time gets longer, deploy more servers and the entire search operation speeds up again. There are no dependencies between the search servers, so there is almost no limit to how much this can be scaled. More important, this property can be used to guarantee response time. As the quantity of data grows, the amount of processing power can grow, guaranteeing that the response time remains constant.

At the same time, breaking the data out into many search servers would still leave the response time unacceptable if it were not for another technique. This is the use of the controlling service to collect hundreds or thousands of queries to be searched at the same time, providing the ability to search for multiple queries at the same time without slowing down the search.

Imagine you have 20 queries looking for home listings in New York and another 20 queries looking for homes in Texas. As soon as a record is tested and found to be about a house in New York, the search mechanism continues processing on the first 20 queries, but the queries for houses in Texas can be ignored for this record, and cause no additional overhead. The search through a large amount of data might take 5 to 10 seconds. However, the ability to return 1,000 result sets from a single pass through the data means you can maintain very high speed, even without an index.

Here's how it works. When a record is found that matches a query condition, the entire record is already in memory, so it can immediately be added to the result set without additional I/O. This adds up to significant I/O savings. Since there is no index to bring in and out of memory, there are no huge partial result sets to manage, and there is no need to go back and retrieve the record contents separately. Instead, the disk is read sequentially, which is the fastest way to read a disk.

Additionally, XML records can be added and removed at any time. There is none of the overhead associated with updating an index, which in traditional index-based solutions can demand a significant amount of processing in certain situations. Therefore, when a record is added, it becomes immediately available for searching by the next query. Similarly, records can be removed quickly and easily.

The combined techniques for streamlining and speeding XML searches can be used in many situations. However, the greatest benefit comes when there are a large number of users requesting queries at the same time – making it ideally suited for a large Web site or a Web service. Furthermore, if the Web site is collecting records, it is quite convenient that the records only need to be structured into XML for storing in the database, since this is something a Web application or a Web service can do easily.

There also are significant performance advantages for Web applications that are mostly searching and displaying data, with some need for adding and removing records. Examples of this include:

- Large classified ad sites where people can add new advertisements while others search for ads that match certain criteria.
- A Web forum where many people are accessing, adding messages, reading other messages, and searching for messages on a particular topic.
- A large trading or bartering site.

All these uses have some common features. They have a large amount of data; they add and remove records but typically don't do a lot of record updates; and there are large numbers of people searching for records.

•••

The real benefit of this index-free approach is that it's easy to manage. There is no need to set up a schema beforehand. The record structure can be changed at any time, or it can be mixed with any other combination of records. There is no need to ever migrate from one schema to another. Additionally, it's easy to add more processing power to improve or maintain performance. Moreover, this approach does not require extensive IT resources to manage the data.

The fact that patented technology around these approaches is commercially available today means the power to quickly and effectively search huge stores of XML data is now available to small, medium, and large enterprises alike.

KSWENSON@US.FUJITSU.COM

WRITTEN BY **MIKE SEGURA**

Using XML to Deliver Critical Messaging Services

XML-based technology offers a proven multichannel emergency notification solution

We all know that in today's threat-conscious world, communication is more than a convenience.

To protect their organizations and the public in the event of a natural disaster, terrorist strike, or other significant threat, businesses and governments have been forced to reassess their ability to monitor events, notify key constituencies, and provide accurate and relevant information. But creating a reliable warning and communication system has been a challenge.

As we learned during both the 9-11 attacks and the Eastern Seaboard blackouts, when both cell phones and LAN lines were largely unavailable, traditional wired and wireless systems are highly vulnerable to both natural and man-made calamities. These events underscored for all of us the need for a more reliable emergency event infrastructure.

Anywhere, Anytime Emergency Notification

Now a new convergence of technologies – most notably the wide proliferation of the Internet, mobile communications, and speech recognition software – is driving the deployment of proactive "anywhere, anytime" emergency notification systems. This new approach to emergency notification combines traditional Web and wireless messaging with the power of interactive voice alerts. One key aspect of this new approach is the use of XML and VoiceXML to deliver critical emergency notifications. When layered on top of traditional applications, an XML- or VoiceXML-based approach can spread the responsibility for emergency notifications across both wireline and wireless services in a true multimodal fashion.

The XML Approach

The XML-driven approach combines sophisticated monitoring and detection with advanced push technology to provide reliable and cost-efficient emergency event notification for the government sector.

To appreciate the impact these technologies may have on the future of emergency notification, let's take a closer look at exactly how XML messaging works.

> "A system that gathers instant information on threats and emergencies and disseminates warnings to both key decision-makers and the general public could save thousands of lives"

XML messaging

XML-oriented systems can now be deployed to meet a variety of emergency notification requirements. Some of the latest systems allow reliable and detailed alerts to be forwarded to virtually any constituent, from government and public safety officials to senior executives, security and facility managers, health and emergency service providers, disaster recovery teams, and others.

These types of alerting technologies allow end users to query any type of content or source and to forward notifications in customized formats to virtually any wireless, Internet-enabled, or custom device, including voice telephones, fax, pager, e-mail, PDA, or SMS-text messages.

Using an open approach to monitoring, detection, and notification (see Figure 1), the XML-based approach gives organizations great flexibility in the selection and deployment of notification solutions while preserving the value of existing infrastructure investments.

Monitoring is a crucial aspect of any notification infrastructure, and these systems can initiate the process by continually checking the status of one or more content resources, which can be an XML or a VoiceXML file. To ensure reliability even during a major event, the system can check content resources on the Internet via HTTP or HTTPS, as well as at selected backup locations.

An alert engine then places a call request to a media server's messaging manager, from which an outbound application writes a Java servlet request to an internal database for confirmation and verification. A voice alert gateway receives the call request from an HTTP server and the XML or VoiceXML document from the alerts engine, and performs outbound dialing to the person or organization to be notified. The notification is then played and verified, or attempted again if the call reaches a voicemail system or busy signal. An automated application then posts the results of the call to confirm the results of the notification.

This XML-based system supports true multichannel message delivery, message receipt acknowledgement reporting, and notification retry across channels until the message is successfully delivered. Communications are

AUTHOR BIO

Michael Segura is responsible for strategy and market development of packaged applications for Intervoice. He has 13 years of experience managing business units within the telecommunications, energy, and consulting industries. He holds a bachelor of business administration from Texas A&M University and has completed executive marketing management programs from Duke University and University of North Carolina. Michael also holds a CPA certification.

configurable by role, message, and content, and sophisticated Web- and voice-based notification services can be securely managed by administrators who do may not have a great deal of technical expertise.

Deploying XML solutions

For systems administrators, the XML and VoiceXML solutions offer great promise for a more practical and reliable means to create emergency notification offerings for various public and private applications. These applications can be layered as a loosely coupled service and provisioned as an ASP offering and delivered via narrowcast alerts on either a national or regional basis.

Systems administrators can leverage the flexibility of this open, standards-based approach to formulate and deploy emergency notification solutions that incorporate a number of advanced features and functions.

A system can be configured, for example, to grant or deny permission to send messages or administer alert lists based on preset authorizations. Multimodal messages can be quickly and easily targeted to user-defined groups or individuals, with selected recipients receiving appropriate notifications under preselected conditions.

The Web-based nature of this approach also allows administrators to manage all emergency notifications at any time and from any Internet-enabled connection. Personalized Web portals allow users to create notification lists and groups, manage permissions and messages, and modify portal content online and in real time with no IT or Webmaster involvement.

When offered to public agencies or companies, this type of emergency notification would be provided on a subscription basis, with cellphone-style pricing for basic services and additional charges for use above a preset level. This platform-based, XML-driven approach allows emergency management personnel to create scenario-specific communications plans that are executed and documented automatically in the event of an emergency.

Future Applications

One practical use of this technology is in the public education system. A school could use the Web or voice to register the contact information of administrators, teachers, and parents, who would then receive selected information in the event of an emergency situation at any school facility. The same type of emergency notification system can and likely will be deployed to serve hospitals, local governments, universities, corporations, and other organizations.

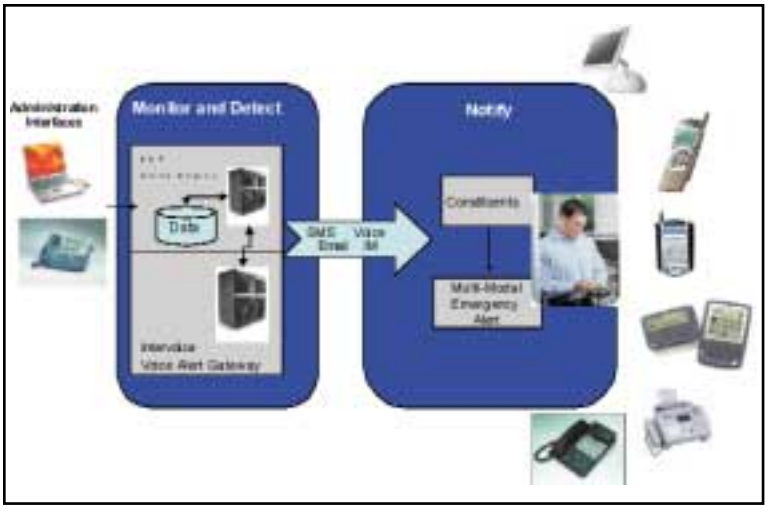


Figure 1 • Emergency event notification system

In a 2002 report on the need for emergency notification, an expert panel of emergency managers recommended the creation of a high-technology national warning system. A system that gathers instant information on threats and emergencies and disseminates warnings to both key decision-makers and the general public could save thousands of lives in a major catastrophic event.

An XML-based solution – by providing fast and reliable notifications through a variety of delivery channels – can give people the warning they need in a serious emergency.

MIKE.SEGURA@INTERVOICE.COM

WRITTEN BY **DAVID S. LINTHICUM**

Understanding Information Transformation

Anticipate the extraordinary

The transformation layer is the "Rosetta stone" of the system. It understands the format of all information being transmitted among the applications and translates that information on the fly, restructuring data from one message so that it makes sense to the receiving application or applications. It provides a common dictionary that contains information on how each application communicates outside itself (application externalization), as well as which bits of information have meaning to which applications.

Transformation layers, such as those that process XML-based messages (e.g., XSLT), generally contain parsing and pattern-matching methods that describe the structure of any message format. Message formats are then constructed from pieces that represent each field encapsulated within a message. Once the message has been broken down into its component parts, the fields may be recombined to create a new message.

Most integration servers can handle most types of information, including fixed, delimited, and variable. Information is reformatted using an interface that the user integration server provides, which may be as primitive as an API, or as easy to use as a GUI.

There are a few aspects to the notion of transformation.

Differences in Application Semantics

Accounting for the differences in application semantics is the process of changing the structure of a message, and thus remapping the structure and data types so that it is acceptable to the target system. Although it is not difficult, application integration architects need to understand that this process must occur dynamically within the integration server.

This process can be defined within the rules-processing layer of the integration server by creating a rule to translate data dynamically, depending on its content and schema. Moving information from one system to another demands that the schema/format of the message be altered as the information is transferred from one system to the next.

Although most integration servers can map any schema to any other schema, it is prudent to try to anticipate extraordinary circumstances. For example, when converting information extracted from an object-oriented database and placing it in a relational database, the integration server must convert the object schema into a relational representation before it can convert the data within the message. The same holds true when moving information from a relational database to an object-oriented database. Most integration servers break the message moving into their environment into a common format and then translate it into the appropriate message format for the target system.

Differences in Content

Related to the concept of accounting for the differences in application semantics, accounting for content changes is another important aspect of transformation. In short, it's the reformatting of information so that it appears native when sent to a target system. The information needs to appear native, requiring that changes be made to source or target systems.

Although many formats exist within most application integration problem domains, we will confine our attention, for the purposes of this manifesto, to the following:

- Alphanumeric
- Binary integers
- Floating point values
- Bit fields
- IBM mainframe floating points
- COBOL and PL/I picture data
- BLOBs

In addition to these formats, there are a number of formatting issues to address, including the ability to convert logical operators (bits) between systems and the ability to handle data types that are not supported in the target system. These issues often require significant customization in order to facilitate successful communication between systems.

In data conversion, values are managed in two ways: carrying over the value from the source to the target system without change, or modifying the data value dynamically. Either an algorithm or a look-up table can be used to modify the data value. One or more of the source application attributes may use an algorithm to change the data or create new data.

Algorithms of this type are nothing more than the type of data conversions we have done for years when populating data warehouses and data marts. Now, in addition to using these simple algorithms, it is possible to aggregate, combine, and summarize the data to meet the specific requirements of the target application.

When using the look-up table scenario, it might be necessary to convert to an arbitrary value. "ARA" in the source system might refer to a value in the accounts receivable system. However this value may be determined, it must be

AUTHOR BIO

David Linthicum is the CTO at Grand Central Communications (www.grandcentral.com), and a leading expert in the application integration and open standards areas. David is the author of eight books, including his latest, Next Generation Application Integration.

checked against the look-up table. Integration servers may use a currency conversion table to convert dollars to yen, which may be embedded in a simple procedure or, more likely, in a database connected to the integration server. The integration server may also invoke a remote application server function to convert the amount.

The application integration architect or developer may encounter special circumstances that have to be finessed. The length of a message attribute may be unknown, or the value may be in an unknown order. In such situations, it is necessary to use the rules-processing capability of the integration server to convert the problem values into the proper representation for the target system.

Abstract Data Types

Transformation mechanisms also need to support abstract data types (ADTs), allowing different representation of data and behavior to meet the requirement of the application integration scenario.

ADTs provide a mechanism with a clear separation between the interface and implementation of the data type, including the representation of the data, or choosing the data structure, and the operations of the data

The interface with the abstract data type is created through an associated operation. What's more, the data structures that store the representation of an abstract data type are invisible to the integration view. The ADT also includes any operations, or algorithms, contained with the ADT.

The internal representation and executions of these operations are changeable at any time and won't affect the interface to the ADTs. Thus, a completely different representation is possible for sets storing information in the ADT.

Having said all that, ADTs consist of:

- An interface, or a set of operations that can be performed
- The allowable behaviors, or the way we expect instances of the ADT to respond to operations.

The implementation of an ADT consists of:

- An internal representation of data stored inside the source or target system's variables
- A set of methods implementing the interface
- A set of representation invariants, true initially and preserved by all methods

Information Routing

In addition to transformation, information routing is another core feature that provides a mechanism to move information from system to system. We have a few scenarios that apply, including:

- One to one
- Many to many
- Many to one

It's important that your integration technology can route information from many systems to many systems, as well as split information coming from one system to be sent to multiple targets, and combine information coming from many systems for a single target. While this sounds simple, the application of the mechanism is far from simple. We must introduce the notion of behavior to operate on this information.

Intelligent Routing

Intelligent routing, sometimes referred to as flow control or content-based routing, builds on the capabilities of both the rules layer and the semantic transformation layer. An integration server can "intelligently route" a message by first identifying it as coming from the source application and then routing it to the proper target application, translating it if required.

For example, when a message arrives at the integration server, it is analyzed and identified as coming from a particular system and/or subsystem. Once the message is identified and the message schema is understood, the applicable rules and services are applied to the processing of the message, including transformation. Once the information is processed, the integration server, based on how it is programmed, routes the message to the correct target system. This all takes place virtually instantaneously, with as many as a thousand of these operations occurring at the same time.

Filters

In addition to intelligent routing, it's important to provide the notion of filtering, as well. In the world of application integration, filters are software subsystems that are able to analyze content and selectively leave out specific information based on content or, perhaps, source or target information.

Filters are important to application integration due to the complexity of information coming from source systems and the need to simplify that information before it's processed in the integration server or sent to the target system. The notion of filtering also relates to transaction controls.

DLINTHICUM@GRANDCENTRAL.COM

TIBCO Software Extends Business Activity Monitoring with TIBCO OpsFactor

(Palo Alto, CA) – TIBCO Software Inc., an enabler of real-time business and an independent business integration software company, has announced the availability of TIBCO OpsFactor 1.0. This software gives users superior visibility into business operations that are being orchestrated by TIBCO's business integration software through the creation of dashboards that visually display real-time information, operational metrics, and key performance indicators that reflect business goals and conditions.

TIBCO OpsFactor helps operational business and IT managers clearly understand the performance and impact of processes so they can be proactive in adjusting resources to support business objectives. It leverages TIBCO BusinessWorks, their real-time business integration platform, as the source for process content.

By incorporating its own palette of monitoring and measurement tasks into the TIBCO BusinessWorks modeling environment, TIBCO OpsFactor lets users add activity sensors to any step or sub-step in a process, define threshold levels so exceptional events and conditions can be identified, and configure "rollup rules" so related metrics can be aggregated and assembled into meaningful information and alerts.

TIBCO OpsFactor is available now and with pricing starting at $125,000 per server.

www.tibco.com

Digital Evolution Extends Netegrity TransactionMinder

(Santa Monica, CA) – Digital Evolution, a provider of Web Services Management and security software, has announced the launch of its Web Services Management solution package for Netegrity customers. This solution adds service-oriented architecture–enablement and Web Services Management capabilities to Netegrity TransactionMinder technology.

The Digital Evolution Web Services Management solution package for Netegrity customers leverages the SOA enablement and Web Services Management capabilities of the Digital Evolution Service Manager and the SOA experience and expertise of Digital Evolution's professional services team to offer Netegrity customers a complete solution for Web Services Management and Security.

www.digev.com

Sarvega Adds Comprehensive Business Integration Capabilities

(Chicago) – Sarvega, Inc., a provider of high-performance XML networking solutions, has announced that the Sarvega Business Integration Module as an optional enhancement to the Sarvega XML Speedway Accelerator and XML Guardian Security products is now available. The Sarvega Business Integration Module will allow Sarvega customers to handle a broad spectrum of non-XML formats in conjunction with their Web services applications, both within and beyond the corporate firewall. This module gives Sarvega's customers a high degree of flexibility to meet their business requirements.

The Sarvega Business Integration Module enables state-of-the-art collaboration and data exchange between Web services, existing applications, databases and non-XML formats. Using an easy-to-use wizard, the module enables customers to define simple or complex data transformations with XML and non-XML content across multiple transport protocols without expensive hand coding.

www.sarvega.com

Liberty Bank Chooses Sigaba for Secure Messaging

(San Mateo, CA) – Secure Data in Motion, dba Sigaba, a provider of secure message management solutions, has announced that Liberty Bank has selected the Sigaba Secure Email solution for its strong security and robust capabilities and features, including ease-of-use and encryption of messages in accordance with corporate policies.

Sigaba enables the bank to send and receive encrypted e-mail without requiring additional software for end users. The Sigaba secure messaging suite includes Sigaba Secure Email, Sigaba Secure Instant Messaging (IM), and Sigaba Secure Statements. Solutions have been tested and certified to provide the strongest level of security with the highest level of scalability and audit capabilities. Sigaba secure messaging interoperates with all major authentication mechanisms for seamless communications between third parties.

www.liberty-bank.com, www.sigaba.com

City of Canton Achieves Real-time Information Sharing with Fiorano ESB

(Los Gatos, CA) – Fiorano Software has announced that the City of Canton, Georgia, has created real-time access to information for meeting the increasing demands of its citizens with integration technology from Fiorano Software, a leader in standards-based business process integration and messaging infrastructure technology.

In keeping with its mission to provide services and support to its citizens as well as businesses, the City was continuously looking at ways to improve its IT infrastructure. The City had several disparate information systems that required manual relation and coordination. Application integration was seen as the only viable solution to increase productivity and data validation.

The Fiorano Business Integration Suite empowered business users by enabling them to create new and efficient business processes and make simple changes quickly, leading to tremendous time and cost savings. Fiorano ESB provides a balance of openness and standardization within a productive framework. The City of Canton achieved significant qualitative and quantitative benefits by implementing the Fiorano solution.

www.fiorano.com/casestudies/cs_coc.htm

Reactivity Delivers Solution to Secure SAP, PeopleSoft, and Oracle Web Services

(Belmont, CA) – Addressing the potentially acute Web services security vulnerabilities of packaged enterprise software, Reactivity, Inc., has announced its flagship Reactivity XML Firewall. This product eliminates the complexity of integrating enterprise application Web services, reducing deployment times from weeks or months to minutes, while providing a centrally controlled Web services security policy with optimized enforcement.

By decoupling security from application development, the Reactivity XML Firewall eliminates the time-consuming and costly process of custom security programming, infrastructure integration, and security policy maintenance for each XML Web service application.

www.reactivity.com

Think .NET development is more productive than J2EE?

Think **again**.

Think **better**.

ThinkCAP™

ClearNova's ThinkCAP is a comprehensive application platform that simplifies and accelerates the development and maintenance of J2EE-based business applications by 50 to 70%.

ThinkCAP's visual & intuitive designers bring high productivity to business developers (those with VB or PowerBuilder-like skills), content owners, and administrators while allowing J2EE architects & programmers to leverage its component infrastructure and build business logic using the tools and approaches they prefer. ThinkCAP utilizes existing infrastructure, web services, legacy systems, and business applications.

ThinkCAP saves organizations time and money—and lowers project risks. Applications are written faster and require less maintenance. Project teams utilize in-house skills and require less training. Existing infrastructure and application servers are leveraged.

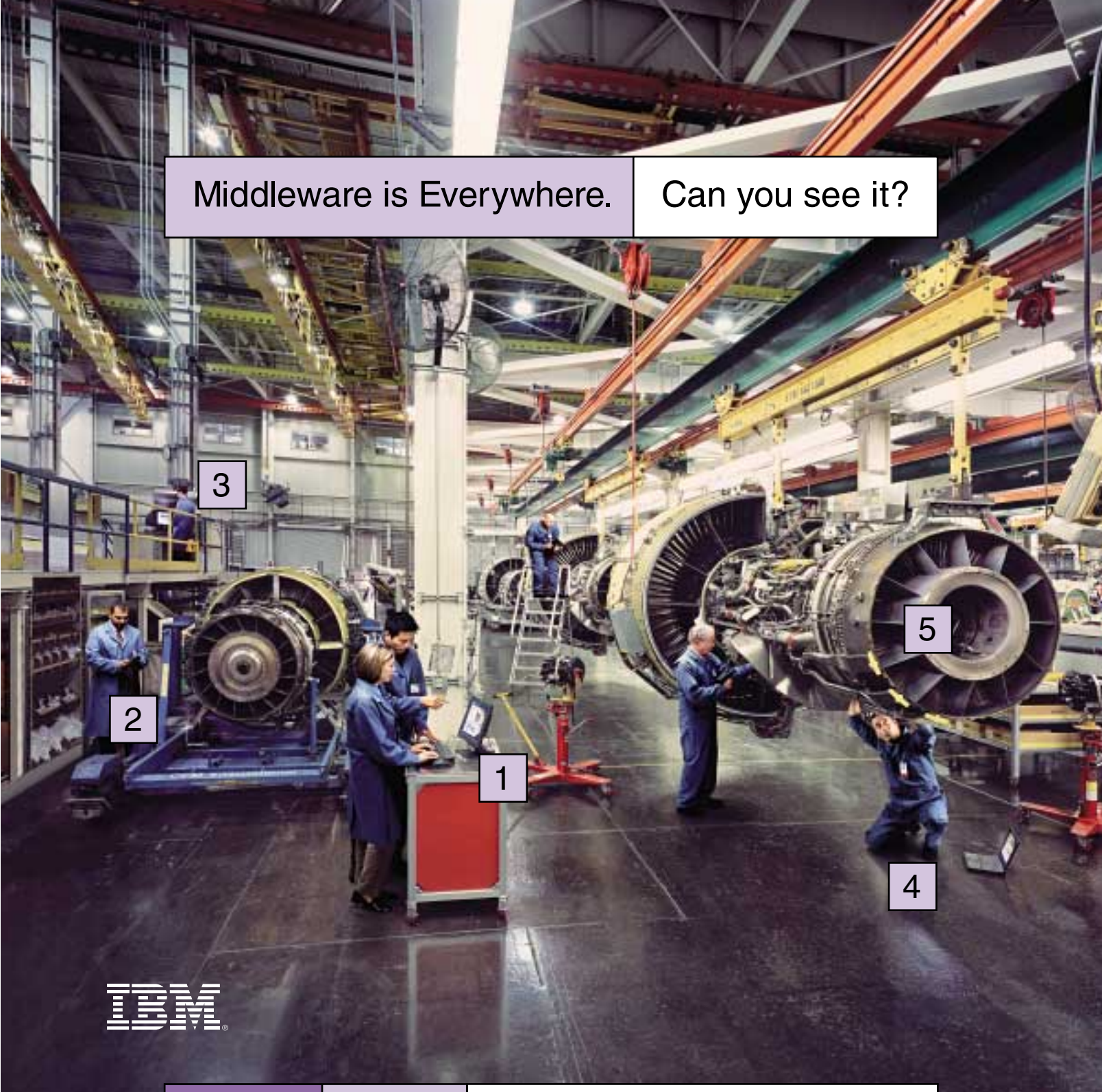
With ThinkCAP you can build quality applications faster.

Learn more about ThinkCAP at www.clearnova.com/thinkcap

CLEARNOVA

APPLICATION DEVELOPMENT • SIMPLIFIED • ACCELERATED

- Highly Visual Development Environment
- MVC Framework with Page Flow & Actions
- Advanced Data Aware Controls:
  Forms, DataViews, Queries, Navigations
  Workflows, Graphs, Treeviews, Grids, Tabs
- Smart Data Binding™ to data, objects, XML, sessions, or requests
- Browser & server-side validation
- Visual unit testing with RapidTest™
- Service Flow Designer aggregates Web Services, EJBs, XML, and POJOs
- Content Management engine & tools
- Supports .NET clients
- Integrated, seamless security
- Use any app server or 3rd party tool



IBM

WebSphere.

Key

1. **New design already tested.**
2. **Suppliers already linked.**
3. **Procurement already automated.**
4. **Blueprints already updated.**
5. **Engine all ready for takeoff.**

MIDDLEWARE IS IBM SOFTWARE. Powerful software like IBM WebSphere.® Using an open and scalable foundation, WebSphere software lets you swiftly respond to change. On demand. Applications are easily updated, tested and deployed. Lead time is shortened. And everything clicks, regardless of platform. WebSphere delivers it all. On the money. On demand.

e **business on demand**™ at **ibm.com**/websphere/middleware