SYS-CON

NOW INCORPORATING XML JOURNAL

WebServices JOURNAL

.NET J2EE XML

December 2004 Volume 4 Issue 12

RETAILERS PLEASE DISPLAY UNTIL JANUARY 31, 2005

$6.99US $7.99CAN

12>

0 71486 03420 9

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

Nothing beats our racks Absolutely nothing

CARRIER CLASS DATA CENTERS

Highly Secure Guarded Facility
24/7/365 Network Operations Center
24/7/365 Technical Support
Redundant Conditioned Air Systems
Redundant Fiber Entry Points
Multiple Uninterruptible UPS Systems
Multiple 1250 KW Generators Onsite
VESDA Early Warning Smoke Detection

Robert Marsh, Head Surfer

START YOUR OWN WEB HOSTING BUSINESS TODAY!

from **$299***

Instant Activation!

Dedicated Server

Dual Xeon 2.4 GHz
2 GB RAM • 2 x 73 GB SCSI HD
Remote Console • Remote Reboot
1700 GB Monthly Transfer Included

Over 20,000 Servers!

1-800-504-SURF | ev1servers.net

PLESK7 RELOADED
Preferred Control Panel

IP Compliant. Price subject to change. Quantities Limited.
*Per month. Set-Up fees apply. See web site for complete details.

XSLmaker

XSL XML CSS Xpath

The Smart Way to Transform XML to Web Pages

1. Define XML
2. Drop XML on Web Page
3. XSL and CSS are generated

Done.

100% WYSIWYG

Transform XML => Web pages

Features

Visual tool with Canvas	✓
Auto-generates XML tree based on sample XML data	✓
Drag-n-drop from XML tree directly onto the Canvas	✓
Generates XSL code & generates CSS code	✓

Advanced Features

Merge multiple XML sources	✓
Include live XML/RSS feeds in Web page	✓
Include other Web pages in Web page	✓
Use other Web languages with XSL in Web page	✓
Supply run-time input parameters to XSL Web page	✓

XSLmaker

The only comprehensive XSL development platform for efficient making of professional Web pages and Web applications from XML data. Easy and intuitive.

FREE trial. Download at www.xslmaker.com

DECEMBER 2004 Volume 4 Issue 12

WebServices JOURNAL .NET J2EE XML

XML JOURNAL

Inside WSJ

Achieve higher quality

in less time, with fewer resources.

Parasoft® Automated Error Prevention (AEP) Products:

Specifically designed for under-staffed, over-committed development organizations like yours.

Parasoft AEP Products ensure that comprehensive error prevention practices are applied consistently and uniformly across your entire team.
By automating compliance to a unified set of testing and coding standards, Parasoft AEP Products enable every team member to follow the same best practices and the same error prevention techniques – including coding standards analysis, unit testing, code review and regression testing.

Automated testing and standards compliance for any team configuration.
Parasoft AEP Products automate error prevention for Java, C/C++, Web Services, the Microsoft® .NET Framework, Embedded Development, Web Development, Database Development and more.

For details go to: www.Parasoft.com/AchieveQuality

Call: 888-305-0041 x3307 Email: AchieveQuality@Parasoft.com

PARASOFT®
We make software work.™

Founded in 1987, Parasoft's clients include IBM, HP, DaimlerChrysler and over 10,000 companies worldwide.

Availability
Parasoft AEP Products are available on Linux, Solaris, and Windows NT/2000/XP.

Contact Parasoft for details and pricing information.
Parasoft Corporation
101 E. Huntington Dr., 2nd Flr.,
Monrovia, CA 91016

HostMySite.com™

Built for ColdFusion Pros
by ColdFusion Pros

plans from **$8.95** / mo.

FREE Domain Name*
FREE Setup
FREE 2 Months

- 24 / 7 / 365 Phone Support
- 99.9+% Uptime
- Macromedia Alliance Partner
- "Full Control" Panel
- CFMX 6.1 or CF 5.0
- SQL Server 2000 or 7.0
- Custom Tags Welcome

Visit www.HostMySite.com/mxdj for:

2 Months Free

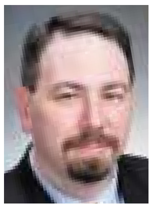
and FREE Setup on Any Hosting Plan*

"When it comes to ColdFusion hosting, HostMySite.com rules them all!"
James Kennedy
mbateam.com

*offer applies to any annual shared hosting plan

call today! **877•248•HOST** (4678)

The Tool I Really Want for Christmas



WRITTEN BY

SEAN RHODY

As I get ready to celebrate the holidays this year, I spent a little time reflecting on what I would like from the various Web services vendors. While a case of scotch was definitely tempting, what I really want is a better toolset.

To date, most of the programming tools available for building Web services have focused on the ultimate deployment platform. In other words, they've looked at the end implementation language first and tuned their work towards that. So, for example, Visual Studio aims at C# or VB, WebSphere Application Developer aims at WebSphere, and BEA WebLogic Workshop targets WebLogic. Borland, who also makes generally excellent tools, has built support into JBuilder for Web services, but again, targeting Java.

Now don't get me wrong. We need tools that will allow us to focus our energy on implementing Web services in particular languages. After all, a service is worthless unless you can actually invoke it. So these tools are all very important parts of the overall Web services development strategy.

At the same time, they've all focused on the ***only*** platform/language/idiom-specific aspect of the entire Web services spectrum – the implementation. Every other aspect of Web services is vendor, language, and platform neutral. And there are no tools for that.

I want a tool that makes it simple for me to first define my Web service, then pass it over to whatever development environment I choose for implementation. And when you look at it from that perspective, you see what I mean.

I want something that makes it easy to define the service so that I can visually create a WSDL file. I want something that understands more than just SOAP, WSDL, and UDDI – I want the tool aware of and capable of managing and integrating WS-Orchestration, WS-Security, BPEL, and all the other standards. I want it to be able to conform a previously defined service to these standards (so that I can make something simple more complex, like adding transactionality) without having to burn incense and read three product manuals.

Ideally, we can use this for more than just a better "developer's" tool as well. Much of the software development process these days comes long before the actual coding – things like requirements gathering and analysis, design, and modeling all have their place in the modern arena of software development. Elements of each of those domains should also be part of this toolset – it should be possible to do an enterprise API with this tool, and then get more granular and refined as we move ideas closer to reality.

Of course, the ultimate goal of this tool should be to produce a Web service stub for many different target platforms. As I mentioned, a service is worthless if no one can use it because it can't be implemented. Likewise, all of this language-independent work is meaningless if it can't be directly applied to a particular implementation environment. That was always a problem with the modeling tools in the past – round-trip engineering always came last, and always had bugs. Fortunately, in this case most of the information we provide, right up to the actual code, resides in XML, which can be modeled and transformed quite easily.

So there's my hope for a happy New Year – a tool that makes it easier to do the front-end work – the design and definition of Web services, before having to actually code them. Of course, I'll take that case of scotch if it comes too! Happy holidays.

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

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

WebServices JOURNAL .NET J2EE XML

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

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

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

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

SYS-CON.COM
VP, INFORMATION SYSTEMS
Robert Diamond robert@sys-con.com
WEB DESIGNERS
Stephen Kilmurray stephen@sys-con.com
Matthew Pollotta matthew@sys-con.com

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

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

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

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

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

SYS-CON MEDIA

Enterprise Service Bus Products

WRITTEN BY
JAMES KOBIELUS

Enterprises are demanding integration tools that enable more seamless interoperability across diverse integrationware paradigms: old, current, and emerging. The industry has coined a new three-letter acronym – enterprise service bus (ESB) – that speaks to the dream of standards-based integration of legacy middleware with the new world of Web services.

ESB denotes an emerging segment of the middleware market. ESB middleware vendors include Cape Clear Software, Fiorano Software, IBM, IONA, Sonic Software, Systinet, TIBCO Software, and webMethods. ESB products bring together the legacy world of vendor-proprietary, message-oriented middleware (MOM) protocols with the growing "WS-*" stack of vendor-independent Web services standards. Fundamentally, ESB is "MOM++."

ESB products support reliable, guaranteed messaging, which has traditionally been the core MOM functionality. They leverage Web services standards and interface with established reliable-messaging MOM protocols such as IBM WebSphere MQ, TIBCO Rendezvous, and Sonic Software's SonicMQ. Common features of ESB products include the ability to bridge heterogeneous MOMs, wrap MOM protocols with Web Services Description Language (WSDL) interfaces, and tunnel Simple Object Access Protocol (SOAP) traffic over MOM transports. In addition, most ESB products support direct, peer-to-peer interactions among distributed applications, in addition to or in lieu of hub-and-spoke interactions through intermediaries such as integration brokers.

The evolution of MOMs won't be complete until the Web services stack includes ubiquitous standards for reliable messaging, event notification, and pub/sub. The Web services stack still lacks finalized, profiled, widely adopted standards in all of these areas, although it has strong contenders in Web Services Reliable Messaging (WS-RM), WS-Eventing, and WS-Notification (though it appears increasingly likely that the industry will produce a convergence draft that incorporates the overlapping functionality of WS-Eventing and WS-Notification). Some ESB vendors – most notably, Systinet – have already implemented these specifications in their products but most still have the specifications on their futures roadmaps.

ESB vendors are seriously committed to implementing Web services reliability standards – such as WS-RM – as they emerge. One fortunate consequence of this trend will be continuing reductions in vendor opportunities to lock customers in through proprietary specifications. Already, vendor-proprietary MOM application programming interfaces (APIs) have taken a backseat to the common Java API – Java Message Service (JMS) – and all ESB vendors are supporting further abstraction through the ability to wrap proprietary MOMs as Web services via WSDL and SOAP.

By the end of this decade, Web services–based MOM functionality will become the dominant ESB approach for reliable messaging. We call this approach "MOM abstraction": the ability to use Web services standards and specifications to provide MOM-grade reliability services in lieu of, or to bridge between, pre-Web services MOM environments. Increasingly, Web services will have native reliability features through WS-RM and other SOAP extensions. As Web services–based MOM abstraction takes hold in the market, enterprises will gradually de-emphasize traditional MOMs, slowly pushing them out of the integration picture.

However, until WS-RM and other Web services reliability protocols are widely adopted, enterprises will continue to rely on traditional MOMs and other established middleware approaches to provide the end-to-end reliable messaging that the Web services stack (in its current incomplete state) can't yet support. We expect that, over the next 1–2 years, consensus, profiled Web services standards will emerge for reliable messaging, event notification, pub/sub, and other robust functionality. It will take 2–3 more years before these are implemented widely, profiled by the Web Services Interoperability Organization (WS-I), and tested for interoperability across multivendor environments.

Once WS-RM, WS-Notification, and other key Web services reliability standards are universally implemented, the need for vendor-proprietary ESB protocol stacks on nodes will start to wither away. Vendors such as IBM, Microsoft, and BEA embed ESB functionality in their application servers. Increasingly, enterprises will access ESB functionality within every Web services application platform that they buy, thereby taking advantage of those platforms' native support for peer-to-peer, Web services–based reliable messaging. As that trend intensifies, ESB pure-play vendors such as Sonic, TIBCO, and Fiorano will find themselves under increasing pressure to distinguish themselves in the crowded middleware market.

About the Author

James Kobielus is a senior analyst in Burton Group's Application Platform Strategies Service, specializing in integration strategies and Web services.

jkobielus@burtongroup.com

Is your SOA really agile?

Building, testing and maintaining SOA solutions requires a flexible and collaborative approach to diagnostics. Finger pointing and confusion occur when all parties do not have a complete and common understanding of a problem. Mindreef® SOAPscope® 4.0 connects developers, testers, support personnel, and operations by combining the testing and diagnostic tools for each discipline and making it easy to share complete problem data. SOAPscope is the only end-to-end diagnostic system for Web services.

Developers • Testers • Operations • Support

Mindreef SOAPscope 4.0

Test - Agile testing approach for both developers and testers allows what-if scenarios without coding or scripting. Includes the industry's most complete interoperability suite and the flexibility to really test the edge cases.

Capture - Easily capture and assemble complete problem data into SOAPscope's workspace. Capture SOAP messages off the wire, test cases, or reproduced problems. Capture a snapshot in time of a WSDL including all related resources.

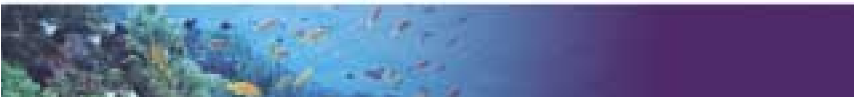
Share - Exported workspaces support sharing among team members preserving complete problem data including messages, WSDLs, and notes. Sharing workspaces is a great way to support web service users.

Solve - Diagnostic suite of analysis, comparison and visibility tools simplifies the understanding of complex XML, yet are powerful enough for the industry's most protocol-savvy experts.

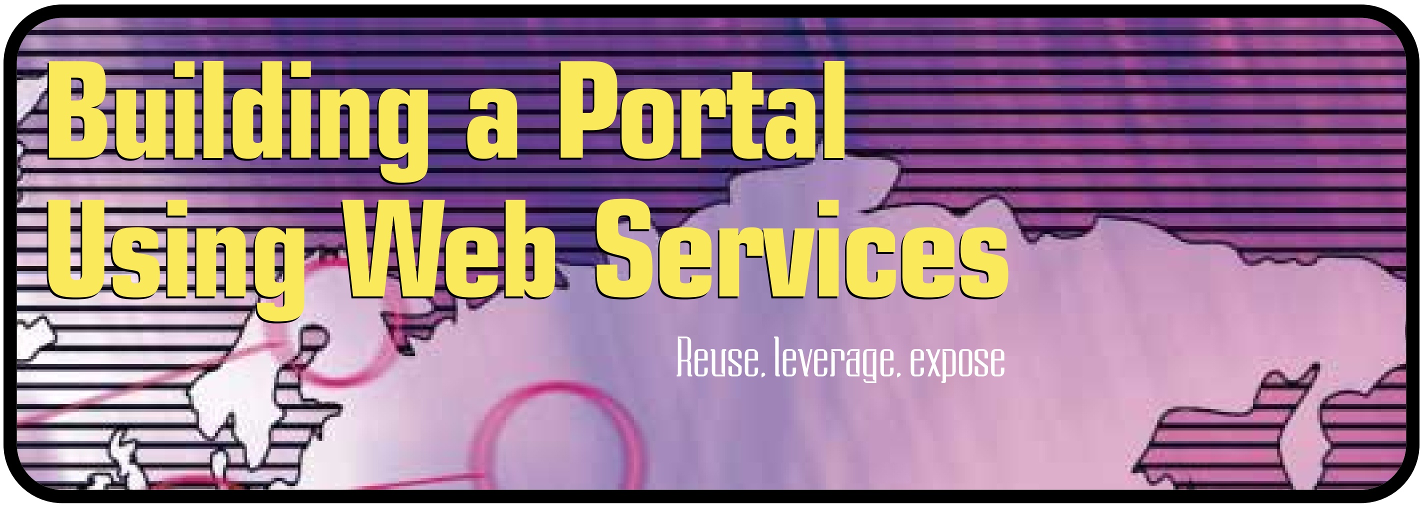
"SOAPscope has been invaluable in testing and diagnosing our enterprise scale Web Services. The new Workspace feature in SOAPscope 4.0 is an excellent way to support customers and collaborate as a team. A must have for every WS-Developer"

Simon Fell
Senior Member of Technical Staff at salesforce.com and author of PocketSOAP

Try SOAPscope **FREE** at www.Mindreef.com/tryout



Mindreef, Inc.
www.Mindreef.com



Building a Portal Using Web Services

Reuse, leverage, expose

The combination of portal technology, service-oriented architecture (SOA), and Web services provides customers with a powerful approach to developing, assembling, and deploying portal-based composite applications. Recent improvements in composite application development tools, coupled with a maturing portal market and a set of standards, provide customers with increased flexibility, promoting the reuse and repurposing of existing investments. While the underlying model and approach are not necessarily new, the integration of SOA and Web services as a key component in today's enterprise infrastructure provides customers with increased flexibility as they embark on new portal projects.

WRITTEN BY
PETER CARLSON &

BRIAN EISENBERG

The SOA model and corresponding Web services standards are breathing new life into the portal market where, over the past few years, the rate of portal adoption has slowed dramatically. Some of the contributing factors that accounted for this slowdown include:

- The reduction in IT budgets associated with the dot-com fallout
- The ongoing cost of integration and maintenance for back-end systems and enterprise applications
- The technical focus of these solutions, rather than a business focus driven by real, measurable business requirements
- The lack of robust development tools and an immature set of portal standards
- Limited flexibility for customers, with vendors offering prebuilt out-of-the-box "portlet packs" for integration with back-end systems

Before SOA and Web services, the only reasonable way to build out portal deployments with heavy integration requirements was to use proprietary software and application programming interfaces (APIs). While customers still derived benefits from these solutions, the cost of implementations was high due to increased service costs for custom development. These proprietary technologies and non-standards-based portal applications slowed business integration both within the enterprise and across the extended value chain. With this approach, each time a vendor released new versions of an enterprise application or system, portal vendors were often forced to reimplement their existing fragile portlet-based solutions. This proved to be a highly ineffective model, where portal vendors were responsible for fixing and maintaining their portlets to account for changes in the underlying applications.

To address these shortcomings, several forward-thinking integration vendors began to offer a common architecture, which ultimately drove portal vendors to leverage this approach in their solutions, often through partnerships or acquisition. While this approach was a great idea, it was only the first step on the way to achieving greater business value and operational efficiency. Fortunately, today leading business integration vendors have extended their architectures with support for SOA and Web services. This provides portal vendors with a new set of standards and technologies with which to pursue a new approach to integration with the back end. This is a key driver behind the current resurgence in portal projects and an overall increase in the rate of adoption.

With standards adoption and advances in development technology, there is a huge cost savings in the initial development of these sometimes complex, Web-based solutions. The usability of these development tools also streamlines the maintenance of these composite applications and promotes iterative development, which reduces implementation costs. Long gone are the days of developing proprietary point-to-point portlet solutions and the proliferation of brittle add-on "portlet packs" or "portlet factories" simply to provide back-end connectivity. To make it more exciting, an increasing number of application and system vendors now offer applications that can easily be decomposed into modular Web services and reassembled in new ways.

As you can see, the current standards and technology landscape have allowed the combination of portal technology + SOA + Web services to be complementary in nature. This combination allows customers to overcome many of the historical challenges that have existed in this space. By exposing high-value services across the business and making these services accessible to standard tools that business users can quickly assemble

Share Your SOAP

Now

LEARN
DEBUG
TEST
TUNE
SHARE

SOAPscope® 4.0

Try it free at www.mindreef.com/tryout

Web Services Diagnostics

into composite applications, customers rapidly achieve greater business value and agility.

Relevant Standards

From a standards perspective, the normal cast of Web services characters (SOAP, WSDL, UDDI, SAML, etc.) is still pertinent to this discussion. To assemble portal solutions with SOA and Web services, you must have a standard way to discover, describe, and secure those services. It can also be helpful to have a suite of tools that speeds the process of modeling your business processes and the decomposition of your existing applications and back-end systems into modular Web services. These ideas and standards quickly take us from traditional portal deployments into the much broader and valuable categories of composite applications, business process management, and business integration.

Something that cannot be overlooked in any enterprise deployment is, of course, security. There are many alternatives in this space, but one standard that is extremely important – as it relates to securely invoking Web services and providing proper information for authorization decisions – is SAML. The Security Assertion Markup Language standard provides the means by which authentication and authorization information can be exchanged between services. Tools and frameworks that are used to compose services together into broader composite applications need to have some mechanism for sharing credentials for single sign-on (SSO) as well as providing the necessary identity information for downstream authorization decisions.

The standards mentioned above are at the service level. Other standards in the industry have brought the idea of reuse "up the stack" and into the portal layer. These standards are JSR-168 and Web Services for Remote Portals (WSRP).

JSR-168 is a portlet specification designed to achieve interoperability between portlets and Java-based portal servers. The goal is to allow portlets to be packaged and deployed in a standard way on any server implementing the specification. By adhering to a well-known API, portlet developers can reach a broader audience and not lock themselves into a certain portal server's implementation.

WSRP is a standard intended to help in portal-to-portal communication. There are WSRP producers as well as consumers. WSRP clients can remotely invoke a WSRP producer or aggregate content. One of the most exciting things about WSRP is the fact that it truly is intended to aggregate and federate portal solutions by making it seamless to the end user as to which server the application or content is being served from.

Building a Portal with Web Services

Customers are asking for an increasing number of portal projects that focus on providing personalized, secure, composite views into a company's business processes. In order to achieve the full potential of portal + SOA + Web services, a broader solution set that extends beyond the reach of traditional "pure-play" portal technologies may provide the most cost-effective approach. In addition to the portal component, a robust (and SOA-enabled) business integration platform, coupled with business process management and optimization capabilities, provides the essential ingredients needed to fully achieve this potential.

The general process of building and deploying a portal-based composite application is described below. Due to the nature and wide ranging focus of portal projects, this is only one of many approaches that can be used to quickly build, assemble, and deploy portal-based composite applications.

Step 1: Define Your Audience and Their Level of Interaction

Here are some key questions to consider:

- Is this project geared for employees? Customers? Partners? Suppliers? All of these constituents?
- Does your project involve information aggregation, transformation, and access to systems and applications?
- What level of interaction is required? How do you currently access these systems?
- What are the key business processes into which you need visibility and control?
- Does your project require access to systems and processes that span beyond the reaches of the corporate network?

How you answer these questions will help drive specific project requirements and allow you to determine if the portal + SOA + Web services approach is well suited for your project.

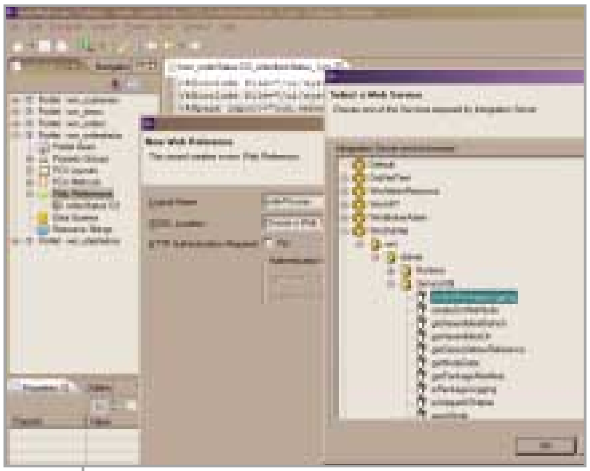


FIGURE 1 **Streamline the process**

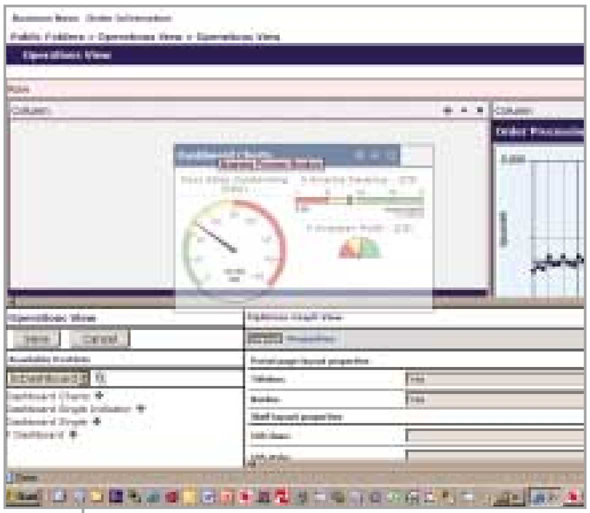


FIGURE 2 **Working from within the IDE**

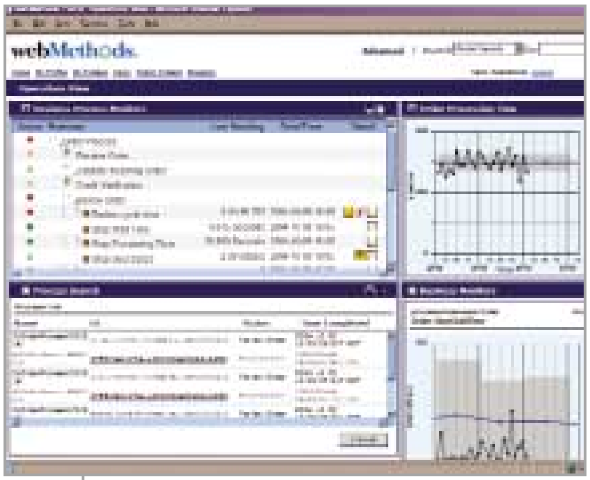


FIGURE 3 **Solution configuration**

Step 2: Involve Your Processes for Management and Analysis

It has become increasingly important to consider leveraging a business process management suite in these solutions: These suites can help to:

- Model the key processes involved and the various user-, data-, and application-level interactions
- Manage the runtime execution of a business process
- Define key performance indicators and metrics to effectively monitor, manage, and optimize processes

Step 3: Involve Your Integration Solution

As discussed in the previous section, portal products have historically provided point-to-point application-level connectivity. Customers now have business requirements that extend beyond what this traditional point-to-point model can deliver. For these more demanding requirements, you can easily leverage a standards-based business integration platform to abstract the application and system-level connectivity from the portlet layer and portal server runtime.

This provides a clean layer of abstraction between the data connectivity layer, the business process management layer, and ultimately, the runtime engine. With this model, the business integration component effectively manages all connections to the systems and applications; the business process management and monitoring component manages the process runtime and monitoring of the processes; and the portal component provides the user interface, personalization, and security for end-user interaction with a business process or composite application.

Step 4: Break It Down

As business processes are modeled and interaction patterns defined, the next step is to define the services and data required for any given business process. Today, many infrastructure and application vendors provide tools and product features that streamline this process. These vendors automatically expose the underlying applications, data connections, and related managed components as Web services. With this approach, it doesn't matter where the Web service is hosted, or what platform it runs on, so long as the service is clearly defined (see Figure 1).

Step 5: Assemble Your Solution

The process of integrating Web services defined in the previous step and building components that take advantage of these services may be slightly different depending on the vendor. For this article, we're going to focus on the development tools and methodologies that are required to develop Web service–enabled portlets, assemble them on a portal page or series of pages, wire them together to create the runtime data-level connections, and, finally, the process of configuring personalization and security for end-user interaction.

Most portal vendors provide development toolkits or plugins to existing integrated development environments (IDEs) for building portlets. Within these environments, portlet authors should have the option of discovering Web services and creating components from one or more of these services. The portlet author can choose the Web services required for a given portlet, drill down to select the exact data elements needed from the output stream, and select the display type from a list of predefined options.

Tools have advanced to the state where they can now generate all of the code needed to properly invoke the Web service from within the portlet container, as well as display its results. In addition, the portlet author can choose properties from these portlets that might be shared by other components on the same page or within the same portlet. This allows a developer to wire Web services together where the results from one Web service can be used as the input for the second Web service. No custom coding is required for the portlet to properly bind to, invoke, and display the results from any given Web service.

Another exciting development trend is the ability to build, deploy, and preview your work, all from within the development environment. This greatly speeds up overall development and helps you achieve your deployment goals faster (see Figure 2)

Step 6: Deploy and Configure Your Solution

After conducting the appropriate testing of your new Web service composite application, the final step is to deploy it to your production server. This may involve further refinement and configuration of security constraints and access controls that can be applied at various levels. It may also involve a solution that can be easily plugged into your staging-to-production requirements. Dependency checking may be needed on all components and many other deployment requirements as you take your new application and roll it out to the end-user community (see Figure 3).

What Next?

Project managers seeking to build portal-based composite applications have a powerful new model and approach to application integration, business process management, and composite application assembly. While portal technologies provide an important component in an overall solution, deploying a portal solution that "supports" Web services will *not* magically make your application infrastructure SOA enabled. Remember, the portal is only one of the many key components that make up a true SOA.

When considering a portal component as part of a larger solution, it's important to pay close attention to each vendor's toolset, integration model, and general approach to composite application development. Make sure you delve into each vendor's approach to application and system-level integration. Demand that each vendor show you the process for building a portal-based composite application using its tools, technologies, and SOA-enabled platform. Understand how the vendor can integrate with your existing business processes. Futureproof your investments by demanding that your application and systems vendors can be Web services-enabled. Demand that a vendor adhere to portal standards thereby not locking you into that vendor's solution. You have spent a lot of time and money on your data, systems, and applications – reuse them, leverage them, and expose them to the right users. Get there faster.

About the Authors

Peter Carlson is vice president and general manager of the Portal Business Unit at webMethods, Inc. He is responsible for driving technology strategy as well as coordinating all portal-related product development deliverables. Peter has over 14 years of technical management and software development experience. Prior to webMethods, he led engineering activities for Netegrity's Provisioning and Identity Management products. In addition to his technical management and strategic duties, Peter has served on many standards bodies and helped to implement server-side Java technologies within the Portal product line.

■■■ pcarlson@webmethods.com

Brian Eisenberg is the senior product manager in the Portal Business Unit at webMethods, Inc. He participated in the development and standardization of several core Web services standards at the W3C and OASIS, including XML Protocol Working Group (SOAP), UDDI, OASIS Security Services Technical Committee (SAML), and the ebXML Technical Architecture specification co-author. Brian holds a masters degree in Information Science and has served as product manager for webMethods' portal technology for over four years.

■■■ beisenberg@webmethods.com

Beyond SOA Platforms and Tools

Higher integration means more effectiveness

Software reuse process and infrastructure are key enablers for SOA success.

WRITTEN BY
MOTY AHARONOVITZ

Software engineering spent the better part of the 20th century stubbornly resisting standard engineering disciplines. Project introspection and peripheral management activities accepted by all other engineering fields as mandatory have frequently been avoided by many software development organizations. This resistance has had severe implications – with cost overruns, schedule slippages, quality and reliability issues, and consequent contract litigations becoming disturbing norms in the software industry.

Despite the huge progress and massive changes in development tools, architectures, and operational environments, it is interesting to note that the fundamental issues related to how individuals, teams, and corporations define and deliver software have not changed much. *The Mythical Man Month*, which is considered by many to be the classical book about the human element in software development, was published about 21 years ago, but has not lost much of its relevancy since then.

Anyone who has managed software projects is quite familiar with the "invariable" risks of the process. Indeed, these challenges manifest themselves over and over again, in thousand of organizations, in project after project. In fact, the changes in the business and technology worlds only make these systemic problems more difficult. Business is becoming increasingly "global," "rapid," "responsive," and "adaptive," and in turn it requires similar changes from its supporting IT processes and infrastructure.

According to a report published by leading industry analyst firm Gartner, "Competitive pressures are growing and the business environment is becoming less predictable, forcing enterprises to adopt the real-time enterprise as a business vision. Enterprises will prosper only if they continually remove delays from their critical processes, detect and respond to events, and accelerate their operations."

Consequently, software delivery cycles have become increasingly compressed and aggressive in an attempt to beat competition to the market as well as to reduce spending. Pressure on software development teams is mounting to predictably deliver technology initiatives that grow the top line while not affecting the bottom line. In other words, IT organizations are constantly required to do more with less. To add to the problem, the complexity of technologies and architecture has increased dramatically, becoming much more difficult to master and use effectively.

The net result of all of these changes is an environment of semi-controlled chaos, where things seem to be under control. In reality, though, it is rare that things are truly under control, and often there is no way to tell whether they are or aren't. All of these factors contribute to the formation of a high-risk environment, where the probability of project failure is significantly higher than its likelihood to succeed.

In examining the problems facing software developers and managers, two issues keep coming up:

- Alignment of IT priorities and capabilities to the needs and priorities of the business
- Making development teams and individual roles within teams more effective and productive

Together, business alignment and IT productivity represent the two most critical guidelines for IT agility, speed, and responsiveness. They make it possible for IT to aggressively support business revenue growth while staying within the boundaries of resource constrains. To close the so-called "IT gap," organizations are constantly looking for methodologies, tools, and platforms to facilitate alignment and productivity. The introduction of service-oriented architecture and its increasing adoption presents analysts, architects, developers, and managers with a new and viable paradigm for closing the IT gap.

The Emergence of Service-Oriented Architectures

Judging by the amount of industry buzz, many consider service-oriented architectures (SOA) and Web services to be the new technology silver bullet. More and more organizations are adopting or attempting to utilize SOA to various degrees, with SOA rapidly becoming a leading pattern in enterprise architectures and software development.

SOA represents an enterprise-wide architectural style that promotes stack agnosticism, interoperability, and loose-coupling between service providers and consumers. Web services is a specialization of the SOA approach, which relies on open standards and protocols such as WSDL, SOAP, UDDI, and HTTP, for discovery and communication between service consumers and service providers. This open approach, which enjoys broad-based industry adoption, ensures the future ubiquity and standardization of SOA, and Web services in particular, as the de-facto foundation for enterprise architectures.

Conceptually, SOA attempts to realign IT around the semantics of the business by representing applications and systems as a set of software services, which can be assembled together to support end-to-end business processes.

Services that expose business functionality or technical utilities with a low level of granularity can be composed and orchestrated to create services with higher and higher levels of granularity. The resulting structure can be thought of as a somewhat hierarchical graph of interdependent services. Since services consumers are typically only exposed to the top SOA layer of business services, this means that the behavior (implementation) or quality of service of these top-layer services can be changed on the fly by binding them to different lower-granularity services or components.

SOA represents an evolution of component-based development (CBD). CBD focus-

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

es on the assembly of applications using platform- and language-agnostic components, as well as managing the life cycle of components by container platforms. SOA represents a higher level of reuse by focusing on assembly and orchestration of business processes using a set of federated, platform- and language-agnostic services.

Despite being an evolutionary approach, SOA represents a true quantum leap in terms of aligning business and IT and closing the IT gap. It has the potential to deliver the architectural agility, resiliency, and flexibility that are required for delivering software-based economies of scale. However, despite the hype and the great potential, SOA and Web services implementations typically fall short of expectations despite massive investments in development tools and runtime platforms.

While there are several reasons behind such failures, the most critical one is associated with the realization that SOA is much more than a set of standards, tools, and platforms. Relatively little attention has been given to the SOA development process itself, which is very different from the "traditional" development process. In fact, many organizations do not realize that in order to succeed with SOA implementations, a radical cultural change has to occur within IT. The reason for this change is the fundamental role of service reuse, which strives to use existing services as much as possible to compose and orchestrate new services.

Software reuse as a concept has tremendous advantages, and many have tried to implement various reuse methodologies. Success, however, remains elusive. Beyond the reuse aspects of CBD, SOA adds service-level reuse typically required by process orchestration. In fact, the concepts of service production and service consumption explicitly imply and require service reusability. The reuse process has a life cycle of its own, which needs to be naturally integrated into the "normal" development process. It is therefore important to realize that establishing a service-oriented reuse process, which can be governed and monitored, is a key milestone on the road to SOA success.

Service-Oriented Reuse Process and Roles

Any large-scale implementation of SOA has to consider two groups of end users that are very distinct, have different skill sets, and require different tools to do their job. These two groups, the service producers (SP) and service consumers (SC), communicate via the SOA reuse process.

Service producers are typically architects and developers. Their part in the SOA reuse process is

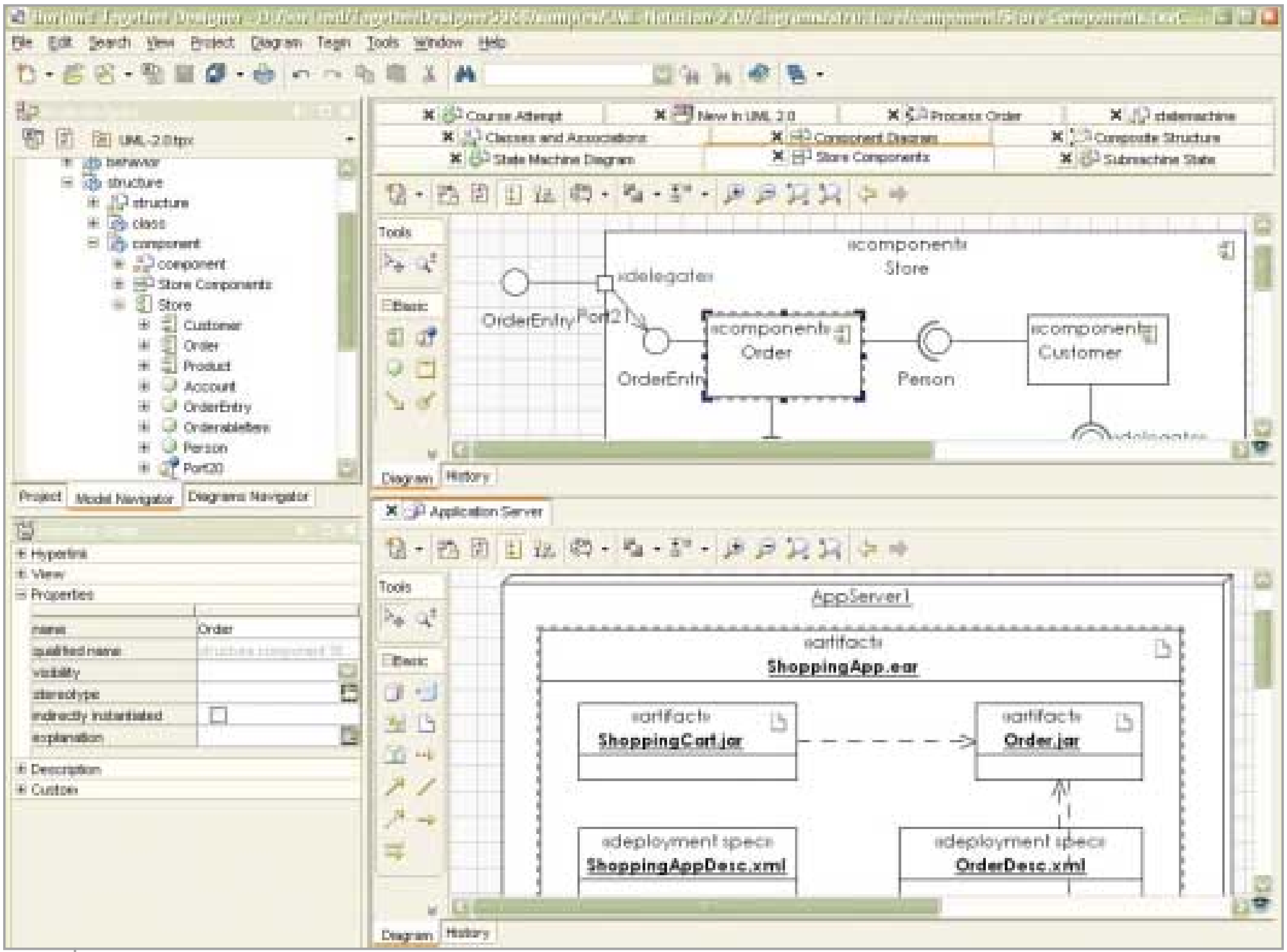


FIGURE 1 **Sample of a code-centric modeling tool**

the production and publication of meaningful business services. Services are typically created from scratch by implementing and wrapping components with remote/SOAP interfaces, or delivered via a “service façade” on top of legacy or packaged applications.

As a group, service producers are required to consider the low-level design decisions that make SOA happen. They deal with issues such as service interface design, service granularity, component design and implementation, deployment decisions such as colocating components based on interaction dependencies, as well as working with the operations team to support a host of required quality of service parameters, such as security, performance, scalability and availability.

In general, most service producers live in a code-centric world. They require sophisticated IDEs and software design tools to model, design, and implement their work products. They are responsible for the “produce” life cycle of the SOA reuse process. As such, they need to identify IT areas with reuse potential, and then specify, design, implement, document, and publish reusable services that expose these areas for service consumers. Special consideration has to be given to interface design and service granularity, as well as to building services in a generic enough fashion (i.e,. not with project-specific details) to enable reuse in as many contexts as possible and prevent service proliferation.

In contrast, service consumers are typically business analysts, although often developers and sometimes business persons also act in this role. In contrast to the service producers, who are experts in the solution and technical domains, service consumers own the business domain expertise. They understand the current business processes, and have the knowledge that is required to model, transform, and optimize them. In addition, they may define additional processes as a result of new business initiatives.

Service consumers essentially act as a liaison between the business requirements and the IT infrastructure. But in contrast to other methodologies, SOA makes this link very explicit by enabling service consumers to assemble and orchestrate business processes by reusing existing services that were defined and implemented by service producers. In a sense, service consumers provide the most coarse-grained services (the top SOA layer), which represent the software realization of the core business processes conducted by the enterprise.

Software Delivery Optimization and SOA

Borland, through its Software Delivery Optimization strategy, is one of the primary companies helping software development organizations close the “IT gap,” optimizing the way they manage risk across the complete life cycle of software creation from the line-of-business through development and operations.

Software Delivery Optimization views the process of producing software as a critical business process. As such, it strives to make it more predictable and reliable, similar to other engineering or manufacturing processes. It also views SOA as the dominant architectural style of the future, fully recognizing its advantages and viability.

Borland’s Software Delivery Optimization platform today includes critical components that enable partial implementation of the SOA production and consumption processes. Currently it has a very strong offering for the service production role, and its roadmap includes the SOA stack components required to automate the service consumption role to deliver a fully automated and integrated SOA process.

The tools used by service consumers are very different compared to those used by service producers. Being business-oriented and nontechnical in nature, service consumers require assembly tools, which enable them to search and locate existing services, and then wire and orchestrate them to form end-to-end business processes. These tools need to provide “soft” and intuitive modeling capabilities that fit the typical skills of business analysts and business people (see Figure 1).

Critical Infrastructure for the SOA Reuse Process

In order to succeed, the SOA reuse process requires automation, enforcement, and monitoring. Effective collaboration between service consumers and service producers mandates the utilization of the following solution stack:

- *A process engine that is deeply integrated with the development tools used by service producers and consumers* (i.e., code-centric IDEs and assembly tools). The purpose of this engine is to enable the definition, enactment, and monitoring of the SOA production and consumption processes, as well as streamline and accelerate information flow and collaboration between producers and consumers.
- *A dashboard that provides managers critical information about reuse level for any software project in the portfolio*. This information could be used to reward both producers and consumers who contribute to the SOA reuse process.
- *A service repository used to publish, document, search, and manage services*. The service repository serves as the primary communication mechanism between service producers and consumers.
- *A requirements management solution*. Both consumers and producers need infrastructure to manage their requirements, which are very different. While consumers manage business-level requirements associated with core business processes, producers manage domain-specific and quality of service requirements.
- *Code-centric IDEs and formal modeling tools* (i.e., UML based) as the primary workbench for service producers.
- *Process assembly and orchestration tools* as the primary workbench for service consumers.

Needless to say, the higher the integration between these stack components, the higher the effectiveness of the SOA process. ⓔ

About the Author

Moty Aharonovitz is a senior director of product strategy at Borland Software Corp, where he is responsible for directing the vision and direction for Borland’s solution platform. Prior to Borland, Moty acted as chief architect and VP of engineering for companies in the enterprise application space, such as diCarta Inc. and Diligent Software Systems.

■■■ moty.aharonovitz@borland.com

What Level Is Your SOA?

Choose for what you need – and maybe a little better

As I work with corporate America, as well as the government, I'm finding that services-oriented architectures (SOAs) are like snowflakes…no two are alike. I'm also finding that everyone has their own definition of SOA, and I've seen everything from messaging systems to portals called an SOA.

WRITTEN BY
DAVID S. LINTHICUM

So, who's right? I'm not sure I'm ready to declare somebody's architecture as non-SOA just yet;, however, there are some patterns that are emerging in terms of types of SOAs. I like to refer to these patterns as levels, since they have a tendency to move from the very primitive, or level 0, to the highly sophisticated, or level 5.

First, let me offer my definition of SOA, so we have a foundation of what is both correct and pure (tongue firmly in cheek).

In short, an SOA is a strategic framework of technology that allows all interesting systems, inside and outside of an organization, to expose and access well defined services, that may be furthermore abstracted to orchestration layers and composite applications.

Certainly not the only definition, but it is good enough for our discussion of SOA levels.

• ***Level 0 SOAs are SOAs that simply send SOAP messages from system to system.***

There is little notion of true services, but instead they leverage Web services as an information integration mechanism. Hardly an SOA, but certainly a first step.

It's also important to note that you don't need Web services to create an SOA. This is true for all levels.

• ***Level 1 SOAs are SOAs that leverage everything in Level 0 but add the notion of a messaging/queuing system.***

Most ESBs are level 1 SOAs, leveraging a messaging environment that uses service interfaces, but really does not deal with true services (behavior), instead moving information between entities as messages through queues.

While services are a part of Level 1 SOAs, it's really all about information and not about application behavior. For instance, while you do indeed invoke a service to push a message on queue and retrieve a message off a queue, it really leverages services as a well-defined interface and not accessing application functionality. Sometimes SOA architects may attempt to abstract application behavior using an ESB; if that's the case, you're moving up to level 4 (discussed below). However, doing this is typically much more trouble than it's worth because you're dealing with information-oriented integration technology that is merely attempting to deal with services/behavior… an unnatural act.

• ***Level 2 SOAs are SOAs that leverage everything in Level 1, and add the element of transformation and routing.***

This means that the SOA is not only able to move information from source and target systems, leveraging service interfaces, but is also able to transform the data/schemas to account for the differences in application semantics. Moreover, by adding the element of intelligent routing, you're able to route the information based on elements such as source, content, and logical operators in the SOA.

• ***Level 3 SOAs are SOAs that leverage everything in Level 2, adding a common directory service.***

The directory provides a point of discovery of processes, services, schemas, and such, allowing all those leveraging the SOA to locate and leverage assets such as services easily. Without directories, the notion of service reuse – the real reason for building an SOA – won't work. Directories are typically standards-based, including UDDI, LDAP, and sometimes more proprietary directories such as Active Directory.

• ***Level 4 SOAs are SOAs that leverage everything in Level 3, adding the notion of brokering and managing true services.***

Here is where the brokering of application behavior comes into play. In other words, at this level we are not only about managing information movement, but about the discovery and leveraging of true services.

At this level we have the ability to broker services between systems, allowing systems to both discover and leverage application behavior as if the functionality was local. This is the real goal of Web services, and the ability to share services not having to worry about platform-specific issues nor where the services are actually running.

"…it's really all about information and not about application behavior"

What's important here is that we understand that the value is in the behavior, as well as the information bound to that behavior. This level of SOA is able to provide capabilities for discovery, access, and management. Most SOAs are built with level 4 capabilities in mind, but may work up from the lower levels. If you do that, make sure you are leveraging the right technology and standards that support all levels.

• ***Finally, Level 5 SOAs are SOAs that leverage everything in Level 4, adding the notion of orchestration.***

Orchestration is key, providing the architect with the ability to leverage exposed services and information flows, creating in essence a "meta-application" above the existing processes and services to solve business problems."Moreover, level 5 SOAs should support both the notion of persistence and user interactions. Persistence means that there should be some sort of data storage mechanism that's able to service all attached applications and services. The data persistence layer is exposed as a service. User interactions mean that a level 5 SOA needs to provide mechanisms allowing services to interact with users through traditional or rich client portals.

Indeed, orchestration is really another complete layer on the stack, over and above more traditional application integration approaches we deal with at the lower levels. Thus, orchestration is the science and mechanism of managing the movement of information and the invocation of services in the correct and proper order to support the management and execution of common processes that exist in and between organizations and internal applications. Orchestration provides another layer of easily defined and centrally managed processes that exist on top of existing processes, application services, and data within any set of applications.

The goal of this type of SOA is to define a mechanism to bind relevant processes that exist between internal and external systems in order to support the flow of information and logic between them, thus maximizing their mutual value. Moreover, we're looking to define a common, agreed-upon process that exists between many organizations and has visibility into any number of integrated systems, as well as being visible to any system that needs to leverage the common process model.

Summary

As services, and architectures that support them, become more of an asset within the enterprise, we need to begin to learn how to categorize the patterns of the architectures, thus the SOA levels discussion in this column. This both provides a better understanding of what is a true SOA, and allows us to pick the right level to meet the needs of our business.

About the Author

Dave Linthicum is the CTO of Grand Central Communications (www.grandcentral.com) and has held key technology management roles with a number of organizations including CTO of both Mercator and SAGA Software. David has authored or co-authored 10 books, including the groundbreaking 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.

■■■ linthicum@att.net

The Best .NET Coverage Guaranteed!
*Offer subject to change without notice
Only $69.99
1 year (12 issues)*
* Newsstand price $83.88 for 1 year
Subscribe ONLINE www.wsj2.com or CALL 888 303-5252
SYS-CON MEDIA
SYS-CON Media, the world's leading i-technology publisher of developer magazines and journals, brings you the most comprehensive coverage of Web services.
WebServices JOURNAL

Was the Universal Service Registry a Dream?

A combination of the features in UDDI and RDF may just make the dream come true

It is sometimes beneficial to stop what you're doing, take a look around, and see where you've come from and where you are going. This regrouping is taking place right now across the software industry and is focused on the problem space of Web service description, discovery, and integration. At a high level, this article briefly discusses the progress made to date at solving the problem, describes the benefits and shortcomings of current technology, and presents a vision of the possible future of Web services infrastructure. At its most fundamental level, this article deals with the idea of programmatically locating a Web service that satisfies a specific need and then (programmatically) integrating that service into an application.

SOAP 1.1 (Simple Object Access Protocol) was published as a W3C Note in May 2000 and has since become an industry-standard format for enabling XML-based messaging and remote procedure calls. WSDL 1.1 (Web Services Definition Language) was published as a W3C Note in March 2001 and, despite some ambiguity that has proven a significant hurdle to interoperability, has gained widespread industry support as a mechanism for describing message structure and required transport details. UDDI (Universal Description, Discovery and Integration), the third member (after WSDL and SOAP) of the Web service standards triumvirate, was first published in September 2000.

WRITTEN BY
HARRIS REYNOLDS &

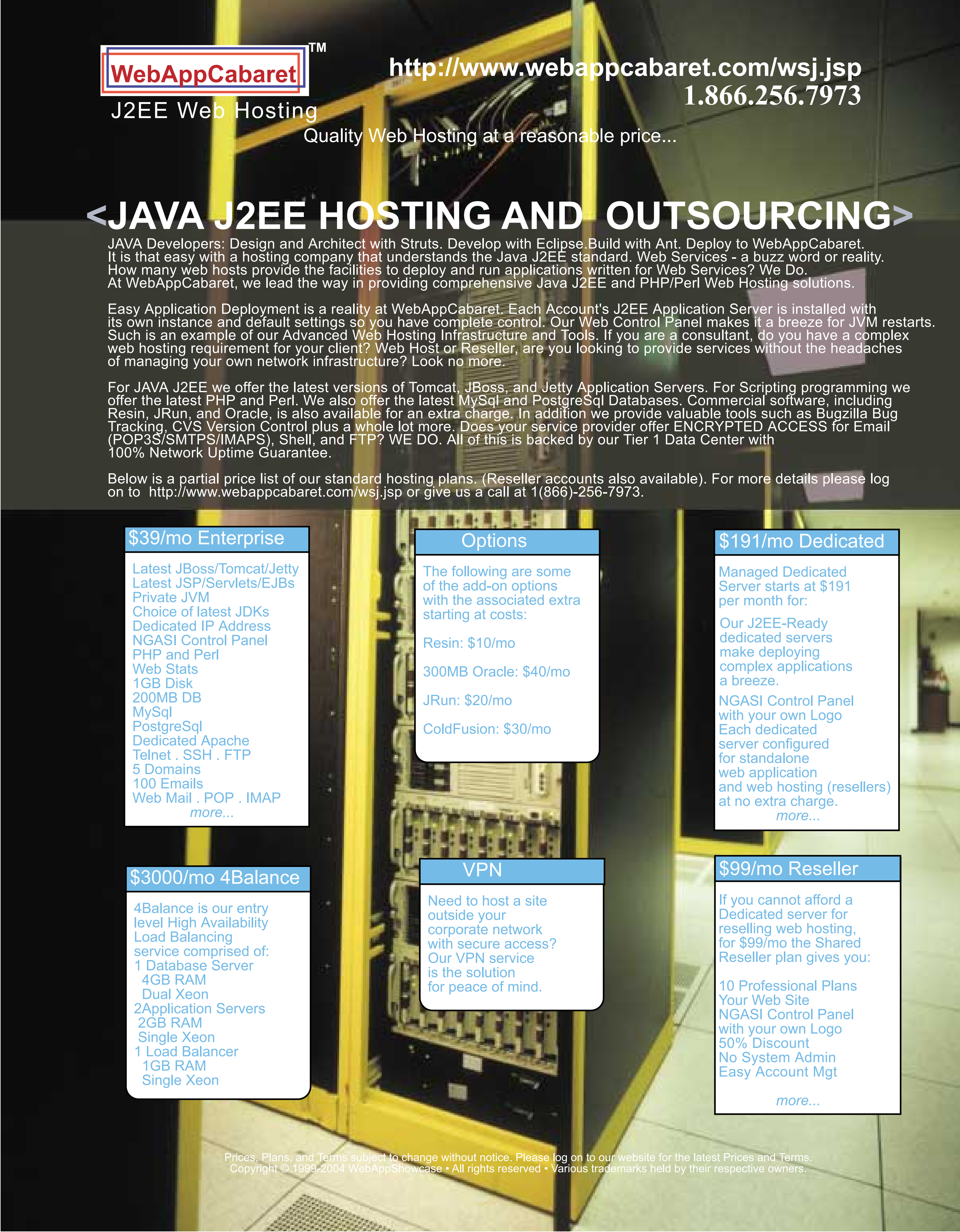
FRED HARTMAN

However, in contrast to its siblings UDDI has not enjoyed the same level of industry acceptance. It is particularly relevant in this article because it is the industry's first attempt to solve the problem of discovery and integration of Web services, in particular as discovery and integration support Web service reuse and sharing. The need to reuse and share services is arguably the most compelling argument for service-oriented architectures (SOAs).

Several factors contributed to UDDI's lackluster start out of the gate.

- UDDI was a specification that was ahead of its time. It was designed to enable management of large numbers of Web services, but at the time of its release, companies tended to have relatively few Web services to manage. Only now, after four years, are companies reaching the point where the number of Web services justifies the need for a registry to aid in organization and discovery of services.
- UDDI was originally intended to serve as a Universal Service Registry that would be a panacea for all the problems of discovery and integration with Web services-based applications. This intention has yet to become a reality. Currently, a search of the registries hosted by companies like IBM and Microsoft typically yields a set of services that are unregulated and thus unreliable – services that do not work, or have been posted by companies that no longer exist, and so on. Enterprises are therefore not using these public forums to organize services and develop meaningful business relationships.
- The UDDI specification has suffered owing to its somewhat esoteric nomenclature, which includes terms such as "tModels" and "BindingTemplates." These terms, while digestible to the infrastructure developers who are building the plumbing, are not as palatable to developers who are writing business-specific applications. These terms negatively affect the usability of UDDI.

As mentioned earlier, this article will also explore how RDF (Resource Description Framework), a developing technology dedicated to defining and maintaining a Web of relationships between different types of information, fills some of the technology holes that have hindered UDDI's ability to fulfill its promise. Although not one of the mainstream Web services specifications, RDF is capable of providing a technology basis for the discovery and utilization of Web services. Specifically, RDF is an XML-based language for representing information about resources and the relationships between the resources intended for representing metadata about these resources. It is one of the building blocks of the W3C's Semantic Web Initiative. Furthermore, OWL (Web Ontology Language; www.w3.org/TR/owl-features) builds on RDF by providing a formal vocabulary to describe the meaning of classes and properties, thus enabling powerful automated reasoning to be performed on RDF data. While still relatively imma-



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:

Our J2EE-Ready dedicated servers make deploying complex applications a breeze.

NGASI Control Panel with your own Logo
Each dedicated server configured for standalone web application and web hosting (resellers) at no extra charge.
more...

$3000/mo 4Balance

4Balance is our entry level High Availability Load Balancing service comprised of:
1 Database Server
  4GB RAM
  Dual Xeon
2Application Servers
  2GB RAM
  Single Xeon
1 Load Balancer
  1GB RAM
  Single Xeon

VPN

Need to host a site outside your corporate network with secure access? Our VPN service is the solution for peace of mind.

$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...

ture, these standards offer interesting alternatives to the traditional Web services landscape and are worthy of investigation within this context.

Combining the capabilities of the current state of UDDI with the capabilities of RDF and OWL promises to resurrect the quest for the Universal Service Registry. The following sections of this article contemplate the features necessary to build what we're going to call, for the purposes of this article, an Ideal Service Registry (ISR) and how the technologies mentioned earlier will play.

The Ideal Service Registry

The following are some of the features that must be implemented to create an Ideal Service Registry (ISR).

Service Description

It must be easy to describe Web services in an ISR

There are many ways one can define a service in UDDI, which makes publishing and using inquiry results quite difficult. A big step toward usability, at least with regard to publishing services, came about with the publication of a technical note entitled "Using WSDL in a UDDI Registry, Version 2.0" (available at www.oasis-open.org) that outlines a set of "best practices" for mapping a service description into a UDDI registry. The best practices outlined in this note led to the incorporation of WSDL (an XML-based method of describing the inputs, outputs, and available methods of a Web service) into the UDDI registry. Following these best practices has greatly simplified the process of publishing a Web service into UDDI. Developers simply supply a WSDL URL and a corresponding "BusinessEntity".

There is no standard for mapping a Web service into an RDF registry, although OWL-S has been released and promises to help satisfy this need. A Web service's metadata, such as a description of what it does, its inputs, its outputs, its pre- and post conditions, along with the technical details of how to physically invoke and interact with the service, can all be described in RDF via vocabulary defined by OWL-S. Encoding the details about a service as RDF greatly increases the ability to semantically discover and use Web services. In an order process, for instance, there needs to be a way to know that after calling the ProcessPO Web service one should next call the CheckPOStatus Web service to gather the status of the order.

A UBR (UDDI Business Registry) must have a standard publication API

The publication API of a registry is a common set of commands that are incorporated into client programs to access the facilities of the registry such as adding, deleting, or modifying a registered service. UDDI is solid in this area because it comes with a standard API defined in WSDL that allows a SOAP interface to the publication API. RDF has a proposal for RDF Net API (available from www.w3.org) that defines a SOAP interface for adding, updating, or deleting RDF statements, but it is not a standard and is not tuned for use in defining ISR content.

An ISR must support data described in multiple languages

To accommodate multiple languages, the specific language used in information objects must be described by standard language tags. This allows data to be presented to the person utilizing the information in their preferred spoken/written language (as opposed to programming language.) The latest, most comprehensive proposal is RFC 3066bis (available from http://inter-locale.com). Both UDDI and RDF support the xml:lang adornment of literal values, which can be set to RFC 3066bis values. This is a very important concept as it moves the registry beyond the general discovery of business logic and actually increases the usefulness of the information resulting from the use of a service.

An ISR must support multiple categorization hierarchies.

In both UDDI and RDF, you can classify an entry using multiple classification hierarchies.

In UDDI, categorization hierarchies are called "taxonomies." Organizations such as UNSPSC (Universal Standard Products and Services Codes) and NAICS (North American Industry Classification System) have created sets of UDDI technical models, known as tModels, which describe an interesting categorization hierarchy. (For more information, see http://uddi.org/taxonomies/UDDI_Taxonomy_tModels.htm.)

RDF calls classification hierarchies "ontologies," which are usually defined using OWL. A number of organizations are creating ontologies. One of the most interesting ontologies is OWL-S, whose authors are trying to create an ontology that will fully define all Web services artifacts and that will include defined places and formats for describing information required during the integration phase of using the registry contents. For more information, see the DARPA Agent Markup Language Homepage at www.daml.org.

Categorization hierarchies should come with user-understandable, locale-specific names

UDDI tModels contain a name property, which can be decorated with the xml:lang property. OWL does not currently have a standard way to attach display information to an ontology although RDFS (RDF Schema) has some predefined properties such as rdfs:label and rdfs:comment, which can be used for the human readable display name and description. These can also be decorated with the xml:lang tags.

It should be easy to derive relationships among entries in the UBR

Two categorization hierarchies can each contain a category that defines the same type of item, but the two categories are often named differently in the different structures (for example, "cars" as opposed to "automobiles"). This situation can occur, for example, when the categories were created by two different people or

"UDDI implementations are solving real-world business problems today"

Fix them fast.

The SIFT™ solution framework from Service Integrity™ is the leading monitoring and analysis software for XML and Web services applications. SIFT will allow you to rapidly identify and resolve performance issues in your distributed applications. SIFT installs in minutes and automatically discovers your Web services and provides complete visibility into HTTP and XML data flows — all without a single line of custom coding. SIFT will untangle your problems fast.

IT Applications of the SIFT™ Software:
- Intelligent Logging to save Time/Expense of Custom Coding
- Deep Visibility into Application Health and Security Status
- Improved Policy and SLA Compliance
- Fully configurable alerting
- Support for Billing and Metering
- And more...

"SIFT has been an invaluable tool and I don't know how we'd live without it. We've been able to monitor system performance and isolate a few bugs that had gone undetected in the past."

Director of Operations at one of the largest commercial banks in the United States

Service integrity

Visit **www.serviceintegrity.com** for more information and enter to **win a FREE COPY of *Securing Web Services with WS-Security,*** by Dr. Jothy Rosenberg, CTO, Service Integrity, Inc. Fill out our registration form and you will be automatically entered to win!

"it is the industry's first attempt to solve the problem of discovery and integration of Web services"

organizations. To rationalize such overlaps, one must be able to indicate that a category in one hierarchy is the same as a category in another hierarchy. Users that query one category will then also get corresponding entries in the other category. This is currently not possible in UDDI, but is the reason OWL was created. This capability is KEY to making registries useful as it allows meta-relationships to be defined that will allow programs to dynamically find alternative services when their intended service is unavailable, for example.

Discovery

There are two significantly different aspects to discovery. One is the ability to discover registries to query. The other is the ability to find the appropriate service in a registry.

The first problem could be addressed by a number of industry-accepted standards and de facto standards such as WS-Discovery (http://msdn.microsoft.com) or zeroconf (www.zeroconf.org), or even the locators that are pluggable into software solutions such as WebMethods Fabric (www.Webmethods.com). After configuring some locator rules on the client, one can find all the available registries that match those rules. The rules could define something like a UDP broadcast, the URL of a well-known server, or parameters for a SOAP call.

After your tool has discovered one or more registries, you should be able to navigate easily through the entries in a registry. There should be at least three navigation methods:

- ***Naive walking through all the entries with links among related entries.*** Optimally, you should be able to infer links so that relationships can grow without having to define every possible link.
- ***Pre-built or user-defined query that returns a specific entry.***
- ***Query that returns a group of entries, where each entry allows you to navigate to entries that relate to any returned entry.*** This navigation method is analogous to performing a Google search and then following HTML links from one of the pages returned by the search.

ISR queries must have the following attributes:

The ISR must nicely support querying data with text descriptions in different languages

Text-matching queries in the current incarnation of UDDI must be written specifically to the desired language of the return text. This means queries cannot be pre-canned. It also means that even simple queries must know that "en", "en_US", "en_UK" and "en_AU" are all English. If a query was written for just one of the flavors of English a complete set of English results would not be returned.

When reading the description of a Web service, the user should not get back all English entries, just the one entry that best matches the user's desired language. This becomes even more complicated when a user speaks two languages. For instance, for someone that reads French and English, a query must contain OR clauses for "fr", "en", "en_US", "en_UK" and "en_AU", but in UDDI there is no way to define that any French dialect is preferable to any English dialect.

There is a nice proposal from Jeremy Carroll of HP and Addison Phillips of webMethods (http://inter-locale.com/whitepaper/iswc2004.pdf) on how to solve this problem nicely for RDF query languages. A similar solution could be implemented as an extension to the existing UDDI inquiry API.

The results of the queries should be filtered based on authorization rules

Even within one company, there are services that some users should see and others they should not. Such a security scheme allows administrators or project leads to set up filters that surface "approved" services to the appropriate developers.

UDDI does not yet have a standard data security implementation. Rudimentary security requires you to have a valid user name and password. After a person has been authorized to access a UDDI registry, he/she can then query all information in the registry. UDDI has no mechanism for selectively sharing a certain subset of information with one group of users and another set of information with another group of users. Many UDDI vendors have added proprietary extensions to support security on the data within the UDDI registry, but the security definitions are not portable.

RDF does not have a standard, meaningful security model, and given its roots in the open world of the Web, there does not seem to be widespread interest in solving this problem.

An ISR needs a standard inquiry API

UDDI comes with a WSDL that defines a SOAP interface to the inquiry API. RDF has a proposal for the RDF Net API that defines a SOAP interface for querying RDF statements, but it is not a standard.

An ISR needs a standard query language

UDDI defines a standard query language. RDF has a proposal for an RDF Net API that defines a SOAP interface for querying RDF statements, but it is not a standard. RDF also does not have a standard query language, although the RDF Data Access Working Group is working to define one.

Integration

After a service has been discovered, the following features are required to enable you to create a composite application:

An ISR contains a link to the service's WSDL

Either the full contents of the WSDL or a link to the WSDL must be available for a client to interact with the service via the SOAP protocol. This is a common property of UDDI service entries.

OWL-S can be used to define the elements of the WSDL as RDF. In theory, an OWL-S–aware program could read the RDF information and invoke the service without a WSDL. The UBR needs the ability to generate (preferably dynamically) the WSDL from the RDF data in order to support non OWL-S aware SOAP tools that require the WSDL.

A UBR contains information about the prerequisites of each operation

Most operations that are part of a business process have some sort of context, required call order, or other prerequisite. WSDL does not describe prerequisites of a service invocation, which means that information needs to reside in the UBR itself. The OWL-S project is trying to describe this for RDF.

The Future

For all the limitations of UDDI, there is currently no alternative registry standard that can match its maturity or features. RDF has much interesting potential, but is still far from being a viable ISR. A combination RDF and UDDI implementation needs mature implementations of a number of components and interfaces that are currently in their infancy. These include:

- A standard query language being worked on by the RDF Data Access Working Group (www.w3.com)
- A query language that usably supports data in multiple languages
- A comprehensive and standardized programming API
- A Web services (SOAP, WSDL) API
- A way to find services across multiple registries

UDDI implementations are solving real-world business problems today and UDDI is not standing still. There are proposals to add RDF features to UDDI. Some UDDI vendors have extended their implementations with more granular security controls, merged service namespace across multiple registries, and identification of equivalent services to allow on-the-fly service failover without requiring hardware clustering. There are solutions for the limitations of UDDI, but it remains to be seen if these and future proposals will be accepted, and if the timeline toward workable implementations will stay ahead of the progress in the RDF world.

About the Authors

Harris Reynolds is lead engineer of webMethods' UDDI server. Prior to joining webMethods, he was a consultant specializing in XML integration. Harris has a BS in accounting from the University of South Florida and a MSA in information systems from Auburn University. He is currently completing a MS in computer science from Auburn University.

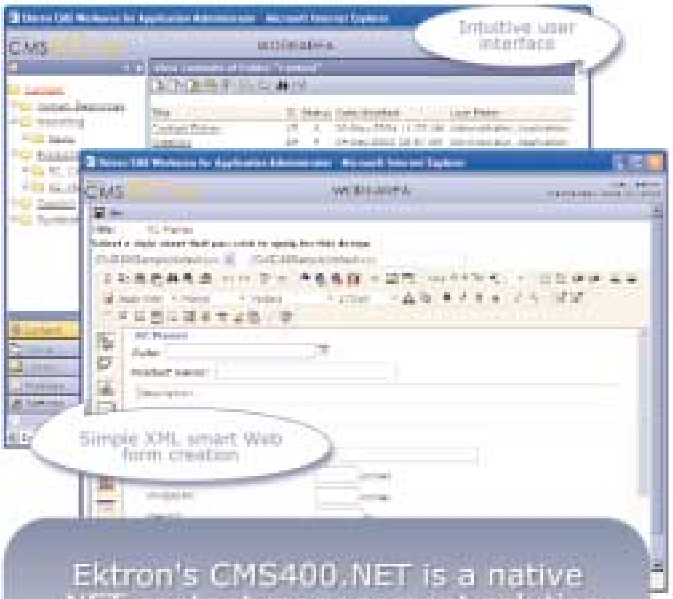
■■■ hreynolds@webmethods.com

Fred Hartman is currently a member of the webMethods Advanced Technology Group and was previously engineering manager for the webMethods Integration Server Team and an engineer on webMethods' XML technologies. Fred has held engineering or management positions for Merant, XDB Systems, Intellution, IBM and EMC Controls, working on database systems, middleware, process control systems, and advanced signal processing networks. Fred has a BS in computer science from the University of Maryland and a MS in computer science from the University of Iowa. He has previously written about music for *Dirty Linen Magazine*.

■■■ fhartman@webmethods.com

Simplify Content Management With

ektron CMS400.net



Ektron's CMS400.NET is a native .NET content management solution with the usability business users need, and the familiarity developers want.

Benefits of CMS400.NET:

- ✓ Extend Visual Studio .NET 2003 with drag-and-drop server controls for template development
- ✓ Design-time rendering of content blocks and other CMS components
- ✓ IntelliSense processing and public-class inheritance for data binding
- ✓ Leverage XML Web services for content reuse
- ✓ Complete workflow, task management, globalization capabilities

Try it for FREE

Download a free 30-day, fully-functional trial or request a personalized demonstration at

www.ektron.com/ws

Ektron - Redefining Web Content Management

web services EDGE conference & expo

Web Services Edge 2005 East

International Web Services Conference & Expo

New for 2005:

- **Colocating with LinuxWorld Conference & Expo** badges give access to both shows
- **Guaranteed Minimum Attendance 3000 Delegates**
- **New, Hot Sessions & Seminars**

The Largest *i*-Technology Event of the Year!

Tuesday, February 15
11 a.m.

Matt Ackley
Senior Director, eBay Developers Program
Topic:
Web Services for e-Commerce

Keynote Speakers & Featured Guests

Wednesday, February 16
11 a.m.

Ari Bixhorn
Director, Web Services Strategies, Microsoft Corporation
Topic:
Indigo and the Future of Web Services

Anne Thomas Manes
Burton Group
Application Server Shoot-Out Facilitator

Hynes Convention Center
Boston, MA
February 15-17, 2005

For Exhibit and Sponsorship Information:
Jim Hanchrow, 201-802-3066, jimh@sys-con.com

Register Now! Hot Sessions & Seminars

- Ensuring Web Services Interoperability
- Web Services Standards: Going Behind the Mask
- The Role of Policy in Web Services Integration – It's More Than Just Security
- Four Abilities SOA will lack without a Registry
- Driving SOA Governance
- BPEL Best Practices from Real-World Projects
- SOA: From Pattern to Production
- Lessons From the Front Line – Building Interoperable Web Services
- Developing E-Commerce Applications with Web Services
- Developing Enterprise Class Web Services
- Orchestrating FORCEnet Engagement Packs with BPEL for Web Services
- CPI: A Global Integrated Problem Tracking and Resolution System using Java Web Services
- Using Service-Oriented Architecture and Web Services to Issue Business Licenses in the District of Columbia
- Developing Web Services with Eclipse
- The Interoperability Challenge of Web Services Security Standards
- Securing Web Services with WS-Security
- Anatomy of a Web Services Attack
- Using a Mobile Phone as an SSO Authentication Device in SOA Solutions
- XML Content Attacks
- XACML and Agnostic Authorities

Register Today: www.sys-con/edge2005east/registernew.cfm

Come and see what everyone is talking about…

The Application Server Shootout!

Find out which server best fits your performance and ROI metrics

Join us in delivering the latest, freshest, and most proven Web services solutions…

At the *Fifth Annual Web Services Edge 2005 East – International Conference & Expo* as we bring together IT professionals, developers, policy makers, industry leaders and academics to share information and exchange ideas on technology trends and best practices in secure Web services and related topics.

Exhibit in *The Web Services Pavilion* and see thousands of buyers!

Becoming a Web Services Edge Exhibitor, Sponsor or Partner offers you a unique opportunity to present your organization's message to a targeted audience of Web services professionals. Make your plans now to reach the most qualified software developers, engineers, system architects, analysts, consultants, group leaders, and C-level managers responsible for Web services, initiatives, deployment, development and management at the region's best-known IT business address– The Hynes Convention Center in Boston.

3-Day Conference & Education Program features:

- Daily keynotes from companies building successful and secure Web services
- Daily keynote panels from each technology track
- Over 60 sessions and seminars to choose from
- Daily training programs that will cover Web Service Security, J2EE, and ASP.NET
- FREE full-day tutorials on .NET, J2EE, MX, and WebSphere
- Opening night reception

PRODUCED BY SYS-CON EVENTS

www.sys-con.com/edge

Sponsored by: wldj JDJ WebServices SD Times .NET Journal Information Storage + Security Journal WebSphere XML Journal MX eaIndustry asp.netPRO

All brand and product names mentioned above are trade names, service marks or trademarks of their respective companies.

Solve Your Application Security Issues

The advantage of building app security into infrastructure

I'm sure I'm like many of you in this respect: I got into engineering because I love the idea of being able to address complex problems with a combination of my talent, my friends' talent, and the tools that I can come up with to make our work as easy as possible (work smart not hard!). It is this approach that has guided me in my work as an application and technical architect. I come to work every day looking for that "wow" feeling that comes when I realize that another problem that seemed intractable has been solved. But in the pantheon of hard problems that beset projects I work on, a disturbing antipattern has emerged regarding application security design.

WRITTEN BY PAUL O'CONNOR

It goes like this: the business declares a requirement for zero tolerance for security flaws (it is a catastrophic problem for one user to see another user's account, or HIPAA requires that no one see some data in a user's profile except the user or her doctor, and so on)…then the architecture team looks at the tools on hand and decides that they don't pass muster…then the architecture team searches high and low for some tool or framework that will do the trick…they don't find anything and decide to construct it for themselves thus turning over to QA a huge burden of making sure that the result always works. And the worst part is that the maintenance cycle is merciless on home-brew security systems since they have to evolve – new roles and constraints are often added on a regular basis, meaning that application developers might need to change code if they did their job without enough foresight, and QA needs to retest the system in any event.

I'm happy to say that I recently had that "wow" feeling again with regard to application security after doing Web services security architecture for a few months. I'm going out on a limb and predicting that Web services and WS-Security standards will lead to a paradigm shift in the way that applications are secured.

Application Security Infrastructure – Separating Policy from Code

Security policy and enforcement must be separated. Architects have long recognized this as a fundamental design tenet for applications. The last paradigm shift in application security architecture was the emergence of tools and frameworks that allowed user access management infrastructures to be deployed as a service that is shared between applications. Identity and access management solutions emerged when the Web got popular as a means to implement Web resource access policy and single sign-on (SSO). These systems were generally quite successful in managing user access to Web sites, and even delivered plug-ins for application servers with which we were able to do static access control list (ACL)–type security against things like JNDI locations and access to EJB methods. They never delivered a means to do fine-grained entitlements or data security. We looked to other means for that. We still wanted an infrastructure approach to the problem and so many of us wrote our own security providers for app servers.

Managing User Identities – Everyone Wants to Control Their Own Destiny

Just as hard to get your arms around if you're an app architect is identity management. Most applications have user self-service functionality. Some have a large number of functions that users can manage. Everything from self-registration to password resets, and personalization settings are managed by the application itself, or so it seems to the user. Same thing goes for delegated administration. If I am a financial advisor, for example, I may well be able to manage my clients' accounts and even grant them access to view their accounts on the Web. I likely will not be able to view the accounts of a colleague's clients. Behind the scenes is a part of the application that manages the identity store – usually an LDAP store. The same access management tools that implement Web access policy often provide a Web app of their own for this purpose, for a jaw-dropping additional license fee. And so we architects have been dealing with these identity and access management (I&AM) tools for the last few years, biding our time for more fine-grained security functions that could look at a parameter or two (data security).

There is a (Much) Better Way

I am glad to say that there is a much better way now to do fine-grained application security. The premise of this article is that Web services security policy can and will supplant existing app security policy and in doing so will greatly limit the difficulty and expense of securing applications and managing user identities. I am suggesting that for the first time we will be able to include fine-grained policy and data security in the security infrastructure itself instead of in application code. And we will be able to expose "raw" SOAP interfaces to the I&AM system and have delegated administration and user self-service schemes encoded as WS-security policy. As you must have surmised by now, to make use of this scheme applications must leverage Web services for interaction with the business service tier. I don't see this as a caveat, I see it as desirable. And so do the clients I currently work with.

Security Standards We Need – A Quick Review

Like all other things Web services, standards are the heart and soul of Web services security. A quick review of the standards upon which we rely is in order. And it all starts with the aptly named WS-Security standard, which is the granddaddy of them all, having been around in some form for two years. WS-Security is all about bringing integrity and confidentiality to SOAP messages, and is also concerned with conveying the identity of the requestor on whose behalf the message is sent. The standards relied upon are XML-Signature, XML-Encryption, and SAML (for which a SOAP token profile is provided). SAML (Security Assertion Markup Language) details both a grammar for representing user identities via tokens and a protocol for gathering user attributes and authorization decisions. XACML is a grammar that details security access policy and a high-level protocol for access requests and responses. The XACML protocol is given life via a SAML profile in the SAML 2.0 draft – we will put this to good use shortly.

The Web Services Policy Ecosystem

Figure 1 illustrates the Web services policy ecosystem as specified in SAML 2.0 and the SAML profile for XACML. Policy is enforced at the policy enforcement point (PEP) and calculated at the policy decision point (PDP) – we still have the separation of enforcement from decisioning that we insist upon. The difference is that we will do all of the enforcement in the infrastructure in this architecture. In doing so, we may need to gather more context information, both at the level of the PEP and the PDP. This is accomplished with the policy information point (PIP). In the XACML policy model, attributes are gathered into one of four categories of the request context: Subject, Resource, Action, and Environment. The PDP will make an access decision based on the attributes in these categories, with respect to the policy for the targets in the request context. The policy administration point (PAP) supplies the policies to the PDP. The interaction between the PDP and the PAP is very interesting and I would encourage you to check out the XACML standard at OASIS. Understand that there are no real XACML tools on the market yet, though they will soon appear – we actually have PEPs now but are

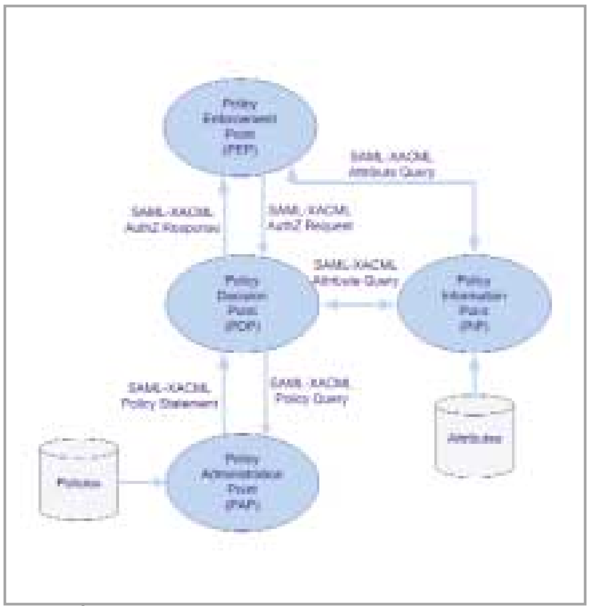

Policy Enforcement Point (PEP)
SAML-XACML Attribute Query
SAML-XACML AuthZ Response
SAML-XACML AuthZ Request
Policy Decision Point (PDP)
SAML-XACML Attribute Query
Policy Information Point (PIP)
SAML-XACML Policy Statement
SAML-XACML Policy Query
Attributes
Policies
Policy Administration Point (PAP)

FIGURE 1 | WS Policy Ecosystem -- SAML/XACML 2.0

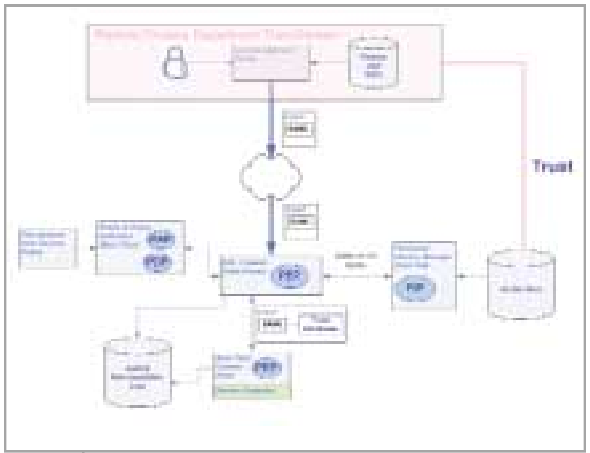


FIGURE 2 | Enforcing policy in the infrastructure – future state

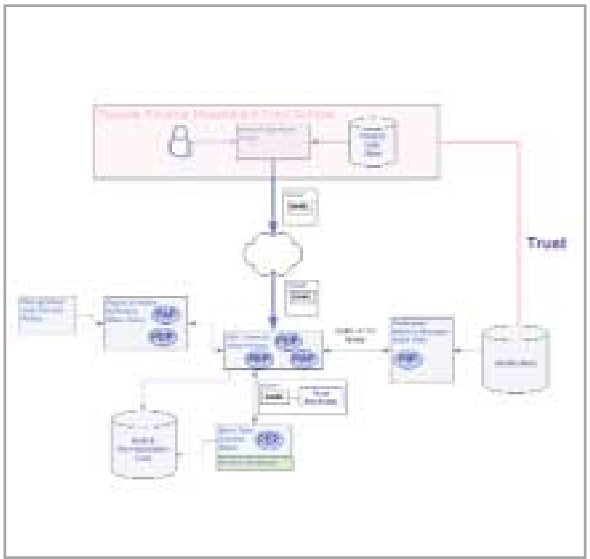


FIGURE 3 | Enforcing policy in the infrastructure today still waiting for XACML PDPs and XACML policy stores and associated administration tools. Instead, let's focus on what we currently have to implement the policy ecosystem.

Tools to Enforce Security Policy in the Infrastructure

Since I can't just overlay each of the policy ecosystem bubbles in Figure 1 with some tool function just yet, we will have to improvise. Luckily, there are some very good tools in this space already, which makes this task bearable. In fact, without these tools it wouldn't be possible at this time to implement the premise of this article. Central among the tools we leverage is the XML security gateway, otherwise known as an XML firewall. There are many of these tools on the market now, mostly from startups, and they are ready for prime time. The best of them do everything from content routing to XML acceleration, but most importantly security policy enforcement. They also offer configurable logging. And the ability to log access decisions and request content is very powerful, as we will see in our use case.

The best of breed, in my opinion, is the XS40 from DataPower. This device is incredibly fast and incredibly flexible at the same time. It is equally adept at running the entire show with everything from PKI management to policy setup and enforcement as it is at deferring everything but the actual enforcement to other systems in your enterprise. And it screams. It is hard to stack enough validation, enforcement, and cryptography functions on it to slow it down below wire speed…at transaction throughputs that would run the economy of a small country. And that's just one box. Load up a cluster of them and the sky's the limit. The main take-away for our purposes is that we will do our message-level security functions, XML attack countermeasures, logging, and our policy enforcement all at the same time and not worry for a second about performance.

The second tool that we will rely upon is the Web services management fabric. Web services fabric tools allow administrators to manage every aspect of a Web services endpoint, from global security constraints a la WS-Policy (like whether messages must be encrypted or signed), service levels and provisioning, and eventually fine-grained policy via XACML. WS fabric monitors service latency and enforces policy via a network of policy enforcement points that are spread throughout the enterprise. This control network is a very powerful distributed management system and will surely assume more functions over time, including the ability to enforce fine-grained policy (I hope). The best tool in this space is Blue Titan Network Director. I am impressed with the maturity of the Blue Titan fabric and with it's ease of use.

To get farther using Web services and Web services security standards to build an application security infrastructure you need trust and federation. These are beyond the scope of this article and still evolving. What I will define is what I call "poor man's trust," which will support federation in our infrastructure. All that is required is an agreement to exchange SAML (version 1.1 for now, but 2.0 in 2005) identity and attribute assertions, along with the exchange of certificates and attribute requirements for relevant applications.

We map identities from the trust domain to a single entity in our user store, which represents the trust domain itself. We add attributes to further describe the trust domain. These attributes will be added to those sent by the asserting application in the trusted partner's enterprise, in keeping with our policy ecosystem. This is actually a very reasonable trust model if you just have a few trust domains. Beyond a few you'll need the full-on trust stack – and you can expect that to get tool support in the next year. As for current federation tools I think RSA's Federated Identity Manager (FIM) leads the pack. It currently supports SAML 1.1 and provides the attribute query service we need to make our simple trust model work.

A Real-World Use Case

We have enough architecture and tools to focus on a real use case. In this use case we have a remote finance department that has surfaced a portal to allow buyers to approve invoices in accounts payable against a centralized accounting system. Picture a global company with accounting offices in each country but with centralized accounting systems – very realistic. Trust has been established in which we have set up an entry in our identity store to represent the remote finance department and receive their digital certificate such that we can verify the SAML assertions they send. Figure 2 illustrates this setup and overlays the policy ecosystem elements onto the tools we would like to leverage (note that I say "like" – this is a future-state view of the tool functions that will shortly be brought into the present…).

All that remains is to specify the security policy we would like to enforce. To make things as realistic as possible, let's assume that the remote trust domain sends an attribute that defines the amount of an invoice that can be approved by the buyer without further approvals. This means we need a policy that compares a subject attribute (the approval limit for the buyer) with an XPath expression to find the amount in the invoice document. Also assume that the buyer is required to have authenticated via hard token, and is required to have come in from the trusted domain "remote.fincence.com". Listing 1 is the XACML policy expression that defines this policy against a fictitious Web service endpoint. This example is given as a point of reference for how policy is expressed with XACML. Listing 1 takes some liberties with the resource and action targets (like ignoring the WSPL binding for Web services!). Listing 2 is the request sent to the PDP by the PEP for an authorization decision for a buyer from the remote site with an invoice approval limit of $100K. For the sake of argument, assume that there is an invoice attached with an amount of $90K – authorization = PERMIT!

Now for the reality check. It turns out that currently the fabric tools on the market do not wrap all of the PAP and PDP functions…one day I hope and assume they will. Therefore, we need to define the policy in two places – in the XML firewall IDE and in the Web services fabric IDE. Each enforces policy around their area of focus. The XML firewall does the bulk of the enforcement including policy around encryption and validation of digital signatures. In this case, we also leverage it to enforce our application security policy. If the invoice amount is $101K and the buyer's approval limit is $100K, the request will be denied. The fabric PEPs are distributed throughout the application deployment space and are used to enforce the same policies that the XML firewall enforces, but in circumstances where the XML firewall can't – such as if it was bypassed by an internal request conduit. In both cases, all policy enforcement actions, including request content and policy enforced can be logged in a configurable manner. This also affords us the opportunity to create a type of nonrepudiation log that is very important in today's regulatory climate, e.g., Sarbanes-Oxley. The DataPower XS40 can be configured to log certain requests (based on almost any imaginable rules) to a separate log that the firewall will then sign every so often. Then when the attorney general walks into the office and demands to see a validated total of all invoices paid against a certain account, you're good to go. Figure 3 shows the policy ecosystem overlays as they exist today with today's tools.

Identity Management Redux

Guess what happens to identity management in this scheme – those pesky policies that we spent so much time and money enforcing are now just additional entries in our PAP! Go ahead and expose those LDAP identity management functions as raw Web services (that's only a few days work) and let the policy ecosystem do the rest. Your application's users will be able to leverage those exposed services via the application portal and the security infrastructure will make sure that your policy desires are followed to the end brace. You can then just tell the portal team where the services are and ask them what policy to enforce and that's that!

Conclusion, and the Road Ahead

I hope I have conveyed to you the great advantage of bringing application security policy functions into the infrastructure. We saw a bit about the policy enforcement standards and protocols that we will enjoy with SAML2.0/XACML 2.0 and looked at the tools to use right now. What's coming in the WS-Security space is compelling indeed. You won't need much more than what we have gone over to do everything from applying digital rights licenses based on policy rights to all forms of media that leave as SOAP attachments, to federating instant messaging identities while applying corporate security policy. And the scary part is that even though we are focused on SOAP, these tools can talk other protocols and even bridge protocols. Think of applying XACML policy to your MQ environment and even translating it to SOAP/HTTP, and at gigabit rates! All I can say is "wow".

About the Author

Paul O'Connor is the New York region chief architect for Anexinet Corporation, a Philadelphia-based boutique consultancy. He has over 10 years of system architecture experience, having focused for the past few years on Java, open source tools, and Web services. He currently assists clients in building SOA infrastructures and applications.

■■■ poconnor@anexinet.com

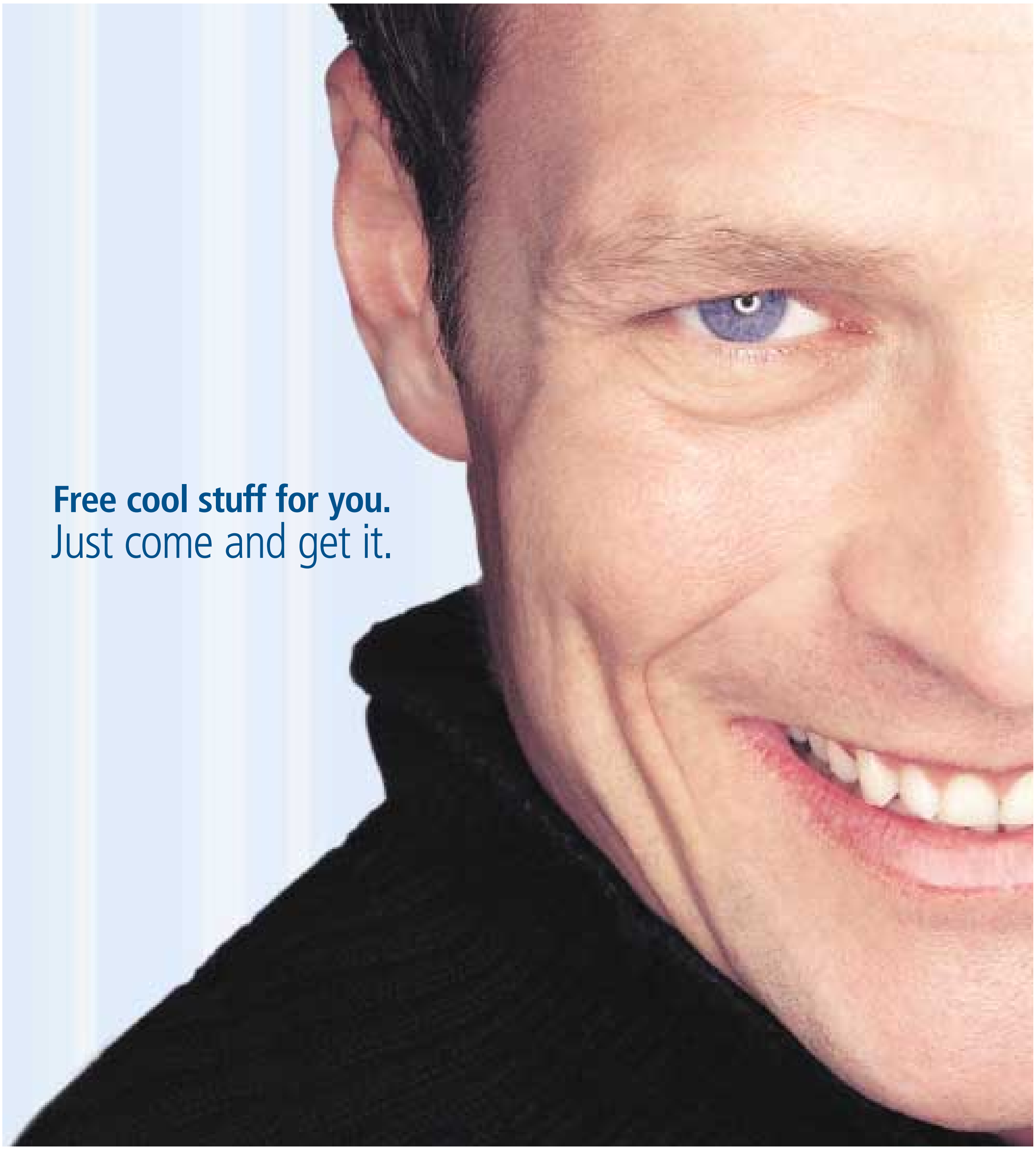
Listing 1: The XACML Policy for the Example

```
<Policy
     xmlns="urn:oasis:names:tc:xacml:1.0:policy"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy
        cs-xacml-schema-policy-01.xsd"
      PolicyId="Finance-Dept-Example"
      RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:
   rule-combining-algorithm:deny-overrides">
     <VariableDefinition VariableId="1">
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
   double-greater-than-or-equal">
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
  function:string-one-and-only">
    <SubjectAttributeDesignator AttributeId="ApprovalLimit"
     DataType="http://www.w3.org/2001/XMLSchema#string"/>
   </Apply>
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
     function:string-one-and-only">
    <AttributeSelector RequestContextPath="//
     xacml-context:Resource/xacml-
context:ResourceContent/invoice:amount/text()"
     DataType="http://www.w3.org/2001/XMLSchema#string"/>
   </Apply>
  </Apply>
 </VariableDefinition>
       <Rule
          RuleId="ExampleInvoiceApprovalRule"
          Effect="Permit">
        <Target>
            <Subjects>
                <Subject>
                    <SubjectMatch

MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
                         <AttributeValue

DataType="http://www.w3.org/2001/XMLSchema#string">remote.
    finance.com</AttributeValue>
                         <SubjectAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:1.0:subject:
    authn-locality:dns-name"

DataType="http://www.w3.org/2001/XMLSchema#string"/>
                     </SubjectMatch>
                    <SubjectMatch

MatchId="urn:oasis:names:tc:xacml:1.0:function:
     string-equal">
                         <AttributeValue

DataType="http://www.w3.org/2001/XMLSchema#string">
     Hardware token</AttributeValue>
                         <SubjectAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:1.0:subject:
     authn-locality:authentication-method"

DataType="http://www.w3.org/2001/XMLSchema#string"/>
                     </SubjectMatch>
                    <SubjectMatch

MatchId="urn:oasis:names:tc:xacml:1.0:function:
     string-equal">
                         <AttributeValue

DataType="http://www.w3.org/2001/XMLSchema#string">Buyer</A
      ttributeValue>
                         <SubjectAttributeDesignator
                               AttributeId="Role"
```

```
DataType="http://www.w3.org/2001/XMLSchema#string"/>
                      </SubjectMatch>
                  </Subject>
              </Subjects>
              <Resources>
                  <Resource>
                      <ResourceMatch

MatchId="urn:oasis:names:tc:xacml:1.0:function:
   anyURI-equal">
                          <AttributeValue

DataType="http://www.w3.org/2001/XMLSchema#anyURI">https://c
entral.finance.com/InvoiceManagement.jws</AttributeValue>
                          <ResourceAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:1.0:resource:
   resource-id"

DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
                      </ResourceMatch>
                  </Resource>
              </Resources>
              <Actions>
                  <Action>
                      <ActionMatch

MatchId="urn:oasis:names:tc:xacml:1.0:function:
   string-equal">
                          <AttributeValue

DataType="http://www.w3.org/2001/XMLSchema#string">Approve</
AttributeValue>
                          <ActionAttributeDesignator

AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"

DataType="http://www.w3.org/2001/XMLSchema#string"/>
                      </ActionMatch>
                  </Action>
               </Actions>
          </Target>
 <Condition>
  <VariableReference VariableId="1"/>
 </Condition>
     </Rule>
</Policy>
```

Listing 2: The Request

```
<Request
      xmlns="urn:oasis:names:tc:xacml:1.0:context"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:
   context
        cs-xacml-schema-context-01.xsd">
    <Subject>
        <Attribute

AttributeId="urn:oasis:names:tc:xacml:1.0:subject:
   subject-id"

DataType="http://www.w3.org/2001/XMLSchema#string">
            <AttributeValue>John Smith</AttributeValue>
        </Attribute>
        <Attribute

AttributeId="urn:oasis:names:tc:xacml:1.0:subject:
   authn-locality:dns-name"

DataType="http://www.w3.org/2001/XMLSchema#string">

<AttributeValue>remote.finance.com</AttributeValue>
        </Attribute>
        <Attribute

AttributeId="urn:oasis:names:tc:xacml:1.0:subject:
   authn-locality:authentication-method"

DataType="http://www.w3.org/2001/XMLSchema#string">
            <AttributeValue>Hardware token</AttributeValue>
        </Attribute>
        <Attribute
              AttributeId="Role"

DataType="http://www.w3.org/2001/XMLSchema#string">
            <AttributeValue>Buyer</AttributeValue>
        </Attribute>
       <Attribute
              AttributeId="ApprovalLimit"

DataType="http://www.w3.org/2001/XMLSchema#string">
            <AttributeValue>100000</AttributeValue>
        </Attribute>
    </Subject>
    <Resource>
        <Attribute

AttributeId="urn:oasis:names:tc:xacml:1.0:resource:
   resource-id"

DataType="http://www.w3.org/2001/XMLSchema#anyURI">

<AttributeValue>https://central.finance.com/InvoiceManagemen
t.jws</AttributeValue>
        </Attribute>
    </Resource>
    <Action>
        <Attribute

AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"

DataType="http://www.w3.org/2001/XMLSchema#string">
            <AttributeValue>Approve</AttributeValue>
        </Attribute>
    </Action>
</Request>
```



Global Knowledge™
Experts Teaching Experts

Want a free Apple iPod Mini?

Purchase an Application & Web Programming training course through our online store by December 31, 2004 and get a free iPod.

Visit www.globalknowledge.com/wsj to discover expert Application & Web Development training. Find the course you need and get the Cool Stuff you crave. Act now: you must purchase by December 31, 2004 to qualify for the Apple iPod.

Delivering, Administering, & Tracking Distributed Web Services

New challenges — greater flexibility

Companies that adopted and deployed early implementations of Web services primarily focused on three fundamental areas:

- Developing Web services from existing enterprise infrastructures to unlock the structured and unstructured data managed within their various systems
- Providing the necessary monitoring and management infrastructure for network-intensive Web services. This was done to guarantee delivery of messages and ensure compliance with service-level agreements (SLAs) and quality-of-service (QoS) contractual requirements
- Deploying Web services components for use within composite applications or business processes to increase system integration

These early implementations were driven by a need for greater efficiency in integrating systems, data and business processes, and in development. As developers look to expand the use of Web services and a services-oriented architecture to more systems, business processes, and users, they will face increased complexity in supporting distributed processes where humans and systems are dynamically interacting to complete processes between companies and their partners, customers and suppliers.

WRITTEN BY
SEAN MURPHY

JAGDISH REDDY

Web Services Delivery Challenges

This new service-oriented process model increases the complexity of consuming and participating in a business service as there are thousands of Web services being created and more consumers and distributors of Web services than in the traditional silo, static business process model. In short, the next generation of Web services enables companies to create an on demand infrastructure where they flexibly and efficiently distribute and consume data between internal and external processes. This new consumption model increases the complexity of consuming and participating in a business service as there are thousands of Web services being created and more consumers and distributors of the Web services than in the traditional silo, static business process model.

As the number of Web services and consumers of those Web services increases, developers will need to extend the current Web services management infrastructure to enable the secure delivery, administration and tracking infrastructure. This service delivery infrastructure enables their partners, customers, and suppliers to seamlessly and securely participate in transaction-based processes. The transaction-centric approach will see greater focus on delivering, deploying, and administering Web services across a company's value chain and enable a more flexible administration and delivery infrastructure to:

- *Manage catalogs of Web services* that enable companies to be both consumers and providers of those Web services
- *Deliver self-service and administration capabilities* to their user communities, enabling subscriber management, self-ordering, and activation to gain scalability in deploying and managing distributed processes
- *Track company and user activities* within the business services to audit, report, and bill on specific company and user activities related to the consumption of Web services

As the number of providers and consumers of Web services increases, developers are faced with the challenge of managing multiple delivery and consumption points for a single Web service. Traditionally, this challenge is addressed with the creation of a services repository or UDDI-based directory and providing a Web services management infrastructure to guarantee message integrity and delivery. This approach has worked well for creating and rationalizing the hundreds and even thousands of Web services being created by developers within a company. However, as more Web services are integrated with external processes, a more flexible delivery and administration infrastructure is required.

Developers now need to be able to enable external parties to easily consume and distribute Web services for the purposes of integrating and distributing those Web services internally and to their own business partners. To enable distributed delivery support for external consumers and distributors of a company's Web services, a company needs to integrate to a Web services delivery infrastructure. Web services delivery infrastructure enables companies to integrate their existing Web services development, deployment, and management infrastructure with delivery, administration, and tracking processes to provide a secure, reliable, distributed management and delivery infrastructure (see Figure 1).

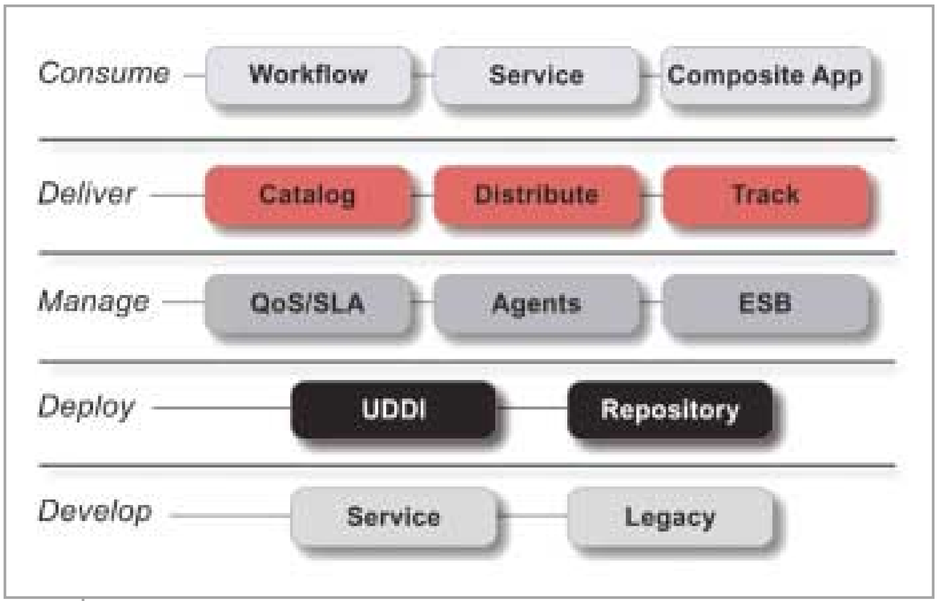


FIGURE 1 **The missing ingredients**

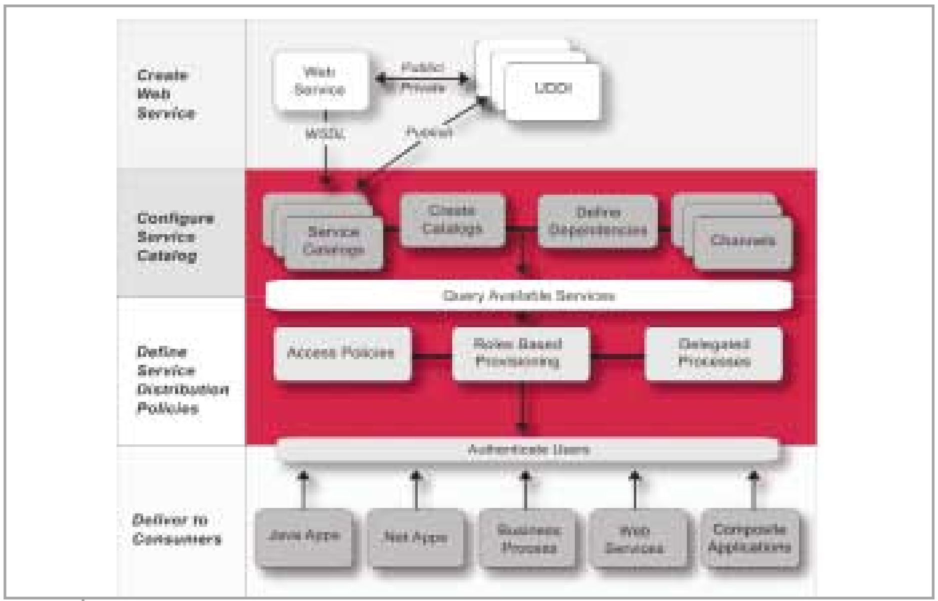


FIGURE 2 **Service delivery services**

Services Delivery Infrastructure

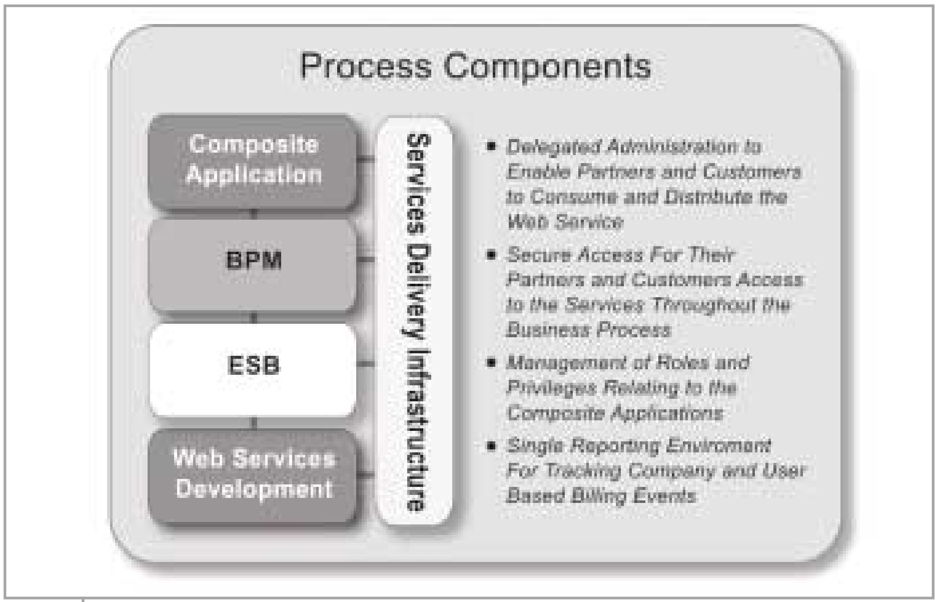
To successfully make the transition to SOA, a services delivery infrastructure requires an integrated set of processes and services from services configuration and distribution, security, *n*-tiered process delegation, and company and user-centric tracking. A service delivery infrastructure enables providers to create and delegate to consumers and distributors the capability to deliver, administer and track services and the companies and users consuming the services. To gain additional leverage from Web services, companies are looking at how to charge for accessing internal Web services as part of the service delivery process. This shift to transaction-oriented service delivery is forcing the adoption of service delivery infrastructure.

A service delivery infrastructure enables companies to manage the secure delivery of Web services through process support for services distribution, services bundling, policy definition, and access monitoring for tracking company and user activities. The major components of Web services delivery infrastructure include:

- *Service configuration tools* for defining service templates based on templates from existing repositories like UDDI, or enabling the creation of a service template if no repository is available
- *Services catalog* that enables the bundling of business services for aggregating multiple business services or discrete Web services (for example, business service is PurchaseOrder or Web services are salesorder, purchaseapproval, inventorycheck)
- *Services distribution* to enable consumption and services syndication by external parties to ensure secure distribution and delivery of services
- *Tracking company and user activity* related to distributing and access Web services from multiple providers to multiple consumers

Service Configuration

Service configuration provides a delivery and administration interface for existing service repositories (UDDI, etc.) and enables service templates to be defined if an existing repository does not exist. During service configuration, a developer or business analyst is able to easily define technical information (URL, data mapping) or transaction information such as creating specific business offers from a single service definition (e.g., service is Purchase, service offers are Internal Purchase, External Purchase, Customer Purchase). In addition, the business analyst is able to extend the service template as required for that specific company's business model. In this manner, they are able to update service metadata and have the update published to external repositories.



Process Components
Composite Application
BPM
ESB
Web Services Development
Services Delivery Infrastructure
Delegated Administration to Enable Partners and Customers to Consume and Distribute the Web Service
Secure Access For Their Partners and Customers Access to the Services Throughout the Business Process
Management of Roles and Privileges Relating to the Composite Applications
Single Reporting Enviroment For Tracking Company and User Based Billing Events

FIGURE 3 **Use case**

Service Catalogs

Service catalogs enable providers to configure services once and distribute access to those services multiple times. This "publish once distribute multiple times" model augments the existing UDDI infrastructure by providing a distribution and delivery process on top of the base UDDI information store. In this manner, companies are able to externalize the management of internal and external catalog creation to both internal and external administrators. Cataloging functions also address the productization of Web services. As companies move to deliver services as products, cataloging enables companies to create products within the catalog that are then mapped to the applicable developer-defined business or Web service. This mapping provides a process bridge between the ordering process and the usually IT-centric delivery process for provisioning companies and users to business services.

Services Distribution

Services distribution provides companies with the capability to create a secure delegated administration environment for consumers of services. Delegated administration provides companies with the flexibility required to deliver and manage Web services in distributed processes. Delegated administration enables consumers to configure services, create catalogs, and securely distribute those services to their own internal and external users. By enabling consumers to manage themselves and their own internal and external users, a service provider is able to distribute the cost of managing thousands of Web services and thousands of consumers and providers. Additionally, secure service distribution requires security not only during the publishing and consumption processes, but also at the delivery and requesting phases. Services distribution provides a critical access control for consumers requesting access to, or syndication approval for, a particular Web service.

"SOA and its associated benefits, flexibility, and efficiency, are achievable"

Tracking and Reporting

Tracking and reporting are critical components when extending Web services consumption and distribution across firewalls and between companies. Services delivery infrastructure tracks company and user activities around delivering information such as when they were granted access to services, when they attempted to access the services, and who is consuming the service. This critical information in conjunction with critical information from a Web service management tool, provides the closed loop tracking required to accurately bill for using the services, whether the consumers are internal or external (see Figure 2).

A service delivery infrastructure extends the capability of Web services management tools to include the distribution and delivery of Web services. By integrating these components as a single set of business services, a developer will reduce the time to delivery of Web services, business services, and the applicable processes. These key components are critical to ensuring successful deployments of SOA as it evolves to incorporate more transaction oriented delivery models. The point is illustrated by the following example of how a service delivery infrastructure is critical to the success of SOA.

Case Study: Process Development Tool

A process development company was looking to enable their customers, primarily in the healthcare industry, to develop SOA-based processes using their integrated tool set. They provided components for composite applications, business process modeling, enterprise service bus, Web services development and management. They began with small pilots that were internally focused. These pilots had a few Web services and a manageable number of consumers. As they started to deploy into production instances, they faced a number of challenges:

- How to enable partners and customers to securely consume and distribute hundreds of services
- How to provide distributed access to their tools for partners
- How to manage the users consuming the services via the composite application
- How to track user-based activity for billing

This company looked at how best to address the distribution and delivery challenges that were arising in a production environment. They could build this component themselves, as one-off implementations or implement a repeatable solution by including a service delivery infrastructure component. Using the services delivery infrastructure component, they integrated it with their other Web services components to provide a single infrastructure, which managed:

- Delegated administration to enable partners and customers to consume and distribute the Web services
- Secure access to provide their partners and customers access to the services throughout the business process
- Management of roles and privileges relating to the composite applications
- Single reporting environment for tracking company and user-based billing events

This infrastructure was integrated with the process components providing for single-service delivery and distribution processes across multiple composite applications. A single infrastructure meant shared components associated with secure access, roles and privileges, access policies, integrated distribution process, and process delegation. By embedding a service delivery infrastructure into their SOA toolkit, they were able to provide their customers with secure distributed process support, augment company and user tracking for subscription-based billing, integrate the user security model, and provide policy-driven access to services and resources (see Figure 3).

Summary

As companies aggressively pursue SOA to transform their businesses for greater flexibility and efficiency in selling and supporting their products, they will face challenges in securely delivering and managing the Web services in these transaction-oriented processes. It is no longer about managing the development and deployment of Web services; these challenges are being addressed with existing solutions. The next

critical area will be in the management and delivery of Web services in a distributed environment.

To date, Web services management has focused on monitoring the availability of the Web services linked to SLA and QoS compliance. As the market and infrastructure matures, demands will increase for more robust distribution and delivery infrastructure to support dynamic, flexible business models. Developers need to start addressing how to enable their SOA processes to provide for customers and partners the ability to easily distribute and deliver Web services. This infrastructure must support aggregation services, delegation capabilities, and tracking services.

Service delivery can be a significant and disruptive process for a company and its IT organization. SOA and its associated benefits, flexibility, and efficiency are achievable provided that the infrastructure is built to support not only system to system integration, but enabling integrated and distributed management of the infrastructure by internal and external users.

About the Authors

Sean Murphy is the technical evangelist for Jamcracker and its On Demand Delivery Platform. He has worked in IT for over 10 years. For the last four years, Sean has focused on enabling the secure integration of systems and processes in software-as-a-service and on demand delivery models.

■■■ smurphy@jamcracker.com

Jagdish Reddy, Jamcracker's architect, leads the architecture and development of Pivot Path, an On Demand Delivery Platform. He has over 10 years of experience in J2EE enterprise application development.

■■■ jreddy@jamcracker.com

WSJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Active-Endpoints	www.activeendpoints.com		15
Ektron	www.ektron.com/ws		25
EV1Servers	www.ev1servers.net	800-504-SURF	Cover II
Global Knowledge	www.globalknowledge.com/wsj		33
HostMySite.com	www.hostmysite.com/mxdj	877-248-4678	6
IBM	www.ibm.com/middleware/process		Cover IV
Information Storage+Security Journal	www.ISSJournal.com	888-303-5282	46
IT Solutions Guide	www.sys-con.com/itsolutions	201-802-3021	53
Mindreef	www.mindreef.com/tryout		9 & 11
Parasoft	www.parasoft.com/soaptest	888-305-0041	Cover III
Parasoft	www.parasoft.com/AchieveQuality	888-305-0041	5
Promind Systems	www.xslmaker.com		3
Secrets of the XML Masters	www.sys-con.com/freecd	888-303-5282	51
Service Integrity	www.serviceintegrity.com		23
SYS-CON Publications	www.sys-con.com/2001/sub.cfm	888-303-5282	45
Web Services Edge	www.sys-con.com/edge	201-802-3066	26 & 27
Web Services Journal	www.wsj2.com	888-303-5252	19
WebAppCabaret	www.webappcabaret.com/wsj.jsp	866-256-7973	21

Advertiser is fully responsible for all financial liability and terms of the contract executed by their agents or agencies who are acting on behalf of the advertiser. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

Above All Studio 2.0

A powerful tool for quick deployment

Much has been written recently about the business-to-business aspects of Web services, but what about the region between the Web service and the desktop? Above All Software lays claim to this "last mile" by providing a platform for delivering composite applications.

WRITTEN BY
PAUL T. MAURER

Gartner defines a composite application as "a software assembly that implements one business function (one step) and where the component parts are heterogeneous in their information architecture." I prefer to define a composite application as one that does not manage any data of its own, but brings together data from various applications around a firm to support cross-functional transactions.

I work for a large systems integrator and many of the projects I've worked on were to provide integration between application silos in the form of moving data from one silo to the other. The silos then grow warts (enhancements) to support the "integrated" data. There were times when I couldn't help but think that the integrated views should reside outside the silos. That's where composite applications come in.

Overview

Above All Studio provides a slick point-and-shoot visual environment for building these composite applications. Studio allows the user to load service descriptions (WSDL) into its dictionary, create forms that operate on these services, and combine these forms into applications.

The Basics

The Above All platform is almost entirely data driven; all data that is entered into Studio is placed within a dictionary. Services can be added to the dictionary through a wizard. The wizard installs the Web service in the dictionary as a set of objects and operations.

Each object contains a list of its elements, operations, and relationships to other objects. The elements and operations are automatically populated by the wizard from the WSDL specification, but the relationships between objects must be created within Above All Studio.

Creating relationships is done by linking output fields from one operation to input fields of another operation. This is comparable to setting up foreign key relationships in a relational database. Relationships can be visualized through an "Enterprise View" (see Figure 1). For complex applications a user can create one or more enterprise views, grouping objects into categories with cross links between views.

Forms

Once the objects have been loaded and the appropriate relationships created, the fun begins. To create a form, just select an object from either the dictionary or enterprise views and launch the form wizard. You'll need to select which operations to support and which relationships to traverse, then the wizard will generate a form.

Using the Above All–provided tutorial I was able to generate a form in a few seconds that allowed me to retrieve a list of customers, browse orders for each customer, and view shipment tracking data for each order. The form was immediately operable and fairly usable. Granted the form was not pretty or optimally laid out, but I can place the form in design mode and tweak it till I'm happy. Better yet, I can hand it over to the guys with the tinted glasses and black turtlenecks to get the "experience" just right.

But if I need the form to operate differently and massage the data in some way, I can use the Above All Studio form modeler (see Figure 2). The form modeler displays form components and the relationships between controls, dataset elements, and operations. Data flows are shown by arrows that indicate the direction of movement. Circles superimposed over the data flow lines denote a transform operation. Transforms can be fairly involved and scripted using JavaScript.

Scripting

Above All Studio does not limit the user to operations provided by imported services. Users can create custom operations that are scripted. The custom operations define inputs and outputs, and then allow the user to enter a script to provide the processing. Scripts can be powerful

Company Information

Above All Software
2755 Campus Drive
Suite 150
San Mateo, CA 94403
Phone: (650) 341-4600
Fax: (650) 341-1258
Sales: 800-819-5530
Web: www.aboveallsoftware.com

Licensing Information

Above All Studio: $4,995
End User Runtime: Varies by number of users, deployment options

Testing Environment

OS: Windows XP Professional (Service Pack 1)
Hardware: Intel Pentium M Processor (1300MHz)
1.29 GHz – 1 GB RAM

in that they can access other operations in the dictionary.

Above All also provides a robust event mechanism. ObjectTypes, Datasets, Operations, forms, and all visual components generate a variety of events that can be used to trigger predefined actions or custom scripts. Overall this is a very nice scripting architecture.

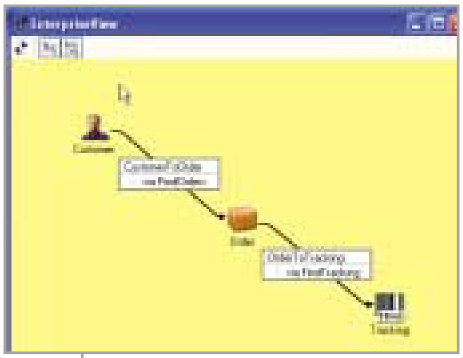


FIGURE 1 **Enterprise view**

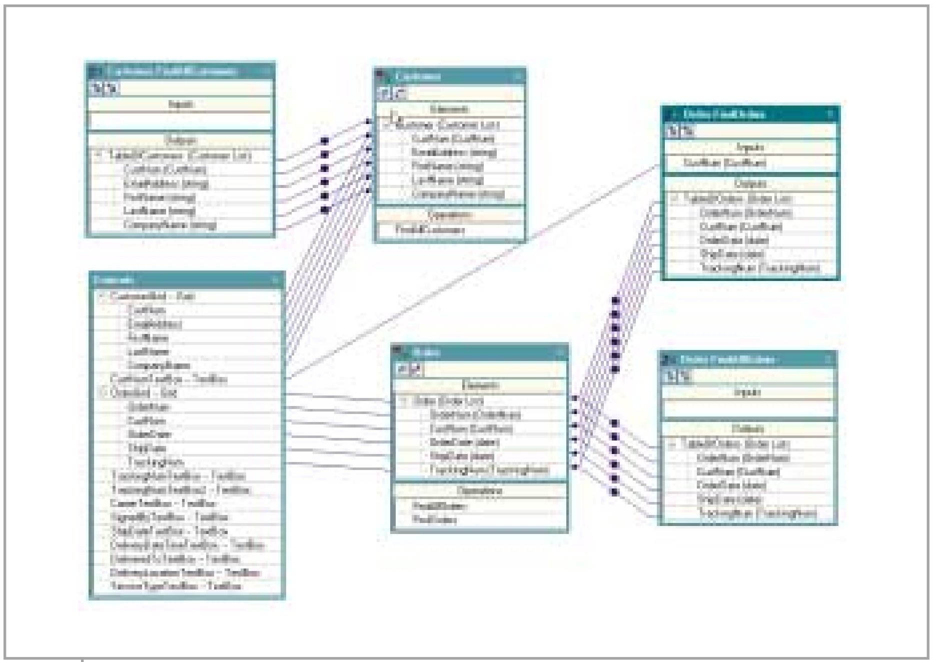


FIGURE 2 **Form modeler view**

Applications

There is the notion of an application within the product. Basically, an application is a group of forms that are linked together. One form is designated as the start form. That form is automatically opened when the application is launched. The remaining forms can be launched from the start form.

Dictionaries

I mentioned earlier that the dictionary was the core information repository for Above All Studio. One of the nice features about dictionaries is that they are stored as XML files and can be managed in any standard source code control system. Note that there are two types of dictionary. The local dictionary, which resides on your desktop, is intended solely for your personal use and the server dictionary, which resides on the network, is for shared use and supports access control. The server dictionary requires additional software from Above All Software to be installed on the server and is licensed separately.

Authentication

When using services that require authentication, Above All supports stored credentials. These credentials can be provided to the service automatically to minimize delays if retrieving data. If credentials are not set up in advance, Above All can prompt the user to enter the credentials that will be stored and used for the entire session.

Extended Services

Above All Software realizes that composite applications will need to integrate with more than Web services. They support an extended set of services through add-ins called “knowledge packs.”

Currently, Above All Software provides four knowledge packs. The ODBC pack provides connectivity to database applications. This pack supports insert, update, delete, and query facilities as well as access to stored procedures. The Oracle knowledge pack supports native Oracle connectivity and direct access to PL/SQL stored procedures modeled as Above All Studio operations. The Microsoft Office knowledge pack supports functionality to perform calculations, coordinate e-mail with customer data in other applications, and use Office functionality in forms and deployed applications.

The last knowledge pack is the most interesting. Most Web service demos provide services with straightforward interfaces. But in the real world Web services can be, and are, considerably more complicated. The application service provider salesforce.com delivers a state-of-the-art CRM platform over the Internet. Part of their integration strategy is to provide a full array of Web services that access its capability. Loading the salesforce.com WSDL into any tool would be daunting, but the salesforce.com knowledge pack automatically creates a set of higher-level business objects, operations and relationships that correspond to Web services thereby providing quick and easy start up. This is the type of add-on I would like to see more of from vendors.

Deployment

Above All Studio deploys applications as a compressed XML file. This file can be run stand-alone using the Above All application runner, or in a browser using the Above All activeX component. In the works as of this writing is the Above All WebRunner component. This component would plug into an application server and provide a true thin client to your Above All composite application.

Conclusion

Above All Studio is a powerful tool that enables a user to quickly build composite applications. Above All also provides a number of deployment options to fit most users' needs. If your organization is starting to deploy services across the enterprise, you should look into the Above All Product suite.

About the Author

Paul T. Maurer is a principal in the Financial Services practice of a leading consulting services company.

paul@paulmaurer.net

Open Applications Group (OAGi) at 10 Years

A look back and forward

Happy 10th Birthday, Open Applications Group (OAGi). This makes Open Applications Group Integration Specification (OAGIS) the most mature XML standard in the industry today! OAGi is the organization that develops and maintains OAGIS.

WRITTEN BY
MICHAEL ROWEL

Over the past 10 years, OAGi has been the standard bearer for integration language standards. This has been achieved by providing a canonical business language for integration that allows industry vertical groups and implementations to extend OAGIS to meet their unique requirements. This is all while maintaining a consistent common business language for integration. Today you will see vertical specific overlays delivered from the OAGi Web site — these are developed in conjunction with the specific vertical groups.

Critical Success Factors

One of the critical success factors for OAGi is that it does not act like a traditional standards organization, but instead acts like a development organization. Because of this, OAGi has been able to develop OAGIS in conjunction with its members at a much faster pace than a traditional standards organization. For more information on the OAGi Open Development Methodology go to the Open Applications Group Web site at www.openapplications.org.

Another critical success factor is that OAGi embraces the ability to extend OAGIS. OAGIS can be extended through scenario extensions, UserArea extensions, and Overlay extensions. The OAGIS Scenarios provide an example set of business processes in which vertical industries, and implementations may by incorporating their additions or OAGIS Business Object Documents (BODs) extend through UserAreas and Overlays. UserAreas can be used to carry simple extensions – one or two fields that are implementation specific. Overlays are used to extend OAGIS by vertical, company, or implementation.

It has been said that the saving grace of OAGIS is its ability to be extended. No other business language standard has embraced extensions like OAGi. This dates from its beginning. We're talking all the way back to 1994! OAGi views extensions as they are going to happen, whether you embrace extensions or stick your head in the sand and ignore them. In ignoring or prohibiting extensions most integrations will make use of existing fields for uses other than their original intent. You can see this in EDI where most implementations use fields for other uses.

Along with allowing extensions, OAGi works with the industry vertical group to define vertical overlays. Through this mechanism there are fewer XML standards for vendors, implementers, and customers to support. This is because these vertical overlays are based on the same standard business language. Objects are added through the use of an overlay where the vertical industry reuses existing standards and objects from OAGIS and also where more definition is needed for a given vertical.

Over the years, a lot of focus has been on the transportation protocol by the industry in general. OAGi made a conscious decision not to compete with these transport standards but rather to work with all transport mechanisms. From the beginning OAGi has realized the importance of being technology sensitive but not technology specific. This means that OAGIS works with all transports. At the end of the day OAGIS provides the structure in which information is to be shared between applications, businesses, or supply chains. This is whether you are using simple ftp, Message Orientated Middleware (MOM), service- oriented architectures (SOA) such as Web services, or ebXML.

A History of Firsts

OAGi and OAGIS predate XML Schema, and even XML itself. In the beginning OAGi defined its own metalanguage that was used in the first generation of OAGIS (releases 1 through 5).

The releases of OAGIS using the original meta-language were used by many companies. The first known implementation of OAGIS was by Sasol in South Africa. Because of this OAGIS has focused on being usable in an international setting since its beginning.

When the W3C began working on XML, OAGi already had a working group to extend the capability of this metalanguage. When this working group discovered XML, it was decided to begin prototyping OAGIS with XML (back in 1997). Based on these early prototypes, when the W3C published XML as a recommendation, OAGi published OAGIS release 6.0 using XML within a week afterward (in February 1998). This made OAGIS the first standard published in XML.

After XML was published, W3C began working on the early XML Schema. One of the inputs into XML Schema was XML Data-Reduced from Microsoft. In December 1999, OAGIS 7.0 was published in both XML/DTD and XML/XDR. This release was implemented in Microsoft's BizTalk.

On May 2, 2001, W3C published XML Schema as a recommendation. In October 2001, OAGi published the initial XML Schema version in release 7.2 along with XML/DTD and XML/XDR. In March 2002, OAGi published OAGIS release 8.0, which makes full use of XML Schema (i.e., using elements, types, and groups). OAGIS 8.0 is the fourth generation of metalanguage for OAGIS.

A Present of Firsts

Release 8.0 is the current release of OAGIS. There are many implementations of OAGIS 8.0 in the industry today. These include Agilent, Lucent, IBM, Ford, and GM, just to name a few.

For more information about OAGIS adopters, go to www.openapplications.org/adoption/usage /UsingOAGIS.htm.

OAGi

Standards Comparison

FIGURE 1 **Standards comparison**

Although XML Schema will soon be four years old, not all the tools support all the features and capabilities of XML Schema. Because of this, OAGi has published the "Practical Guide to XML Schema" to give guidelines to the features of XML Schema that must be supported. Any vendor whose tool claims to support XML Schema should support these capabilities. Also, any customer looking for tools has a checklist of capabilities these tools must meet.

This fall OAGi was the first organization to publish Web Services Description Language (WSDL) that conforms to the WS-I.org Basic Profile 1.0 for WSDL. This describes possible interfaces for all of the 65 nouns or objects and 200 BODs included in OAGIS release 8.0 and release 7.2. For more information, see the press release www.prnewswire.com/news /index_mail.shtml?ACCT=104&STORY=/www /story/09-30-2004/0002261958&EDATE.

"IBM is pleased that OAGi is evolving its rich set of integration standards to support Web services, an important technology used extensively within IBM, as well as by our customers," said Vice President of IBM Software Standards, Karla Norsworthy.

In addition, this fall OAGi and Automotive Industry Action Group (AIAG) published the AIAG version 1.0 overlay of OAGIS for the automotive OEM industry. AIAG joins STAR (Automotive Dealer) and AAIA (Automotive Aftermarket) in the automotive industry providing overlays of OAGIS. Today OAGIS is used in over 40 vertical industries and is running live in 41 countries worldwide.

A Future of Firsts

OAGi will soon release OAGIS 9.0, which continues OAGi's leadership role in providing a canonical business language for integration. Release 9.0 incorporates UN/CEFACT Core Component Types and a template for enabling the inclusion industry standard code lists that the implementation specifies. It also includes additional content from OAGi.

OAGIS release 9.0, similar to release 8.0, provides a normalized view of the content such that a single definition of components and nouns provides a consistent interface. OAGIS 9.0 will provide a flattened view or standAlone set of BODs that provides everything for a given message within a single XML Schema file. OAGi will also provide examples of using OAGIS 9.0 within service-oriented architecture (SOA), Web services, and WSDL. Looking at a comparison of prominent XML standards, OAGIS has a significant advantage and a bright future.

Therefore, Happy Birthday to OAGi and congratulations to all of its members. At 10 years old OAGi has a long and distinguished history that we have only touched on here. It continues to lead the integration community and the XML community by defining a true canonical business language for integration. OAGi's leadership role in these areas will continue long into the future. Take a look at the OAGi Web site for more firsts.

Canonical Messaging

There are many advantages to using a canonically based messaging standard such as OAGIS for application integration. Over the years, IBM has deployed a large number of custom and configured "package" applications. Integrating and upgrading these applications is challenging because applications typically have their own proprietary messaging interface definitions.

The OAGIS canonical messaging architecture defines a common set of standardized message structures (business objects) that are independent and nonproprietary. OAGIS provides a base set of objects that can be "extended" to meet the unique needs of companies or industry consortiums. IBM has extended the OAGIS messaging structures to define standardized message packages to facilitate the deployment of Web services as well as for application integration within IBM.

A canonical messaging model defines a superset of fields (elements and attributes) needed for all transactions for a particular business object (e.g., customer, product, order, etc.). This allows all applications to communicate with each other by creating simple translations to and from the business object. Message conversions from proprietary application interfaces convert to and from the common enterprise vocabulary, eliminating expensive spaghetti coded point-to-point messaging interfaces. When an application is enhanced or upgraded, its message translation code may need to be modified. All other applications should remain unaffected. This approach lowers overall application integration and maintenance costs, and it accelerates IBM's ability to deploy new services as business needs evolve.

by Patrick Rooney, Senior IT Architect, IBM CIO Architecture and Standards

About the Author

Michael Rowell is the chief architect of the Open Applications Group, Inc. With over 14 years of enterprise developement, integration and standards experience. Michael is a leader in integration and XML.

mrowell@openapplications.org

Data Services for Next-Generation SOAs

A shared data layer can meet a critical business need

This article discusses the advantages of implementing shared "data services" to deliver on the true promise of service-oriented architectures – rapid application development through reusable components without sacrificing fast, accurate enterprise data access.

WRITTEN BY
CHRIS KEENE

With a shared data layer, you can avoid integrity, performance, scalability, and availability issues that might otherwise occur.

We have entered an exciting period in the evolution of enterprise system design. More than ever, standards influence the way architects define and plan new projects. The component approach to development focuses on building blocks and provides a structure for solving complex problems. Sophisticated development tools relieve engineers of "nuts and bolts" work and allow them to concentrate more on business requirements.

I've had the opportunity to work closely with our customers as they transition into component-based technologies such as Web services and service-oriented architectures (SOAs). Their experiences highlight the importance of planning ahead for an efficient and robust data access strategy.

Because data access is such a basic requirement for enterprise development, the tendency is to pick standards such as Enterprise JavaBeans (EJB). The underlying data access may be performed with ADO, JDBC, or ODBC APIs, but it is common to leave the responsibility for database performance with database administrators. However, moving to a new architecture often means exponential growth in the demands placed on the data infrastructure – demands for increased volume and data integrity that cannot be solved in the database layer alone.

Data Access Challenges

Data access logic consumes a high percentage of development resources and plays a significant role in the success or failure of a development project. An R.B. Webber study concluded that coding and configuring object/relational (O-R) data access typically accounts for 30–40% of total project effort. Ultimately, data access logic often determines whether the resulting systems meet performance and scalability requirements.

The typical implementation of a component-based architecture makes each functional component responsible for its own data access logic. In the September 2004 issue of ***WSJ*** (Vol. 4, issue 9), Dr. Adam Kolawa confirmed this in his article's definition of application logic:

Application logic (or business logic): Handles requests from customers and agents, makes necessary connection to the database, and returns responses to customers and agents.

This is a concise description of the common architecture illustrated in Figure 1. In such systems, a request to the order application might require a database lookup of a price. In separate billing and shipping transactions, each of those applications again make their own database request. This architecture poses problems in three different areas:

1. ***The team writing each application implements similar, but slightly different, data access logic.*** Even when the data access is standards based, this low-level coding is tedious, error prone, and inefficient. The costs multiply when you add redundant testing and maintenance over an application's life cycle.
2. ***Requests that require database access are expensive.*** Each application's performance degrades when more requests come in than can be handled by the number of database connections available.
3. ***This architecture often gives more individuals access to data, opening the floodgates and creating even greater demands on the database.*** Every application handles its own data access, even when they need the same data. Databases are an expensive and finite resource. You don't want critical business functions waiting on a queue to update the database while less important requests clog the network.

As shown in Figure 2, by separating the data access out of the application logic, you can avoid these problems. These shared "data

> "it is common to leave the responsibility for database performance with database administrators"

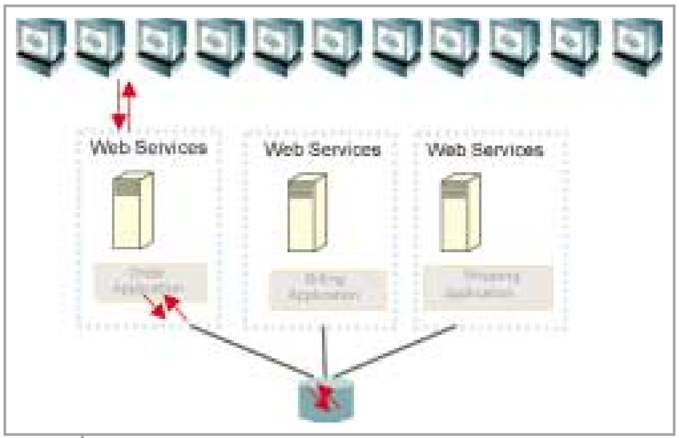


FIGURE 1 **Common architecture**

services" reduce the number of database connections required and support a stateful architecture. By caching frequently requested data, more requests can be satisfied without querying the database, which improves performance and increases scalability and reliability. (It should be noted that the effectivesss of caching varies across systems, but that your typical CRUD application will benefit nicely.) In addition, the reusability and flexibility of data services allows new services to be developed and rolled out more quickly.

However, most enterprise systems are much more complex than this example, and data integrity can become a concern. Traditional applications often have their own database "silo," which contains a copy of business reference data such as customer information, product information, and inventory levels. Typically, each database is synchronized only once a day, so each application operates with slightly different data. When applications are redistributed as enterprise services without integrating the data silos, these data inconsistencies can create unanticipated business errors.

Figure 3 illustrates the inconsistencies that can arise when silo applications are exposed as services, each with different inventory data. In this example, the "show_status" service thinks the inventory level is 27, while the "check_avail" service thinks the inventory level is 0.

Shared Data Services Enable SOA Success

An increasing number of enterprises recognize the need for a shared data service that offers domain-specific data classes used by multiple applications. Each application might use only a subset of the data classes managed by the data service. The data service manages relationships between the data classes and serves data changes to each application, regardless of the source of change.

Using the SOA paradigm, it is preferable to implement a credit card authorization, for example, as a single service that can be reused by many applications. Similarly, it is preferable to implement a single customer data service to retrieve current customer information for a set of related applications.

To be successful, an SOA initiative requires data access infrastructure software specifically designed to provide consistent performance and highly available data across distributed computing environments. Ideally, system architects should seek cross-platform data access products that are

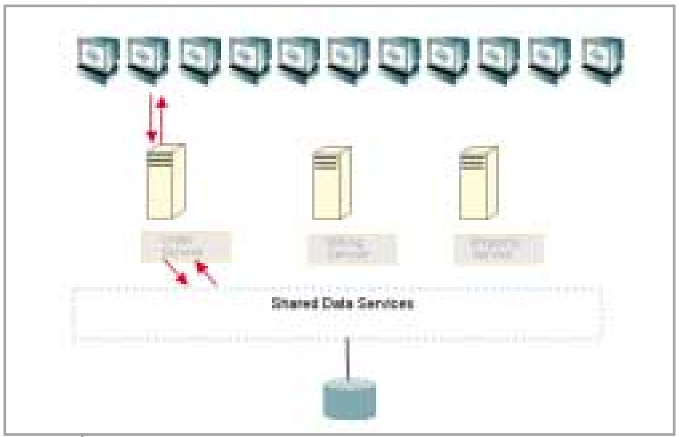


FIGURE 2 | **Separating the data access from the application logic**

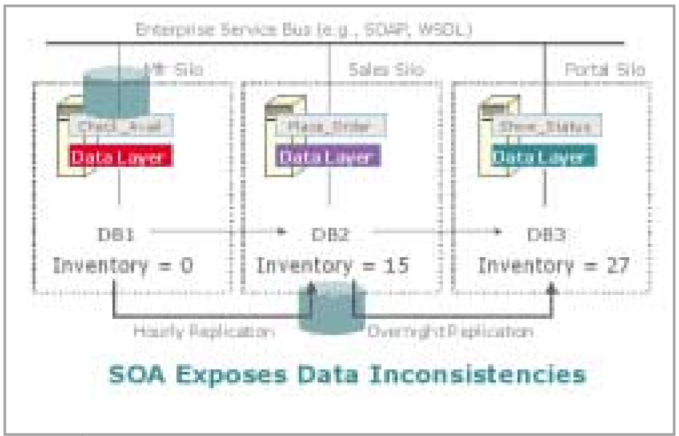


FIGURE 3 | **Possible inconsistencies**

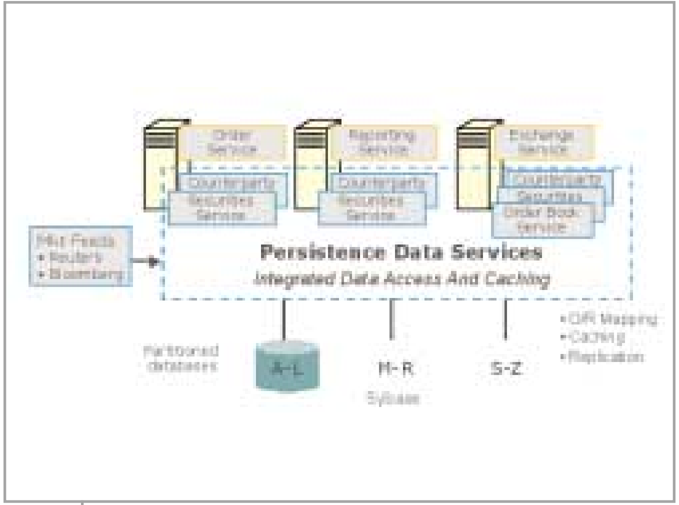


FIGURE 4 | **Case study of SOA deployment**

capable of meeting requirements across the project life cycle – from development through tuning and deployment.

From our customers' experience we've found that many organizations implement Web services or an SOA without realizing how this can increase the load on their back-end database and result in data bottlenecks. There are really three main concerns: performance, scalability, and data integrity. High-volume, complex systems require careful design of their data access to be successful.

Case Study: An SOA in Financial Services

A leading financial services firm implemented more than 40 equity trading applications on top of a shared data services layer. With rigorous requirements for reliability, performance, and scalability – up to $7 billion per day in trades and thousands of transactions per second at peak volume – they gave careful consideration to data access.

In equity trading, a single data consistency error can result in business-breaking consequences. They expected their shared data services layer to protect data integrity, deliver immediate response to end users, have the ability to scale to meet the growing needs of their businesses, and finally, to ensure 24x7 availability.

In their architecture, the data service layer provides caching, optimized updates, distributed cache synchronization, load balancing, failover, and client notification. These capabilities are far more robust than the homegrown data persistence layer used in previous generation of applications.

Figure 4 illustrates the structure of this SOA deployment. The data services layer provides data management for relational data and real-time market data feeds. Because the applications are related and share a common data model and common data, data services deliver up-to-date business information to each server and application.

The economic benefits that they realized include:

- ***Doubled developer productivity:*** Shared functional and data services account for more than 50% of new application functionality
- ***Tripled maintenance productivity:*** Systems deployed using SOA can be maintained with 75% fewer resources
- ***Dramatically higher availability:*** Fault tolerance within the data services layer eliminates application failures due to intermittent database or network failures
- ***Significant infrastructure and operational savings:*** Distributed application deployment with centralized data storage can achieve 40% capital cost savings and 30% annual operating cost savings over traditional data centers

Conclusion

Handling data access and updates accounts for the lion's share of enterprise application development efforts. Most IT groups today use ad hoc data access solutions, such as ADO.NET, that work well within a silo architecture but are unable to support data consistency enterprise wide.

A shared data layer can meet critical business needs while supplying consistent data across all applications. When designed and implemented with the appropriate development tools, a shared data layer delivers the following benefits:

- Increases developer productivity by allowing developers to focus more on business-critical logic
- Maintains data integrity when migrating existing data and application silos to enterprise services
- Ensures the performance and scalability of the deployed system

About the Author

Christopher Keene is vice president of caching products for the ObjectStore division of Progress Software. He has consistently been recognized as a software visionary, securing him a place in Intelligent Enterprise's Top Ten Technology Leaders. Prior to founding Persistence, Chris was an engagement manager with McKinsey & Company. Before that, he worked at Ashton-Tate and Hewlett-Packard. He holds an MBA from the Wharton School and a bachelor's degree with honors in mathematics from Stanford University.

■■■ ckeene@progress.com

A LIMITED TIME SAVINGS OFFER FROM *SYS-CON MEDIA*

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

AND SAVE UP TO $340 AND RECEIVE UP TO 3 FREE CDs!*

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS

3-Pack

Pick any 3 of our magazines and save up to $210.00

Pay only $99 for a 1 year subscription plus a FREE CD

- 2 Year – $179.00
- Canada/Mexico – $189.00
- International – $199.00

6-Pack

Pick any 6 of our magazines and save up to $340.00

Pay only $199 for a 1 year subscription plus 2 FREE CDs

- 2 Year – $379.00
- Canada/Mexico – $399.00
- International – $449.00

9-Pack

Pick 9 of our magazines and save up to $270.00

Pay only $399 for a 1 year subscription plus 3 FREE CDs

- 2 Year – $699.00
- Canada/Mexico – $749.00
- International – $849.00

CALL TODAY! 888-303-5282

Pick a 3-Pack, a 6-Pack or a 9-Pack

☐ 3-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.
☐ 6-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.
☐ 9-Pack	☐ 1YR	☐ 2YR	☐ U.S.	☐ Can/Mex	☐ Intl.

TO ORDER •Choose the Multi-Pack you want to order by checking next to it below. •Check the number of years you want to order. •Indicate your location by checking either U.S., Canada/Mexico or International. •Then choose which magazines you want to include with your Multi-Pack order.

☐ **LinuxWorld Magazine**

U.S. - Two Years (24) Cover: $143	You Pay: $79.99 /	Save: $63 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $39.99 /	Save: $32
Can/Mex - Two Years (24) $168	You Pay: $119.99 /	Save: $48 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $79.99 /	Save: $4
Int'l - Two Years (24) $216	You Pay: $176 /	Save: $40 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **JDJ**

U.S. - Two Years (24) Cover: $144	You Pay: $99.99 /	Save: $45 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $69.99 /	Save: $12
Can/Mex - Two Years (24) $168	You Pay: $119.99 /	Save: $48 + FREE $198 CD
Can/Mex - One Year (12) $120	You Pay: $89.99 /	Save: $40
Int'l - Two Years (24) $216	You Pay: $176 /	Save: $40 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **Web Services Journal**

U.S.- Two Years (24) Cover: $168	You Pay: $99.99 /	Save: $68 + FREE $198 CD
U.S. - One Year (12) Cover: $84	You Pay: $69.99 /	Save: $14
Can/Mex - Two Years (24) $192	You Pay: $129 /	Save: $63 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $89.99 /	Save: $6
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **.NET Developer's Journal**

U.S. - Two Years (24) Cover: $168	You Pay: $99.99 /	Save: $68 + FREE $198 CD
U.S. - One Year (12) Cover: $84	You Pay: $69.99 /	Save: $14
Can/Mex - Two Years (24) $192	You Pay: $129 /	Save: $63 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $89.99 /	Save: $6
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **Information Storage + Security Journal**

U.S. - Two Years (24) Cover: $143	You Pay: $49.99 /	Save: $93 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $39.99 /	Save: $39
Can/Mex - Two Years (24) $168	You Pay: $79.99 /	Save: $88 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $49.99 /	Save: $34
Int'l - Two Years (24) $216	You Pay: $89.99 /	Save: $126 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $59.99 /	Save: $48

☐ **Wireless Business & Technology**

U.S. - Two Years (12) Cover: $120	You Pay: $49.00 /	Save: $71 + FREE $198 CD
U.S. - One Year (6) Cover: $60	You Pay: $29.99 /	Save: $30
Can/Mex - Two Years (12) $120	You Pay: $69.99 /	Save: $51 + FREE $198 CD
Can/Mex - One Year (6) $60	You Pay: $49.99 /	Save: $10
Int'l - Two Years (12) $120	You Pay: $99.99 /	Save: $20 + FREE $198 CD
Int'l - One Year (6) $72	You Pay: $69.99 /	Save: $2

☐ **MX Developer's Journal**

U.S. - Two Years (24) Cover: $143	You Pay: $49.99 /	Save: $93 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $39.99 /	Save: $32
Can/Mex - Two Years (24) $168	You Pay: $79.99 /	Save: $88 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $49.99 /	Save: $34
Int'l - Two Years (24) $216	You Pay: $89.99 /	Save: $126 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $59.99 /	Save: $48

☐ **ColdFusion Developer's Journal**

U.S. - Two Years (24) Cover: $216	You Pay: $129 /	Save: $87 + FREE $198 CD
U.S. - One Year (12) Cover: $108	You Pay: $89.99 /	Save: $18
Can/Mex - Two Years (24) $240	You Pay: $159.99 /	Save: $80 + FREE $198 CD
Can/Mex - One Year (12) $120	You Pay: $99.99 /	Save: $20
Int'l - Two Years (24) $264	You Pay: $189 /	Save: $75 + FREE $198 CD
Int'l - One Year (12) $132	You Pay: $129.99 /	Save: $2

☐ **WebSphere Journal**

U.S. - Two Years (24) Cover: $216	You Pay: $129.00 /	Save: $87 + FREE $198 CD
U.S. - One Year (12) Cover: $108	You Pay: $89.99 /	Save: $18
Can/Mex - Two Years (24) $240	You Pay: $159.99 /	Save: $80 + FREE $198 CD
Can/Mex - One Year (12) $120	You Pay: $99.99 /	Save: $20
Int'l - Two Years (24) $264	You Pay: $189.00 /	Save: $75
Int'l - One Year (12) $132	You Pay: $129.99 /	Save: $2

☐ **PowerBuilder Developer's Journal**

U.S. - Two Years (24) Cover: $360	You Pay: $169.99 /	Save: $190 + FREE $198 CD
U.S. - One Year (12) Cover: $180	You Pay: $149 /	Save: $31
Can/Mex - Two Years (24) $360	You Pay: $179.99 /	Save: $180 + FREE $198 CD
Can/Mex - One Year (12) $180	You Pay: $169 /	Save: $11
Int'l - Two Years (24) $360	You Pay: $189.99 /	Save: $170 + FREE $198 CD
Int'l - One Year (12) $180	You Pay: $179 /	Save: $1

☐ **WLDJ**

U.S. - Four Years (24) Cover: $240	You Pay: $99.99 /	Save: $140 + FREE $198 CD
U.S. - Two Year (12) Cover: $120	You Pay: $49.99 /	Save: $70
Can/Mex - Four Years (24) $240	You Pay: $99.99 /	Save: $140 + FREE $198 CD
Can/Mex - Two Year (12) $120	You Pay: $69.99 /	Save: $50
Int'l - Four Years (24) $240	You Pay: $120 /	Save: $120 + FREE $198 CD
Int'l - Two Year (12) $120	You Pay: $79.99 /	Save: $40

*WHILE SUPPILES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE

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

SYS-CON MEDIA

XML JOURNAL SUPPLEMENT

XML-Based Interop, Close Up

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

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

Content Management:
Organization, dissemination, and presentation of information

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

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

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

Web Services Interfaces

WRITTEN BY
ENRIQUE CASTRO-LEON
& SRILAKSHMI UPPALA

Back to the future, or the revenge of the clients?

The first Web, the World Wide Web as envisioned by pioneers such as Tim Berners-Lee and Marc Andreessen in the early 1990s, was a human-to-machine (H2M) interface. The technology allowed a human using a special program, a Web browser, to access content posted on the Internet. The new protocol, known as the Hypertext Transfer Protocol (HTTP), was created along with a data format, the Hypertext Markup Language (HTML). HTML-formatted documents support the notion of Universal Resource Locators (URLs). URLs are pointers to other documents posted on the Internet, including pictures, application files, or other HTML documents. The aggregation of all documents in the Internet linked together by URLs leads to the World Wide Web metaphor.

During the 1990s, the Web morphed into a front end for enterprise applications and resulted in a self-service model that is quite pervasive today. This is the technology that allows online banking, in which a bank customer can use a browser to retrieve account balances, transfer money between accounts, or make payments. This technology also allows individual investors access to brokerage accounts to perform stock trades or company employees to change investment allocations in a retirement account.

Beyond Browser Interfaces

Despite the sophistication of a Web-based infrastructure, the front end is always a browser running in a client computer. Because browsers need to be supported on a multiplicity of platforms, browsers have a lowest common denominator for client capability (although the user interface is awkward at times for some applications). For example, scrolling a graphic, such as a map from www.mapquest.com, requires reloading the entire page from the server. This is a very slow process when the network is congested. Likewise, deleting a message from a Web-based mail client requires checking a box in a form and requesting a page reload. The user waits for tens of seconds for updates that could be done almost instantly.

The client on which the browser runs may have capabilities in terms of graphics and computation that are not used at all. Efforts to extend a browser's capabilities through plug-ins exhibit poor integration and often fail to install. Because a browser is essentially stateless, it is almost useless in offline mode beyond caching a few pages. Compare this functionality with that of a mail client. A sophisticated mail client, such as Microsoft Outlook, enables users to browse through the incoming messages mailbox, compose replies, and delete or keep messages as needed. If the same service is accessed through the Web browser interface, content is available only when the client is connected to the network.

The browser was designed as a point of consumption and is a terminal node in the network formed by the Web, with the assumption that a real human is interacting at this node.

This situation presents a challenge from the standpoint of applications, especially the way business applications are built today. These applications are rarely built from scratch. Instead, they are cobbled together from pre-existing, deployed applications, whether shrink-wrapped modules or ancient in-house built applications. There is a strong motivation against building from scratch: The cost of doing so might be prohibitive. Also, if a module is working correctly, replacing the software and the host on which the software runs does not make economic sense.

Web Services Composable Interfaces

From an interface perspective, a fundamental flaw of the first Web was a lack of support for composability. This means that once data is exposed in a Web site, it is very difficult to use this Web site as input for other Web sites, or as input to any other program. The World Wide Web makes it easy to link together documents on the Internet, but it is very difficult to build applications by composing pre-existing applications. A classic example is of a purchasing agent who works in a supply chain office in Company A and places an order of widgets to a supplier, Company B. In the early 1990s, these orders would be placed via telephone, and the seller and purchaser would enter the details of the order manually in their respective companies' client/server systems. The move to Web-based applications, where sellers post their wares in a Web portal, removed the manual intervention on the supply side. The purchasing agent in Company A could enter a purchase order in Company B's B2B portal. Increasing automation makes the manual intervention of the purchasing agent unnecessary as well: The order to purchase may be triggered by a program executing business rules at Company A. The application of these rules would be autonomous, with human intervention needed only to handle exceptions. Emerging technologies, such as radio frequency identification (RFID) and competitive pressures, will make this level of automation a necessity.

A naive way of automating the purchasing transaction described here is to have a program running in Company A parse Company B's Web site and provide the same responses that a human would have provided. Needs arising from the previous example resulted in the machine-to-machine (M2M) Web, also known as Web services.

The existing protocols were insufficient. The naive way of composing Web sites falls short. HTML carries formatting constructs that provide a pleasing presentation to human browsing. There is no provision for incorporating semantic components in the parsing of a Web site. An automaton could be programmed to parse a Web site "right" the first time, but if the Web site owner makes even a small change, the original parser could be thrown into utter confusion. For instance, if a number in a certain field is changed to represent kilometers instead of meters, the numbers retrieved will be off by a factor of 1,000 and the parser won't know.

A number of alternatives are possible. One would be to use a distributed computing framework, such as Common Object Request Broker Architecture (CORBA) and Distributed Component Object Model (DCOM). At issue here is interoperability. If Company B publishes a DCOM Application Programming Interface (API), whereas Company A uses CORBA, their systems will not communicate. Yet another alternative is the use of Electronic Data Interchange (EDI) technology. However, EDI is less than satisfactory. The overhead of setting up an EDI link is high, and an EDI link amounts to an inflexible, private agreement between two companies using proprietary technologies.

"HTML carries formatting constructs that provide a pleasing presentation to human browsing"

Web services have been implemented as a series of layered protocols. At the lowest level, M2M interactions under Web services use Extended Markup Language (XML)-formatted data using a newer protocol, Simple Object Access Protocol (SOAP). SOAP can be piggybacked on top of HTTP. The semantic component in Web services is attained by industry agreement in which members of an industry vertical agree on the meanings of certain XML constructs.

A Web services–enabled portal, in addition to having a traditional Web-based interface designed for use by humans through a Web browser, also exports a Web services API. Most well-known portals, such as www.amazon.com or www.google.com, already do.

The bindings to a Web service are self-describing. Thus, items such as matching arguments by type and number and some of the semantics can be done immediately prior to or at invocation time. Long-winded negotiations about the meanings of every construct are not necessary. Presumably all these issues would have been hammered out at the standards committee level.

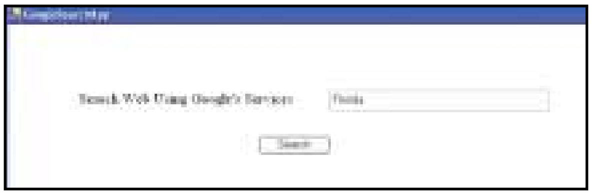

Search Web Using Google's Services
Florida
Search

Figure 1 • Self-standing Web application

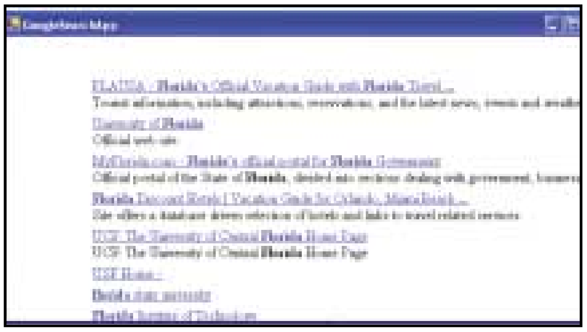
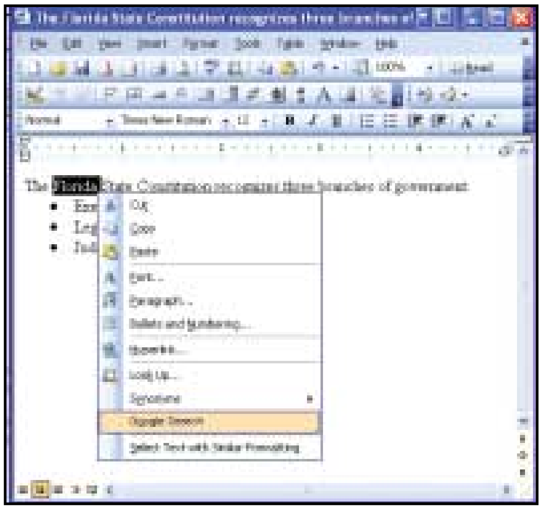

FLAUSA - Florida's Official Vacation Guide with Florida Travel ...
Tourist information, including attractions, reservations, and the latest news, events and weathe
University of Florida
Official web site
MyFlorida.com - Florida's official portal for Florida Government
Official portal of the State of Florida, divided into sections dealing with government, busines
Florida Discount Hotels | Vacation Guide for Orlando, Miami Beach ...
Site offers a database driven selection of hotels and links to travel related services
UCF: The University of Central Florida Home Page
UCF The University of Central Florida Home Page
USF Home -
Florida state university
Florida Institute of Technology

Figure 2 • Application displaying information from the Google Web service

Examples

Unlike a Web browser, which out of necessity is a general-purpose program, a program using a Web services API brings infinite potential. Plus, it can be as rich, if not richer, than earlier client programs. What follows are some new capabilities:

1. The client application can provide a Web browser–like interface to a user interacting with the client. Although possible, this is not very interesting in most cases. This setup might be of interest in a multilingual environment, for providing access to impaired people or to apply content filtering and security policies.
2. The implementation of applets, such as time of day or stock tickers, becomes trivial. All it takes to implement an applet is a call to the Web service that provides such information. The information can be displayed in traditional fashion in a box inside a browser. A browser is not essential anymore. The applet can even run inside an embedded device. A self-adjusting wall clock with a WiFi or wired Ethernet connection can make a periodic call to a time Web service to ensure there is no drift. The display is not a browser; it just looks like a regular clock.
3. Implement a Web-based application. This is different from a Web-enabled application where a Web browser interface is added to a pre-existing application. The distinction between a client and a server gets blurred. For instance, the www.amazon.com Web services interface allows independent booksellers to partner with Amazon and integrate their offerings with Amazon's Web site.
4. The implementation of an ultra-reliable service is possible through transparent failover of independent, redundant service providers, each less than perfect if taken separately. A combination of them can be used to bring up uptime levels.


The Florida State Constitution recognizes three branches of government

- Exe
- Leg
- Jud

Figure 3 • Microsoft Word as an example of a Web services–based compound application. Right-clicking on a text selection exposes an operation to do a Google Web services search.

Web services make a computer "democracy" possible, with the potential for each device to play its capabilities to the fullest and with a simple way to request supporting functionality when needed. The technology allows apportioning the demands of a distributed application to the capabilities of available devices. For instance, when an application is accessed through a PDA, the system can allocate some computations to be done by a distributed service instead of the client. On the other hand, accessing the same application from a powerful client can tilt the balance toward local computations for fast response and to support sophisticated rendering.

With Web services, applications designers can tailor the application exactly to the needs of the application instead of having to force-fit the application to the quirks of the Web browser interface. The use of local computation resources generally can make the interactions much faster. Changes can be transparent to the user with little required in terms of a learning curve. Let's illustrate this with a few example applications.

Grid Computing

Web services–based client programs enable not only data sharing, but also the sharing of practically any resource, including computation. Traditionally, access to a cluster has taken place through a remote log in to a front end computer in the cluster, in which data and programs are transferred manually using a file transfer protocol (FTP) utility. A Web services infrastructure will allow providers to deliver virtualized grid services. The service provider would function as an aggregator for third-party grid services. A subscriber to the grid services is presented with a Web services–enabled client program that finds an appropriate host to run a specific job based on the characteristics of the job. This client program functions as a broker for computation and data services. As conceived initially, grids enable offloading the execution of long running jobs. Under a grid infrastructure, programs that would take hours or days to run locally can be rerouted anywhere in the world to be run in a cluster with the features—in terms of performance and security—desired by the requester. Standards-based middleware that does the handshaking or interface between the requested and the target execution platform, such as the Globus Toolkit, makes heavy use of Web services. Although XML has helped enormously in attaining data portability, the delegation of application execution is harder to achieve than data when the executable programs are sent in binary form. Rewriting the applications to run on a virtual machine will help in terms of compatibility, but doing so entails some overhead. This will become less of a concern as CPU cycles become less expensive. Techniques such as library substitution for the performance-intensive portions of an application can be used to minimize the impact of this overhead.

Online Phone Directories

Prior to 1995, it was not uncommon for the online form of the corporate phone directory to be distributed on a floppy disk. When installed on a PC, the program on the diskette would become the user interface for the phone directory application running on the server. The protocol between the client application and the database running on the server was likely proprietary. Unlike Web enabled applications today, which tend to be sluggish, these applets were actually quite responsive. Applications with similar look and feel are practical in a Web services–based reincarnation. Instances of the application would run on any device imaginable, be it a desktop, laptop, third-generation wireless device, or PDA. The application would also take care of online or offline access and automatic directory updates.

To illustrate the subjective performance difference between a Web-based interface and a rich client interface, do a directory lookup in a well-known portal such as www.411.com. The response time is fair. Next, the reader is asked to do a directory lookup in the address book of a rich client, such as Microsoft Outlook. The entry is found almost instantly because there is little display information to refresh, and the directory database is cached. If the application is providing the interface, it is possible to design each instance to match the capability of the client device. A laptop version could cache the database for offline access, whereas a PDA or a cellphone version could be designed to minimize the screen real estate. There won't be a need for a heavy-handed approach like creating a new industry HTML standard just for wireless applications.

The Web Google Versus the Web Services Google: "Look, Ma – No Browser"

As a technology leader, the Google Web site was one of the first sites to also publish a Web services interface. It is now possible to write a stand-alone, non-browser–based program that requests the Google service to do a search on a specific keyword. The look of the result is quite different from a "normal" Web search because the formatting is actually provided by the application.

In this example, instead of using the browser, a simple self-standing application is used to enter a string and invoke the Google Web service to do a Google search on that particular string. For example, entering "Florida" (Figure 1) yields the results shown in Figure 2.

A more interesting example is a compound application. An example would be an extension to the Microsoft Word application that invokes the Google Web service on a text selection by right-clicking on that text selection. The output of the search would be the same pop-up window of the stand-alone Web search program (see Figure 3).

This technique could be applied to invoke any service imaginable: dictionary and thesaurus, today's date, or clip art.

Conclusion

Web services represent an evolution of the original Web because the Web becomes accessible by any program and not just a browser. The Web is no longer just a front end infrastructure for an often complex Web enabled–application. An existing Web site can be retrofitted to publish a Web services API, simplifying programmatic access to the facility enormously. This has happened with well-known sites such as www.amazon.com and www.google.com. Programmatic access to Web functionality encourages building applications by composing services. This new paradigm for building applications also changes the user interfaces, potentially much richer than are possible with a browser, and resembling the richness and responsiveness of traditional client applications. This new paradigm also allows the developer to optimize how and where computations in a distributed system are performed. This enables the design of a system with the best user experience regardless of the client equipment, whether handheld, mobile, or desktop.

Author Bios

Enrique Castro-Leon is an enterprise architect and technology strategist architect consultant with Intel Solution Services (www.intel.com/go/internetservices/intelsolutionservices). He holds master's of science degrees in both electrical engineering and computer science and a PhD in electrical engineering from Purdue University. Enrique is a cofounder and director of Neighborhood Learning Center, a nonprofit organization providing education and tutoring services to K–12 students and senior citizens.

Srilakshmi Uppala has a bachelor's degree in electrical engineering from India and master's in computer engineering from the University of Kansas. She has been with Intel for more than four years as a software developer and has expertise in Microsoft technologies including .NET, SQL Server, BizTalk, and Share Point Portal Server as well as XML and Web services. In her current role as a technical consultant with Intel Solution Services, Srilu is able to work with end customers in architecting and implementing cutting edge solutions.

ENRIQUE.G.CASTRO-LEON@INTEL.COM

SRILAKSHMI.UPPALA@INTEL.COM

WRITTEN BY **PG BARTLETT**

Why Publishing is Getting More Complicated and Costly

A look at the problems of capturing and sharing information

Are you just beginning to try to figure out how XML and content management can help you wrestle with your publishing problems? Are you confused by all the jargon and acronyms that the experts are throwing around? Do you wish that someone would clear it all up for you? If so, then this 4-part series of articles is for you. When we spoke recently with someone who is just beginning to find his way through the standards, technologies, products, and vendors related to XML publishing, we were struck by the lack of helpful information resources for the uninitiated.

In this first article, we discuss the practical problems of capturing and sharing information that you and many others are facing today. This means problems such as delivering information to multiple types of media, making updates faster and easier, and reducing the cost and time to translate and publish.

On occasion, each of us has wasted way too much time fooling around with documents. With the advent of WYSIWYG ("What You See Is What You Get") word processing and desktop publishing software in the 1980s, everyone became a publisher. Who among us has not experienced the gratification that comes from handcrafting the appearance of a document until it looks absolutely perfect?

But how much does that freedom and control cost? When authors have been relieved of the responsibility for decorating documents, their productivity soars. We have repeatedly seen improvements in output of 50% to 100% among full-time authors.

There will always be a role for software that gives you fine control over the appearance of every page you produce, but many types of information could be automatically published, such as the following:

- Catalogs
- Datasheets
- Technical manuals
- Instruction manuals
- Service/troubleshooting guides
- Training courses
- Reference publications
- Journals/periodicals
- Research reports
- Policy/procedure manuals
- Manufacturing instructions
- Regulatory submissions

In speaking with many people who are responsible for capturing and sharing information, both inside and outside their organizations, several common themes emerge.

Too Much Work

The most common complaint is that it simply takes too much manual effort to create, review, update, and publish information. Most people responsible for publishing feel enormous pressure to deliver more quickly, with greater accuracy, in more forms, and with greater customization. But there's no way they'll be allowed to raise spending to do so.

Difficult Reuse

Everyone recognizes the value in creating a "single source" of information: greater accuracy, higher productivity, improved consistency, easier and faster updates, and no worrying about which version is correct. But today's tools and processes make reuse very difficult, especially when you want to be able to incorporate the same information in different documents with a different appearance.

Some of you also want to be able to make some fields in your documents contain "live" data from databases or other business systems, which would save the cost of updating that information every time it changes.

Expensive Translations

Related to the reuse problem, many companies waste millions retranslating information that does not change. Translation companies charge for all the words you send them, even if they can reuse previous translations, so the costs add up fast. We've repeatedly seen that when organizations reduce translations to only the content that has changed, that benefit alone pays for the cost of overhauling their content creation and publishing systems.

Multiple Tools

Some people use word processors, oth-

AUTHOR BIO

PG Bartlett is Arbortext's vice president of Product Marketing. He is responsible for building and executing a unified product and applications marketing strategy. Since joining Arbortext, PG has hosted electronic Webinars on topics ranging from "What is XML?" to "Enterprise Content Management and Single Sourcing," and authored many white papers and articles. He is a frequent presenter at major industry events.

ers use desktop publishing software, and some even use simple text writers. All of these prevent the loss-free transfer of information from one person to another. The lost time converting documents from one format to another and reconstructing the lost formatting continually drains productivity and enthusiasm.

Handcrafted Publishing

Time lost to decorating documents and squeezing paragraphs is almost incalculable, and it undoubtedly represents a significant drain on corporate productivity. In many cases, the waste is compounded when the same information is formatted multiple times on multiple tools.

For example, we've seen situations in which engineers contributing content to tech manuals not only use a word processor to create their content, but also spend time decorating it. This occurs even though the tech writing group discards all of their careful formatting when they aggregate the engineer's content into the rest of the book.

Publishing to Multiple Media Types

For organizations that must produce their information in multiple forms, the cost of publishing mounts quickly. In many organizations, entire groups exist solely for the care and feeding of each type of media that the organization supports. Because the usual process involves converting information published for one medium (e.g., print) to another (e.g., Web), keeping the information consistent and up-to-date is nearly impossible.

Improving Information Quality

Those responsible for producing content find themselves under continuous pressure to improve the quality of the information they create. Whether the information is for use inside or outside the organization, it is not sufficiently accurate, consistent, fresh, and complete, or else it has cost an enormous amount of time and money to create and maintain.

Customized Information

Your customers want information that's more relevant to their needs, and they have learned from the best sites on the Web that the technology exists to tailor information precisely. However, in most cases, you publish "one size fits all" documents that contain considerable irrelevant information for any one user.

Summary

Next time, we'll look at the key elements of addressing this pain.

We will offer the basics for solving publishing problems, including automating the publishing process, enabling the reuse of information, and using XML as your content format.

PGB@ARBORTEXT.COM

WRITTEN BY **BRIAN BARBASH**

Adobe FrameMaker 7.1

Adobe Systems Incorporated
345 Park Avenue
San Jose, California 95110-2704
Tel: 408-536-6000
Fax: 408-537-6000

Powerful capabilities for working with XML documents

XML's surface-level simplicity hides a deceptively complex beast. At first glance, creating an XML document does not take a lot of effort. Create some tags, ensure they are well-formed, and that's it. Throw in a DTD or Schema and now there are a set of rules against which the document can be validated. There is an innumerable amount of XML documents created in the world today in this manner. And for many scenarios, this is a perfectly reasonable approach.

But XML is so much more than that. Among other things, XML promises and delivers the separation of many concerns. The rules governing the structure of a document may be separated from the document; a document's presentation and formatting information may be independent of its content; and XML documents are not tied to any single platform. These are but a few of the benefits of the technology. However, fully realizing these benefits can become extremely complicated.

Take content and presentation separation as an example. In many ways, XSL Stylesheets achieve this goal. However, they can quickly grow to contain complex and convoluted logic even for relatively simple formats. Achieving the results worthy of professional publications, including image manipulation, cross-references within documents with large volumes of text can be all but impossible. This is where Adobe FrameMaker 7.1 steps in.

FrameMaker is an authoring and publishing solution that brings the benefits of XML to the publishing world. Incorporated into its standard feature set is the ability to author and publish content based on the structured data within an XML document, while maintaining the business rules associated with the content of the document. For the purposes of this review, only the XML components of FrameMaker will be addressed.

Working with Structured Documents

Structured documents are any documents that follow a predetermined hierarchy. Editing structured documents in FrameMaker requires set up of several standard supporting files, a few of which include the following:

- ***Structured template***: Document that defines, in FrameMaker terms, the elements and attributes of a document. This is comparable to a DTD or XML Schema but specifically for FrameMaker.
- ***XML DTD***: Standard DTD associated with an XML document.
- ***Import and export (read/write) rules***: This document provides the translation between XML and FrameMaker. This is loosely analogous to an XSL Stylesheet for transforming XML tags to FrameMaker document elements.
- ***EDD***: The Element Definition Document defines the FrameMaker-specific structural rules and formatting information. In a way, this acts as a secondary XSL Stylesheet for transforming XML tags to FrameMaker document elements.

It is important to note that when editing structured documents in FrameMaker, the author is not working directly with raw XML. Only when the document is exported and transformed according to the import/ export rules is XML actually generated.

Figure 1 shows an example of the structured FrameMaker environment. Pictured is the setup for this article. (*Note:* In the interest of full disclosure, this article was authored in both MS Word and Adobe FrameMaker. However, it was submitted to the magazine's editors in Word format.) Visible in the left portion of the screen is the EDD for the document. In it appears the FrameMaker element names, the subelements that are valid for each, the cardinality for the occurrence of each, and the relevant text formatting rules. In the middle of the window is the text of this article as it appears after all formatting rules have been applied. The top half of the right side of the image shows the XML hierarchy of the article itself; the raw XML of which is shown in Listing 1. The bottom half acts as a contextual guide listing what elements are available for placement into the document at the current insertion point.

Writing This Article

In working with the tool, it was clear that the majority of the effort of working with XML centers on three main tasks:

1. Setting up the FrameMaker application definition
2. Establishing the read/write rules for the XML document

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

3. Creating and applying the transformation and formatting rules

An application in Adobe FrameMaker is a series of configuration settings that establish a general environment for working with content. Typical settings include the location and filename of the read/write rules, the location and filename of the document's DTD, the FrameMaker template used for new XML instance documents, and the XML elements that may appear as the DOCTYPE within an XML instance document. FrameMaker ships with several preconfigured applications and supporting files for working with DocBook XML documents, XHTML, XDocBook, and DocBook 2.1.

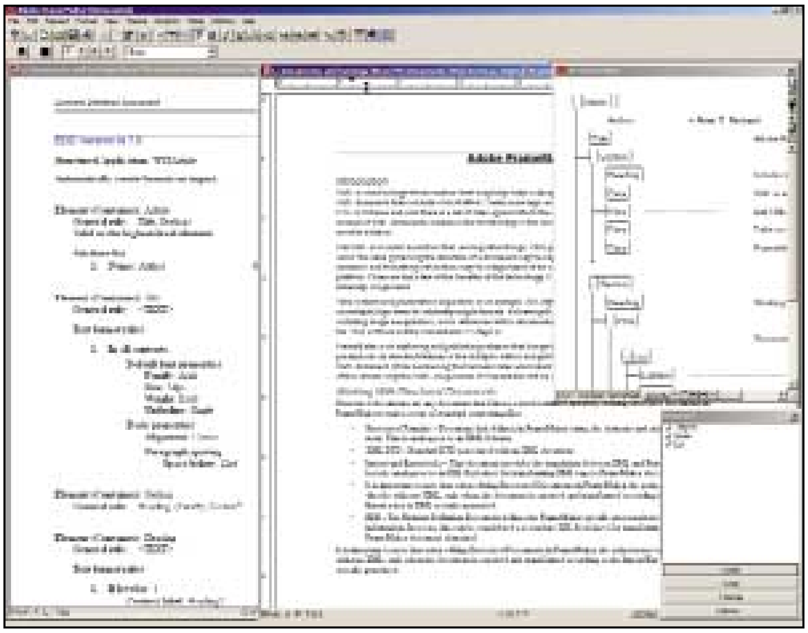


Figure 1 • FrameMaker environment

For the purposes of writing this article, I have created a simple DTD that supports the desired structure. As a result, I've created a new application definition that references the DTD.

The second part of the process is to define the read/write rules for the XML document. Read/write rules act as a kind of XSL Stylesheet to translate the elements of an XML instance document to the elements of FrameMaker. Read/write rules may also be used to define how attributes are assigned to elements. For example, the DTD for this article includes a graphic element for figures. Assigned to it is an attribute that contains the type of image used and the allowable values: JPEG, SVG, GIF, and so forth. FrameMaker presents this attribute to the user in the form of a dialog box with a drop-down menu containing the predefined enumerated attribute values.

Finally, the third part of the process is to create the transformation and formatting rules. This document also acts as a kind of stylesheet. However, it deals with the resulting FrameMaker elements after they have been converted via the read/write rules. In it, all the formatting rules and special handling rules necessary for the document's elements are defined to reach the formatted result. The example created for this article is very simple. As seen in Figure 1, it assigns some basic formatting properties to the article's title, section headings, and body content. Because this rule document is independent of the XML content, any number of variations may be created and applied easily for different results. All of the characteristics available to a FrameMaker element may be accessed in this document. Therefore, it gives users access to the full suite of the tool's standard authoring and publishing capabilities.

Once all of the setup is complete, it is just a matter of entering or importing the content into the appropriate sections of the document. As data is entered, all formatting and structural rules for the presentation of the content are applied, while the integrity of the DTD and read/write rules are checked and maintained on the fly.

Summary

Adobe FrameMaker 7.1 provides a powerful set of capabilities for working with XML documents. Although this is a product that is geared to the authoring and publishing audience, its capabilities may be used to bridge into the technical world of XML. It delivers on the promise of separating presentation from content and allows those with expertise in each area to work independently. This is a sophisticated solution for a complex problem.

BBARBASH@SYS-CON.COM

LISTING 1 • Article text in XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE article SYSTEM "article.dtd" [

<!-- Begin Document Specific Declarations --
>

<!-- End Document Specific Declarations -->

]>
<?xml-stylesheet href="article.xsl"
type="text/XSL"?>

<?Fm Condition Comment Red SINGLE_UNDERLINE
show?>
<article author = "Brian R.
Barbash"><title>Adobe FrameMaker 7.1</title>
<section><heading>Introduction</heading>
<para>XML is a technology whose surface-
level simplicity hides adeceptively complex
beast. At first glance, creating an XML doc-
umentdoes not take a lot of effort. Create
some tags, ensure they arewell-formed, and
that's it. Throw in a DTD or Schema and now
thereis a set of rules against which the
document can be validated. Thereis an innu-
merable amount of XML documents created in
the world today
```

WRITTEN BY **D. ROBERT ADAMS & DANIEL WILLIAMS**

Visiting the DOM

Extending the Visitor Pattern

It is well known that traversing the XML DOM is a sometimes difficult and often tedious task. Executing code based on data retrieved from the DOM is even more complex. This article will demonstrate one way to abstract much of the logic from this repetitive task. The implementation of patterns is a technique that is often used to help simplify and intellectually manage projects, and the Visitor Pattern is appropriate and useful to help solve this problem. Reflection also plays a key role, and is used to determine the executed code based on the DOM node names at runtime.

AUTHOR BIOS

D. Robert Adams' current research areas include XML, Web applications, and distributed computing. He holds a PhD in computer science from the University of Kentucky

Daniel Williams received his MS degree in Computer Information Systems from Grand Valley State University, where he developed the infrastructure for applying the Visitor Pattern to the DOM. His professional interests include Java, 3D data processing, and Microsoft .NET technologies. He currently works for Fishbeck, Thompson, Carr, and Huber in Grand Rapids, Michigan.

XML has proven itself to be useful for storing the data for many types of applications (www.oasis-open.org). However, when you get down to it, XML is just a highly structured text file. Applications that use XML as their input format must process the XML file in order to get the data from it. Currently, there are two primary APIs that applications can use to traverse an XML file: DOM and SAX.

DOCUMENT OBJECT MODEL

The Document Object Model (DOM) provides the most obvious way of accessing data in an XML file. An XML document is essentially a tree: the root element of the XML document is the root node of the tree, and child elements in the XML document are the children of the root node. The DOM API provides an application with a tree data-structure that directly mirrors the XML file. The DOM API provides the ability to traverse the tree from one node to the next: parent-to-child, child-to-sibling, child-to-parent, etc.

Application developers use the API to write code that traverses the tree and performs processing on the tree's nodes (e.g., extract some piece of data). The trouble with the DOM API is twofold. First (and obviously), application developers must write the code that traverses the tree. However, this is often an unnecessary re-invention of the wheel. Many applications traverse the tree in a standard depth-first approach. It is unfortunate that the developer must spend time writing the actual tree-walking code, when it has been written before by countless developers.

The second drawback of the DOM API is the integration of the tree-walking code with the node-processing code. One of the hallmarks of good software design is the separation of concerns. This means the developer tries to separate the logical pieces of the code into separate methods/functions/modules. The unfortunate design of the DOM API requires that the complex code that walks the XML tree must be interspersed with the, quite possibly, complex code to process individual nodes.

The following code fragment illustrates the complexity of processing XML nodes using the DOM API. In this example we traverse an XML tree grabbing all the "color" elements and swapping the black and white ones.

```
demoChildNodes =
document.getElementsByTagName("colo
r");
for (int i = 0;i <
demoChildNodes.getLength(); i++) {
if(demoChildNodes.item(i).getFirstC
hild().toString().equals("black"))

demoChildNodes.item(i).getFirstChil
d().setNodeValue("white");


if(demoChildNodes.item(i).getFirstC
hild().toString().equals("white"))

demoChildNodes.item(i).getFirstChil
d().setNodeValue("black");
}
```

Simple API for XML (SAX)

The Simple API for XML (SAX), first published in 1998, was developed as an event-based API for processing XML. Hot on the heels of the XML specification itself, it was well received by the development community.

SAX works by processing the XML one node at a time, creating a streaming process opposed to a static one. Event handlers are added to an XML document much in the same way they are added to a user interface. Such events are triggered as the application processes the document. Using this technique creates a very fast processing method, with a much smaller memory profile. This makes SAX appropriate for tasks such as searching a document for a specific node, or making small change to the entire document such as search and replace.

The down side is the lack of directional control, the document can only be process red in a "top to bottom" direction. Tasks like reordering nodes and cross-referencing are not practical. Also, while the specific syntax for simple documents (or simple processing tasks) is not overwhelming, SAX does not scale well for more complicated solutions. Listing 1 is a short example (the code for this article is online at www.sys-con.com/xml-j/ sourcec.cfm).

The other drawback to SAX is the granularity of the event handlers. SAX is "coarse-grained" in the sense that large structural XML components invoke the same handler. For example, all "element" nodes would invoke the startElement() method. Many applications require a "finer-grained" event handler. For example, invoke a specific method when an "employee" element is reached.

Visitor Pattern

The problem of separating data-structure traversal from data-structure analysis/process is well-known. One standard technique of ensuring a separation of these components is the Visitor Pattern. One way of understanding the Visitor Pattern is that it allows operations to be added to a class (the pattern is an object-oriented one) without having to actually change the class. However, in the context of tree-walking, the Visitor Pattern is best understood as providing the capability of applying an endless number of node processors to a tree without having to change the definition of the node itself: it separates the node-processing code from the node definition.

For example, assume we have a tree data structure encapsulated in a tree class. This class defines a root node class that has links to its child nodes. Using the Visitor Pattern, we can define a MyVisitor class that has a method visitNode() that performs some kind of processing at each node in the tree. However, the Visitor Pattern allows us to focus the method on the node-processing code itself, and not the tree-traversal code. Once these two classes are defined, we could "apply" our visitor to our tree with the code: my Tree.accept(my Visitor); This would start a depth-first traversal of the tree and at each Node the My Visitor.visitNode()method would be called, allowing us to apply processing to each node.

In the future, if we want to update the processing, we would only need to modify the MyVisitor class. The tree and node classes would remain untouched. If we want to perform a different kind of processing on the tree, we would simply define a MyOther Visitor class with a visitNode() method and then apply our new visitor to the tree with: myTree. accept(my Other Visitor). Once again, the existing tree and node classes are untouched.

"Extending the DOM API to support the Visitor Pattern means that XML application developers are freed from writing tree-traversal code"

Visiting the DOM

We have developed a prototype DOM visitor in Java that supports the Visitor Pattern. The implementation is quite straightforward. A visitor-adapter class encapsulates the tree-walking code and provides for standard depth-first traversal. Obviously, this class can be overridden to provide for more complex tree traversals.

The other major component of the prototype is the alteration of the NodeImpl class and the Node interface inside org.w3c.dom, which provides for the implementation of a DOM Node. The only change necessary is the addition of an accept() method that takes a Visitor class as a parameter and invokes the appropriate visitXXX() method in the Visitor class (where XXX is the name of the element). The accept() method uses reflection to determine which method to call in the visitor. Using reflection, the visitor Pattern provides the kind of "fine-grained" visitation that most applications require.

Another benefit of the Visitor Pattern is that it combines useful features of both DOM and SAX. The Visitor Pattern is "event-driven" to the extent the specific methods are called when specific nodes are reached in the tree. Furthermore, within a specific visit method, one can use the DOM API to traverse the tree in non-linear ways, overcoming a significant limitation of SAX.

Hopefully, you have already begun to see the ease of using the Visitor Pattern as it applies to the XML DOM. Using this Pattern the developer can define numerous DOM analyzers and processors and apply them to a DOM tree. All of the tree-walking code can be handled automatically through the adapter class.

For example, Listing 2 encodes games in a chess library.

Using a Visitor Pattern, one can intuitively visit the nodes in the XML tree and apply processing to selected elements (see Listing 3).

If other kinds of processing need to be done, then the developer must only declare a new visitor class (extending VisitorAdapter) and provide appropriate visitXXX() methods.

Summary

Extending the DOM API to support the Visitor Pattern means that XML application developers are freed from writing tree-traversal code and can focus their efforts on the processing of each XML element. Another benefit is that if several operations are required to use the same XML data, the development workload can be easily divided among several programmers. The processing workload can be split among several processors or threads as well.

Extending the Java/Xerces DOM implementation is straight-forward and we urge Sun to consider incorporating our changes into a future release of the API.

References

- Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. (1995) *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Professional.

ADAMS@CIS.GVSU.EDU

StrikeIron adds D&B Business Information to Premium Web Services

(Research Triangle Park, NC) – StrikeIron, Inc., a pioneer of online services and software to simplify working with Web services, has announced the availability of its StrikeIron D&B Business Information Premium Web Service. The offering is comprised of eight reports available through the StrikeIron Web Services Business Network (WSBizNet) and includes data such as D&B Ratings, Paydex scores, financial profiles, credit and payment information, credit scoring, risk assessment, company backgrounds and supplier evaluations.

With this new Premium Web service from StrikeIron, companies can increase productivity and optimize their decisions with access to real-time data. This is the first Web service to provide customers with a more effective on demand option to take advantage of the comprehensive research provided by D&B to improve business decisions and accelerate their business processes.

www.strikeiron.com, www.dnb.com.

Dralasoft Announces Dralasoft BPEL Orchestrator

(Westminster, CO) – Dralasoft, Inc., an innovator in Java technology for business process management (BPM), has released the Dralasoft BPEL Orchestrator, the first commercially available BPEL (Business Process Execution Language) workflow product with a mature, robust core engine and comprehensive management toolset. With Dralasoft's BPEL Orchestrator, users have a complete solution for optimizing service-oriented architecture (SOA) processes within, and between, enterprises.

Dralasoft BPEL Orchestrator leverages Dralasoft's existing and highly-regarded workflow technology. BPEL Orchestrator provides comprehensive BPEL design, execution (runtime), and monitoring support

Dralasoft BPEL Orchestrator is available now for evaluation purposes. Volume and OEM licensing options are available.

www.dralasoft.com

Digital Evolution Service Manager Supports Netegrity TransactionMinder 6.0

(Santa Monica, CA) – Digital Evolution, a provider of Web services security and management software, has announced its integration with the latest versions of market-leading identity and access management solutions from Netegrity and others.

The Digital Evolution Service Manager extends identity and access management platforms with comprehensive Web services management capabilities. Its integration with Netegrity and others enables the ability to provision access controls and performance levels of Web services according to the user and application accessing them.

www.digev.com

Reactivity Gateway Sets Speed Records for XML Web Services Security

(Belmont, CA) – Reactivity, Inc., a leader in secure XML Web services deployment systems, has released XOS 4.1, the latest version of its patent-pending XML operating system software. Running on the Reactivity Gateway 2400 Series platform, XOS 4.1 set new industry benchmarks for Web services security performance, delivering comprehensive enterprise scalability while accelerating time to market for secure XML Web services.

XOS 4.1 enables the Reactivity Gateway 2400 Series platform to achieve GigE throughput and eliminate potential bottlenecks in deploying secure XML Web services. Using XOS 4.1, the Reactivity Gateway 2400 Series appliance also set performance records for message parsing, cryptographic functions and schema validation, enabling enterprises to offload these power-robbing operations from their application servers and achieve significantly improved performance and security.

XOS 4.1 also includes several new XML schema management, SAML interoperability, and advanced SOAP header management features. Schema bundling allows several schema documents to be uploaded to the Reactivity Gateway and managed as a bundle. XOS 4.1 offers full support for all four interoperability scenarios given in the Web Services Security: SAML Token Profile, and supports SAML audience restrictions. Advanced SOAP Header Processing enables enterprises to require, reject, filter, and transform SOAP Headers in any namespace.

www.reactivity.com

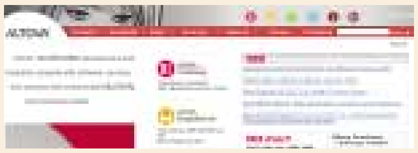
Companies Coauthor WS-Dynamic Discovery

(Fairfax, VA) – The Web Services Dynamic Discovery (WS-Discovery) specification. The specification, has been released by webMethods, BEA, Canon, Intel, and Microsoft, and is a major building block in creating flexible and adaptable Web services-based systems. It furthers the promise of Web services to reduce the cost and complexity of integrating systems between different platforms, operating systems, and programming languages.

WS-Discovery defines a protocol for locating Web services that fulfill a given set of requirements. It supports the use of multicast discovery in ad hoc environments, but also provides for the use of a discovery proxy or registry to allow scaling to a large number of endpoints.

Altova Announces General Availability of Software Version 2005

(Beverly, MA) – Altova, creator of XMLSpy and other software development tools, has announced general availability of version 2005 of its award-winning product line, which enables accelerated application development and data integration. Altova software version 2005 includes



robust new features such as automated function building, Eclipse integration, relational database content editing, and the revolutionary SchemaAgent that enables visual management of complex schemas and their components in workgroups. In addition, version 2005 brings substantially more power to XML related development with end-to-end support for the World Wide Web Consortium's new XSLT 2.0, XPath 2.0, and XQuery 1.0 specifications.Version 2005 covers Altova XMLSpy 2005, MapForce 2005, StyleVision 2005, and Authentic 2005.

The new products are immediately available for purchase via the Altova Online Shop at www.altova.com/order. Customers with a valid Altova Support and Maintenance Package (SMP) are eligible for a free update to version 2005 production software. A 30-day free trial may be downloaded from www.altova.com/download.html. www.altova.com

DECEMBER

BUZZ

SOAPtest®

Coding Standards and Unit Testing: A Natural and Necessary Progression

The software industry usually recognizes coding standard analysis and unit testing as two valuable — yet separate — error prevention practices. However, they are actually two necessary steps of achieving the same goal — ensuring that code is reliable and satisfies designated quality requirements.

No matter how extensively you unit test your code, your test results will not identify problems such as coding constructs that make the code error-prone during updates/maintenance or make it vulnerable to security attacks. Moreover, no matter how many coding standards you check, you won't be able to efficiently expose critical problems such as uncaught runtime exceptions, memory leaks, core dumps, and code that does not execute as you anticipated. You might identify some potential problems, but separating the false positives from the true problems would be too difficult and time-consuming if you were not also executing the code though unit testing or other dynamic testing.

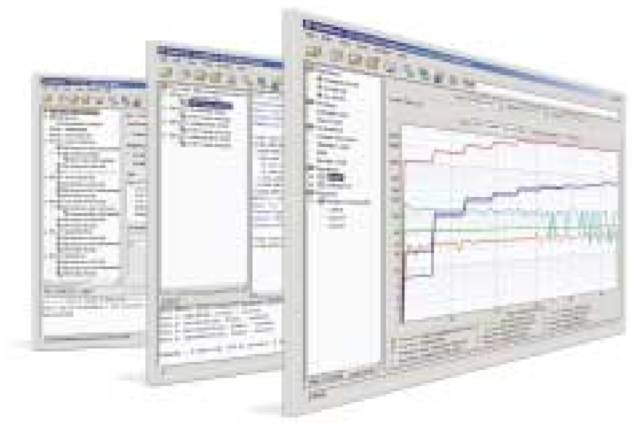
To identify the wide range of potential software risks, you need to read or parse source code to verify its compliance to "traditional coding standards," as well as dynamically exercise the compiled code during unit testing to verify compliance to "dynamic coding standards."

Traditional coding standards are typically technology-specific. For ideas of traditional language-specific coding standards to follow, refer to industry publications such as OASIS and W3C Specifications.

Some "universal" dynamic coding standards that you might want to follow are:
- Expose and address all exceptions, core dumps, and memory leaks
- Achieve 100% coverage of exception handling code
- Ensure that code does not treat valid inputs as exceptions
- Achieve 100% line coverage of all code
- Achieve 100% coverage of the specification

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

It's automated. It's fast. And it's the most versatile Web Services testing tool



Introducing Parasoft® SOAPtest®

The simple fact is, no other Web service testing tool can do what SOAPtest can do. Or do it as fast and reliably. From ensuring functionality and interoperability to telling you how your Web service is performing, SOAPtest automates all critical testing processes across the entire life-cycle of your Web service.

But don't take our word for it.... Go to www.Parasoft.com/SOAPtest and try it for free—no commitment, no obligation. If you like it (and we suspect you will) just let us know. We'll be more than happy to help you and your development team get up and running.

For Downloads go to www.parasoft.com/soaptest

Call 888-305-0041 x1209 or email: buzz@parasoft.com

PARASOFT®
We make software work.

Features

- WSDL schema verification and compliance to standards
- Automatic test creation using WSDL and HTTP Traffic
- Data-driven testing through data sources (Excel, CSV, Database Queries, etc)
- Scenario-based testing through XML Data Bank and Test Suite Logic
- Flexible scripting with Java, JavaScript, Python
- WS-I Conformance: Basic Profile 1.0
- WS-Security, SAML, Username Token, X.509, XML Encryption, and XML Signature support
- WS-Security Interop testing emulator
- MIME Attachment support
- Asynchronous Testing: JMS, Parlay (X), SCP, and WS-Addressing support
- Windows Perfmon, SNMP, and JMX monitors
- Detailed Report generation in HTML, XML and Text formats
- Real-Time graphs and charts

Benefits

- Uniform test suites can be rolled over from unit testing to functional testing to load testing
- Prevent errors, pinpoint weaknesses, and stress test long before deployment
- Ensure the reliability, quality, security and interoperability of your Web service
- Verify data integrity and server/client functionality
- Identify server capabilities under stress and load
- Accelerate time to market

Platforms

Windows 2000/XP
Linux
Solaris

Protocol Support

- HTTP 1.0
- HTTP 1.1 w/Keep-Alive Connection
- HTTPS
- TCP/IP
- JMS

Contact Info:

Parasoft Corporation
101 E. Huntington Dr., 2nd Flr.
Monrovia, CA 91016

www.parasoft.com

Middleware is Everywhere. Can you see it?

WebSphere **Key**

1. Guest checked in wirelessly.
2. Staff queries guest preferences.
3. Vendor services integrate seamlessly.
4. Supplies are procured automatically.
5. Repeat customers increase profits.

MIDDLEWARE IS IBM SOFTWARE. Powerful WebSphere software. It's the strong, seamless bond that can unite your business, vendors, partners and customers. A dynamic link designed to make your entire organization more efficient. More responsive. More flexible. On demand. WebSphere connects processes, with open standards. And it's easy to manage, too. So all involved get a better night's sleep.

Middleware for the on demand world. Learn more at **ibm.com**/middleware/process ON DEMAND BUSINESS™