SPECIAL FOCUS: WEB SERVICES STANDARDS!

July 2002 Volume 2 Issue 7

WebServices JOURNAL

.NET J2EE XML

WSJ2.COM

web services EDGE world tour 2002

COMING TO A CITY NEAR YOU

SEE PAGE 60 FOR DETAILS

2002

BOSTON ---------- JULY 10
SAN FRANCISCO -------- AUGUST 6
SEATTLE ---------- AUGUST 27
AUSTIN ---------- SEPTEMBER 10
LOS ANGELES ---------- SEPTEMBER 19
SAN JOSE ---------- OCTOBER 3

AND MANY MORE!

RETAILERS PLEASE DISPLAY UNTIL AUGUST 31, 2002
$6.99US $7.99CAN
07>
0 71486 03420 9

SYS-CON MEDIA

IBM

www.ibm.com/websphere/winning

WebServices JOURNAL
.NET J2EE XML

INTERNATIONAL ADVISORY BOARD
Jeremy Allaire, Andrew Astor, Steve Benfield, Philip DesAutels, Graham Glass, Tyler Jewell, Paul Lipton, Norbert Mikula, Frank Moss, George Paolini, Simon Phipps

TECHNICAL ADVISORY BOARD
JP Morgenthal, Andy Roberts, David Russell, Ajit Sagar, Simeon Simeonov, Richard Soley

EDITORIAL
EDITOR-IN-CHIEF
Sean Rhody sean@sys-con.com
EDITORIAL DIRECTOR
Jeremy Geelan jeremy@sys-con.com
INDUSTRY EDITOR
Norbert Mikula norbert@sys-con.com
PRODUCT REVIEW EDITOR
Joe Mitchko joe@sys-con.com
.NET EDITOR
Dave Rader davidr@fusiontech.com
EXECUTIVE EDITOR
Gail Schultz gail@sys-con.com
MANAGING EDITOR
Cheryl Van Sise cheryl@sys-con.com
EDITORS
M'lou Pinkham mlou@sys-con.com
Nancy Valentine nancy@sys-con.com
ASSOCIATE EDITORS
Jamie Matusow jamie@sys-con.com
Jean Cassidy jean@sys-con.com
ASSISTANT EDITOR
Jennifer Stilley jennifer@sys-con.com

PRODUCTION
VP, PRODUCTION & DESIGN
Jim Morgan jim@sys-con.com
LEAD DESIGNER
Richard Silverberg richard@sys-con.com
ART DIRECTOR
Alex Botero alex@sys-con.com
ASSOCIATE ART DIRECTORS
Cathryn Burak cathyb@sys-con.com
Louis Cuffari louis@sys-con.com
Aarathi Venkataraman aarathi@sys-con.com
ASSISTANT ART DIRECTOR
Tami Beatty tami @sys-con.com

CONTRIBUTORS TO THIS ISSUE
Jeremy Allaire, Jim Culbert, Don Ferguson, Paul Fremantle, John Fou, Greg Heidel, Murali Janakiraman, Heather Kreger, Anne Thomas Manes, Joe Mitchko, Klaus-Dieter Naujok, Santanu Paul, Sean Rhody, Nigel Thomas, Chris Weaver, Sanjiva Weerawarana

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

SYS-CON MEDIA

The End of the Beginning

Written by
Sean Rhody

Author Bio:
Sean Rhody is the editor-in-chief of ***Web Services Journal****. He is a respected industry expert and a consultant with a leading Internet service company.*
SEAN@SYS-CON.COM

I guess the title begs the question, if this is the end of the beginning, is it the beginning of the end? Hardly. But it is time to close the book on the first phase of Web services – the beginning of the hype curve.

Almost a year ago we decided that Web services would receive enough attention that we should consider devoting an entire magazine to the topic. We started with two teaser issues, one of which came out this time last year, and the response was dramatic enough that it justified our existence. In fact, ***WSJ*** is one of the fastest-growing titles in the history of **SYS-CON Media.**

This really has nothing to do with the end of the beginning, but it sets the stage well. A year ago we were all discussing possibilities. Would .NET catch on? Would Java support Web services? We even wondered exactly what Web services was and who was going to use it.

Some of the initial thoughts were grandiose and off base, but in general we got it right. Many of the initial visionaries and evangelists of Web services pictured a world where people could look up any service they wanted and then simply invoke it – on the Web, across the Web. This vision was coupled with the ultimate scenario, one in which companies exposed their business processes to one another via a global directory service. While the possibility still exists, we've seen that in the near term, companies are largely content to use Web services to integrate their own internal processes first.

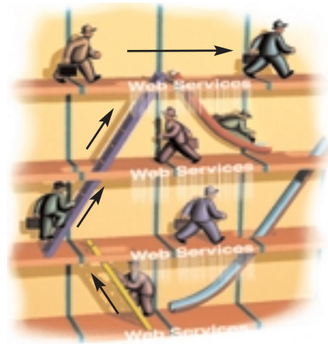
What's become apparent in the year since we started exploring this topic is that Web services has achieved critical mass. It has enough horsepower to be a viable player in the market.

How can we judge this? Largely by the fundamental shift in the thought processes of vendors and implementers, from the realm of trial applications and rough-hewn support fit only for hardcore developers to the domain of working implementations and smooth packages aimed at high-level developers and business users.

Adding WSDL, UDDI, and SOAP support was relatively easy for the J2EE vendors, and Microsoft built it on top of their platform. Even a large number of vendors who aren't in the application server business provide a Web services stack. At this point it's not a question of where to get a UDDI provider, but which UDDI provider to use.

There are already choices for Web services deployments – multiple choices, with increasingly large numbers of referenceable implementations. What really starts to point to the next step in the evolution of Web services is the rise of companies focusing on two entirely different aspects of provisioning: ease of use and management of service.

On the one hand we have a variety of vendors building workbenches, IDEs, suites – all designed to make it easier to develop and deploy a Web service without ever having to actually write WSDL. These are the tools that take Web services from the esoteric realm of the hardcore developer into the realm of corporate America. These are the tools we really need for Web services to become endemic.



On the other hand, there's management. Web services, like any other corporate asset, needs to be watched, managed, and measured. Service-level agreements are meaningless without tools to measure the ability of a service to comply with them. Much like application servers, Web services won't fully penetrate the corporate infrastructure until it can be managed by a corporate management application such as Tivoli, Unicenter, or OpenView. As in the tools domain, vendors are debuting products that make Web services an integral player within the management space.

So it's the end of the beginning. But not to worry, it's not the beginning of the end.

DESIGNING CONTENT Management Systems Using Web Services

Written by Santanu Paul

AUTHOR BIO:

Santanu Paul is the chief technology officer at Openpages, a leading provider of enterprise content production systems. With more than 10 years of industry and R&D experience, Paul is an expert on designing and developing state-of-art content management and workflow systems.

Historically, content management systems (CMSs) have been notorious for falling short of enterprise expectations. This is because, despite claims to the contrary, most first-generation CMSs were essentially packaged C or C++ applications originally conceived and designed to solve specific problems, such as publishing departmental Web sites out-of-the-box.

As content management problems proliferated through the enterprise and corporate CIOs looked to deploy their CMSs on a larger scale, these first-generation systems came under increasing pressure to morph from simplistic departmental applications into scalable enterprise infrastructures for content management. However, given their proprietary architectures, first-generation CMSs were poor candidates for enterprise-wide deployment. Weak application program interfaces (APIs) made them incapable of responding to emerging content production needs other than the specific scenarios contemplated by the vendor. Furthermore, these systems had a poor track record in terms of enterprise integration; they were expensive to own and difficult to scale in a cost-effective manner.

Second-generation CMSs now on the market are better suited to the challenge of enterprise-wide deployment. On one level, these systems provide application functionality comparable to that of their predecessors – they support standard Web site publishing and document management functions out-of-the-box. However, these systems are radically different in terms of architecture.

Based on open standards such as J2EE, second-generation content management systems have modular designs and expose core content management functions as rich APIs. In other words, the key characteristic of a second-generation CMS is that it's not merely an application, it's a software platform that can be programmed to support emerging business-critical content management applications not contemplated by the system vendor. Furthermore, second-generation systems provide a rich set of service provider interfaces (SPIs) that allow businesses to easily integrate with enterprise IT infrastructure. The net effects are greater flexibility, extensibility, and ease of integration, which reduce the total cost of ownership.

While second-generation systems represent a significant improvement over first-generation systems, much remains to be done to make content management functionality readily accessible to business users. Consider a content production scenario where Jenny Meyers, vice president of marketing at an online brokerage company, decides to launch a weekly e-mail newsletter called *Investor Update* that will provide affluent customers with personalized market commentary based on their portfolio composition. The objective of the newsletter is to leverage the company's famed research group. Setting up a closed, first-generation system to support this new process is virtually impossible. Second-generation systems are programmable; within a few weeks, a team of programmers can develop and test a specialized application to support *Investor Update*.

But what if, for competitive reasons, the business goal is to launch *Investor Update* within seven days and there are no programmers to help develop a specialized application? Is a third-generation content management system – one that empowers Jenny to configure the system to suit her content production needs – possible? To generalize the problem, what kind of content management system do business users need to engage in self-service content production? Is self-service content production only applicable within an enterprise? How about content extranets where enterprises can invite their business partners and customers to create custom-content products in self-service mode?

Web services hold the key to self-service content management systems and to the

The evolution of content management technology

rapid development and deployment of applications such as *Investor Update* by business users like Jenny.

Self-Service Content Management

A self-service content management system is best designed as a collection of loosely coupled, best-of-breed software components, where each component is an autonomous Web service that performs a specialized content management function and is implemented using a Web services framework such as .NET or J2EE. This article uses J2EE to illustrate how Web services may be implemented.

In the loosely coupled model every specialized component can be a potential provider and/or requester of services. Since the software components can have heterogeneous implementations, they must be able to communicate with each other via platform- and data-neutral protocols. Equally important, business users like Jenny must be able to configure their content production processes by assembling these Web services in the correct order via highly intuitive, wizard-like user interfaces.

Elements of Content Management Web Services

A self-service content management system consists of a core set of building-block services, including the library service, workflow service, transformation service, import and export services, publishing service, and categorization service. As shown in Figure 1, these services must coexist within a Web services ecosystem. Specifically, their WSDL interfaces must be registered with an enterprise-wide UDDI registry. This registry can serve as the "yellow pages" for services and applications that need to interact with these services.

Within a J2EE application server these services can be implemented as stateless session beans. These session beans can be exposed via JAX-RPC as Web services that support synchronous SOAP/XML-RPC interfaces. Figure 2 shows the underlying J2EE architecture of content management Web services.

FIGURE 1 | **Web services ecosystem**

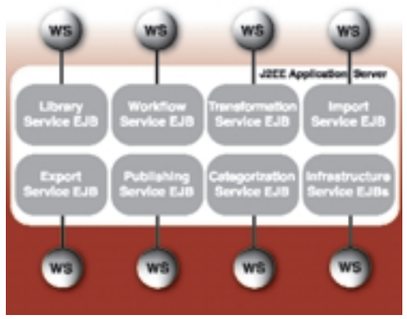


FIGURE 2 | **CMS Web services on J2EE**

Library Service

The library service is at the heart of a self-service content management system. It's responsible for managing content – both as native files and as XML – and metadata properties. Every content item must have a content type that describes its semantics. Press releases, FAQs, and product datasheets are examples of content types with specific properties, file formats, and creation templates. For example, a press release can have a property "Release Date," and the author of a press release can create it using a specialized MS Word template or HTML form. In each of these cases XML can serve as the intermediate content representation.

The library service offers services related to content types and their properties, content check-in and checkout, versions and renditions, folder hierarchies, access control, and component-level XML manipulation. In terms of implementation, the library service can use a relational database to store content properties and file servers to store content files. A powerful storage framework can allow multiple, disparate databases and storage servers to be integrated with the library service via service provider interfaces. This is advantageous for scaling the repository as well as interfacing with legacy content stores.

The library service supports at least two kinds of searching capabilities. It provides parametric search on content properties and indexed search on the content itself. Parametric search can be based on SQL and leverage the fact that content properties are maintained in a relational database. Indexed search can be based on conventional search engine technology that maintains an index of XML renditions of all text content in the repository.

Workflow Service

The workflow service is responsible for running active processes and associated tasks that relate to content creation, production, and publishing. It can be used to support simple edit-review-approve workflows as well as complex, collaborative projects. The workflow service supports routing constructs such as task nodes, conditional nodes, parallel split nodes, merge nodes, and independent subprocesses. Workflow instances should be activated from workflow definitions that describe commonly used business processes. Users should be able to create workflow definitions graphically using a visual builder application.

The workflow service is optimized to support collaborative workflows common to content management scenarios, as well as to

provide a high degree of flexibility to workflow owners. For example, the workflow service allows users with appropriate permission to modify active workflows midstream, as well as add new tasks and update routing information to reflect a project's changing needs. Such flexibility is critical in content production projects where the precise workflow may not be fully known at the beginning of the project.

The workflow service also assigns tasks and roles to individuals via task lists. It should be possible to automate tasks; they may be performed by scripts or by invocation of external applications. Timers allow overdue tasks to be escalated and notifications to be delivered to appropriate individuals via a separate notification service. Extensive audit trails can be maintained for monitoring and analysis.

Transformation Service

The transformation service specializes in format conversions. As an autonomous service, it allows specific transformation algorithms to be plugged in as transformer objects. It also provides conversion to XML from commonly used text formats, including MS Word, PDF, Excel, and PowerPoint. In addition, the transformation service converts between image formats such as GIF, JPEG, BMP, TIFF, EPS, and SVG. The engine can be invoked at any time with a specific transformation request on a particular content file. On receiving such a request, the engine utilizes the appropriate transformer to perform the request and return the transformed content to the requester. The transformation service is usually invoked by the import, export, and publishing services and by applications to perform file conversions on demand.

Import and Export Services

The import service is responsible for importing content from external systems. It needs to contain a polling engine and specialized importer objects that can periodically monitor files on network drives, FTP sites, and HTTP sites. When new files are detected, they are imported and processed according to the business logic specified inside the importer objects and inserted as new content into the repository via the library service. Typically, the importer objects use the transformation service to convert incoming text files to their equivalent XML renditions. Imported objects are configured using XML descriptors. Other Web services or the J2EE Connector API can be used to develop custom importers for importing content from proprietary enterprise information servers such as ERP systems or legacy databases.

The export service pushes content out to external systems. In addition to acting as a file transfer service, it delivers periodic, incremental updates to destinations by transferring only changed content. In the case of updating a group of destinations that mirror each other, the service can ensure that the updates are part of an atomic transaction. The service may also provide a rollback facility.

Publishing Service

The publishing service is a collection of individual services responsible for repurposing content to a variety of channels. A channel is a logical representation of a publishable entity such as a Web site, print publication, syndication feed, CD-ROM, or wireless broadcast. A channel publisher is usually a Web producer or print editor. Once original content has been created and reviewed, the producer can use it to publish new pages by assigning it to specific presentation templates. Examples of presentation templates include XSL stylesheets, HTML pages with embedded tags, JavaServer Pages, and QuarkXPress page layouts.

An important principle is single-sourcing of content across multiple channels. Multiple Web sites can refer to the same image file, and changes to the file are reflected within every channel either immediately or as soon as the channel is republished. Assigning content to a channel is an important event; it should trigger a set of rules that cause channel-specific transformations to occur via invocations of the transformation service. For example, a rule can indicate that every TIFF image assigned to a Web channel must be transformed into a JPEG image.

Categorization Service

The categorization service classifies content into meaningful categories. For example, a mutual fund prospectus may be classified as belonging to both Equity Fund and Foreign Fund categories. Content may be classified by human experts or by auto-categorization tools, or a combination of the two. A central facet of the categorization service is a taxonomy hierarchy that describes content categories and subcategories available for classification. Industry-specific taxonomy hierarchies are becoming available from industry consortia; these can serve as a basis for content categorization.

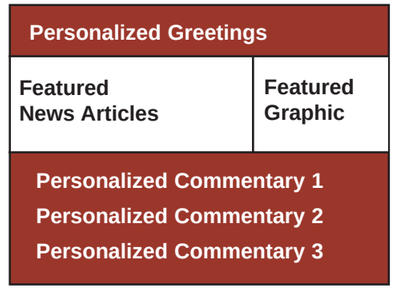


FIGURE 3 | ***Investor Update*** **template**

Infrastructure Web Services

To function effectively, content management Web services need to interact with infrastructure Web services that serve content management Web services as well as other Web services in the enterprise. Two such infrastructure Web services are the user management and authentication service and the event notification service. These are Web services that act as shared enterprise resources.

User Management and Authentication Service

This Web service handles users and their groups as well as authentication. Users and groups can be created, updated, and deleted and user entitlements can be managed via this service. For example, user Joe Smith may be assigned a role called "Author," which has the right to create content and perform collaborative tasks but doesn't have the right to publish channels. To implement authentication and user management, the service can use a delegation model. For simple usage, the service can provide mechanisms to batch import user account and rights information from external directory and policy servers. For advanced needs, the service supports integration with security realms such as LDAP, Windows NT domains, or proprietary security realms.

Event Notification Service

This Web service is responsible for handling notification events generated by services and delivering them to their intended recipients. This is particularly useful for multicasting actions and events such as content submission, content assignment, and channel publication. The intended recipients of these events may be other services, end users, or system administrators. In the J2EE world, the event notification service can be implemented using the Java Messaging Service (JMS). When notifications must be delivered to human recipients, the JavaMail API can deliver e-mails via an SMTP server. For other forms of delivery, the e-mail delivery system can be hooked to pagers or wireless gateways to deliver notifications to handheld devices and personal digital assistants.

FIGURE 4 | ***Investor Update*** **application workflow**

Real-World Application

How do content management Web services, if implemented as described above, benefit Jenny Myers, vice president of marketing at an online brokerage company? How do they help her solve the *Investor Update* publication problem?

In the new world, Jenny launches a wizard that allows her to set up and configure her *Investor Update* publication process. The wizard guides her through a series of high-level choices and end-user actions pertinent to the publication of *Investor Update*. Once she's completed her choices and actions, a script is automatically generated that ties together invocations to the Web services required to support the *Investor Update* publication process. This script is then saved as "Jenny's *Investor Update* application." Jenny can invoke it at any time, assign it new content, and automatically generate personalized *Investor Updates*.

To create the application, the wizard guides Jenny through the following steps. First, she is asked to enter information about the project, such as goals, budget, human resource allocations, timelines, and other project information. Next Jenny has to indicate the channel format – in this case, rich e-mail. Based on her selection, the wizard presents a set of HTML e-mail templates with different layouts and allows Jenny to make a selection. She selects one (she can also initiate a request for a new template to be created based on her specifications). Once she's identified her template of choice, the wizard gives her the flexibility to make simple adjustments to the look and feel of the template. Figure 3 shows an example of a template Jenny can select. She also has the ability to associate content containers within the template with rules that can dynamically extract content from the repository. As the last step, the wizard guides her to select a production workflow to produce *Investor Update*. Once Jenny's done, she saves her application. Under the covers, the wizard generates a script that stitches together the appropriate invocations to relevant Web services. To verify that the application is indeed working correctly, Jenny can now test the *Investor Update* application with sample content.

To generate e-mail newsletters each week, Jenny simply launches her *Investor Update* application. The application displays a list of recipients, and she selects the ones to whom she wants to send the newsletter. Next, Jenny is asked to assign a feature story for the standard feature story spot in *Investor Update*. She uses the library service to find a feature article. After collecting all the relevant information, the application generates a set of e-mail newsletters, one for each intended recipient. Each *Investor Update* has a personalized greeting at the top, a standard feature article and graphic in the middle, and three top news items personalized per recipient based on their portfolios. Jenny can inspect these e-mail newsletters and, if satisfied, direct that they be sent. Figure 4 shows the workflow orchestrated by Jenny's *Investor Update* application as the newsletters are produced. It involves multiple interactions with the user as well as various Web services.

Conclusion

Web services hold the key to self-service content management systems. These systems will enable business users to take control of their content production needs and engage in the rapid development and deployment of applications, without depending on software developers. A self-service content management system is best designed as a cooperating collection of loosely coupled software components. In this model each software component is an autonomous Web service that performs a specialized content management function.

Even though Web services frameworks make it increasingly feasible to develop such content management systems, it is likely that first-generation content management systems conceived as packaged applications will not easily make the transition to self-service content management. Second-generation systems based on open platforms such as J2EE, or that are modular enough to take advantage of .NET, are better positioned to take advantage of the Web services revolution. Among them are the harbingers of self-service content management.

Over the past year or so Web services has developed into the latest and greatest development craze. The Web services concept provides a strong impetus for current development of both of the major competing enterprise platforms – Microsoft's .NET and Sun Microsystem's J2EE. In the Java world the Web services initiative is one of the main focus points for ongoing J2EE 1.4 development.

Is Web services just a passing fad, or does it really represent a new paradigm in application development?

What Is Web Services?

Essentially, Web services is the long-distance glue that allows services (application components) to be pulled together across IP networks. The analysts at D.H. Brown Associates, Inc., have described Web services as "XML-enabled standards to allow publishing of applications to other parties across platforms."

Web services is based on a service-oriented architecture. Imagine a simple Web service: managing credit card transactions for small online businesses. When you make a transaction on the Web site, you enter the credit card number and the shopping basket displays the total amount to be submitted for authorization. You also provide a name and billing address. The authorization transaction verifies that your card is legitimate and that your name and address match the bank's records. The credit card processor authorizes or rejects the transaction, and the Web site operator can then accept or cancel the order.

Credit card processors have offered these services before, but now the interface can be defined using WSDL (Web Services Description Language), and the service can be advertised in a UDDI (Universal Description, Discovery, and Integration) registry. In principle at least, the Web site operator can build around a standard Web services–based credit card transaction, then "plug it in" to any suitable card processor. The service is executed by sending an XML message using SOAP (Simple Object Access Protocol) – essentially an XML synchronous remote procedure call.

As long as the WSDL interface definitions are identical, the Web site can easily switch card processors by searching the UDDI registry for alternative providers of the same Web service.

Who's Using Web Services?

A number of organizations have already begun to, or are about to, deploy Web services:

- The UK government is preparing to embrace Web services standards in order to speed up the massive task of creating an Internet-based infrastructure for all government services by 2005.
- Financial services institutions: Always early adopters of new technology, they are beginning to contemplate Web services–based fulfillment of straight-through processing requirements. However, these organizations need to be convinced that Web services can provide sufficient reliability and security before they expose critical financial systems to the technology. Expect instead that simple opportunities (like our example) will develop first.
- Online retailer Nordstrom .com is using Web services technology to help integrate its ERP system with mainframes at parent Nordstrom, Inc., and is extending these services to other suppliers and partners. They have also been using Web services to track gift card redemptions from off-line purchase through online redemption to Nordstrom's own banking subsidiary.

Author Bio

Nigel Thomas is the director of product marketing at SpiritSoft. Prior to SpiritSoft, Nigel spent five years with EAI vendor Constellar, serving in consulting, support, sales support, and development positions. He also spent more than eight years at Oracle Corporation architecting and delivering Oracle's accounting products and then moving to worldwide performance consulting and CASE development assignments. For more information on SpiritSoft, visit www.spirit-soft.com.

NIGEL.THOMAS@SPIRIT-SOFT.COM

- Software vendors such as Sun Microsystems, Oracle, and others are including Web services protocols in their platform and development tools products. Microsoft, perhaps more than any other vendor, is planning to offer public Web services – notably .NET Passport, which is an online service that makes it possible for you to use your e-mail address and a single password to sign in securely to any participating Web site or service. This will eventually have a strong impact, particularly on consumer-focused applications.

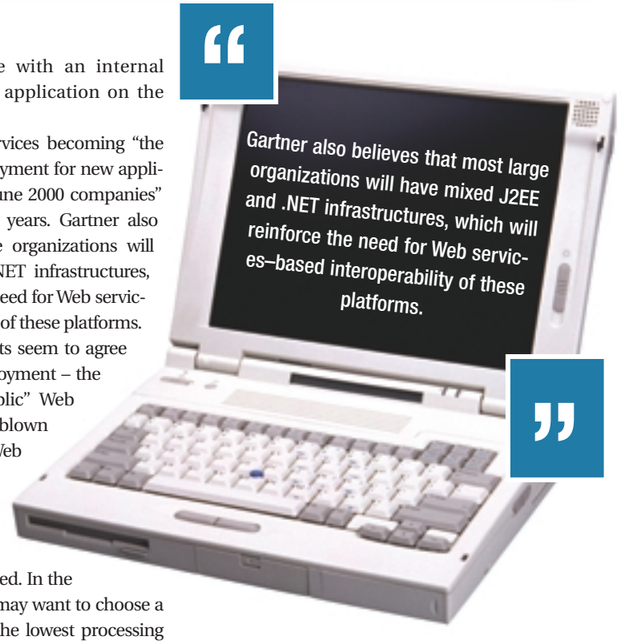
Three Waves of Web Services Deployment

Industry analyst Forrester Research, Inc., expects three waves of deployment. The first wave will be "partner" Web services to help cement complex supply chains. This initial deployment phase, based on Web services–friendly standards like RosettaNet and ebXML, already has some support in cooperative transaction-oriented industries such as electronics. Our earlier example of a credit card authorization service is a very simple case of this, in which the bank is the Web site owner's "partner."

The next wave will be what Forrester calls "private" Web services. These will essentially use Web services protocols to extend internal enterprise application integration efforts. This will appeal especially to organizations that have complex internal structures, many packaged and custom applications, a wide geographical spread, and a high level of local autonomy for their operational and IT departments.

An informal Giga Group survey published in December 2001 stated that 60% of the respondents (120 IT executives attending Giga's "Emerging Technology Scene" conference in December 2001) saw Web services as a strategic component of their internal integration strategy, rather than specifically for interbusiness use. This approach is seen as more immediately useful; after all, the business can be sure that the service will be used and will lower risk. In a large retailer, for example, a bank-provided Web service for credit card authorization might interface with an internal credit card–processing application on the retailer's side.

Gartner sees Web services becoming "the dominant mode of deployment for new application solutions for Fortune 2000 companies" over the next couple of years. Gartner also believes that most large organizations will have mixed J2EE and .NET infrastructures, which will reinforce the need for Web services–based interoperability of these platforms.



Finally, all the analysts seem to agree that a third wave of deployment – the widespread use of "public" Web services and the full-blown adoption of the entire Web services stack, including registries and the automated discovery of services – will be delayed until standards are finalized. In the case of our example, we may want to choose a replacement bank with the lowest processing charges. We can do this by searching a UDDI registry to find an alternate service with the same WSDL specification, but only if the registrations include pricing, and only if the service interface is standardized. Otherwise, we end up having to recode our own application.

A Slow, But Steady, Adoption Curve

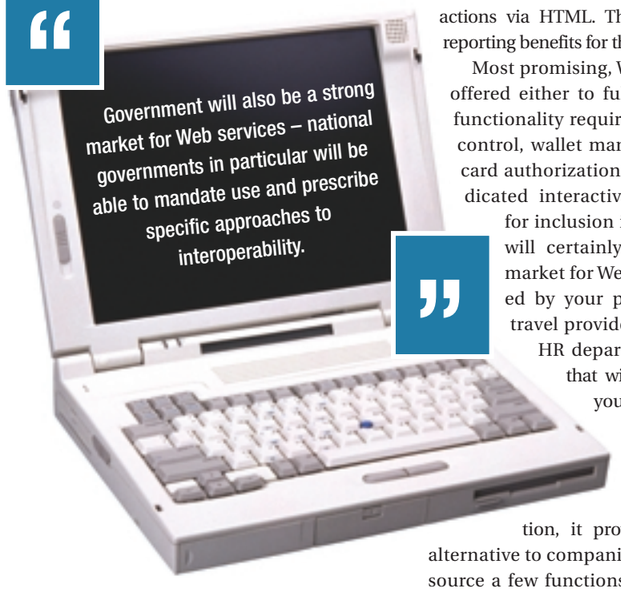
The low-level components of Web services – the plumbing, if you like – will make a useful contribution to interplatform interoperability as a lowest common denominator *lingua franca*. This will help to isolate business relationships (within and between companies) from technology decisions by providing a vendor- and architecture-neutral point of integration.

In particular, we can expect to see a Web services "skin" being used to cover platform-specific components such as J2EE Connector Architecture–based adaptors. This will meet the needs of internal application construction and integration as well as B2B projects.

Behind the Web services facade, we'll still see the same old applications, and developers will still have to deal with the age-old problems of application construction, application integration, and component reuse. Applications (whether designed to offer a Web service or constructed by aggregating a number of Web services) still need to offer the appropriate levels of reliability, availability, performance, scalability, and security. Under the counter, what may look to the outside world like a single Web service will actually require complex integration with existing enterprise applications. Developers will see the Web services interface as yet another feature to add to their EAI landscape.

Gradually, we can expect to see packaged applications delivered with predefined Web services interfaces. This will replace existing proprietary interfaces, such as SAP's RFC (Remote Function Call), or add standards-based functional interfaces where none existed before. EAI vendors will become more tightly focused on business process management, data mapping, and transformation – based on a general purpose J2EE- or .NET-based application server platform and Web services standards – rather than having to invest in different physical connectors for each application vendor.

At SpiritSoft, we don't really think large corporations are going to fall for the "discovery"



aspect of Web services anytime soon. Would you want to conduct business with partners selected from the Web services equivalent of a Google search, without any human checking of the partner's reliability (or even existence)? Would you trust your entire business to an anonymous search engine? That seems to be one of the more extravagant premises on which UDDI is being promoted.

It seems more likely that industry communities, all with their own memberships, will develop rules and regulations. This may sound a lot like the trading hub model, which was somewhat deflated last year. In fact, the communities are more likely to gather under the wing of a dominant member (suppliers to a food retailer or car manufacturer, for example) or be policed by an existing industry body or consortium (RosettaNet, ebXML, Microsoft's Passport) that can add value and trust to the basic Web services framework.

Government will also be a strong market for Web services – national governments in particular will be able to mandate use and prescribe specific approaches to interoperability. Web services represent a more convenient interface for G2B (government-to-business) integration than either proprietary tools or manual interactions via HTML. Think of the possible reporting benefits for the IRS!

Most promising, Web services will be offered either to fulfill common point functionality requirements (like access control, wallet management, or credit card authorization) or to provide syndicated interactive content suitable for inclusion in a Web site. There will certainly be a competitive market for Web services – provided by your parcel service, your travel provider, your self-service HR department, and so on – that will be integrated into your corporate portal. Because Web services offers a very granular approach to application integration, it provides an attractive alternative to companies that want to outsource a few functions, but not an entire system. No CIO wants to be forced to outsource everything, and with Web services, he or she can control just how much is outsourced (or "insourced" to autonomous departments), and in what size application "chunks." Meanwhile, subsidiaries and departments will be able to mix and match their own Web service providers.

Conclusion

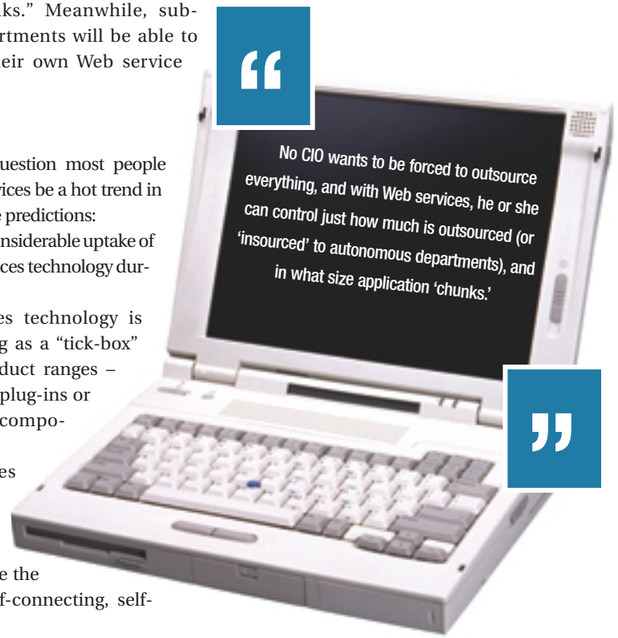
To answer the question most people have – "Will Web services be a hot trend in 2002?" – I make these predictions:

- Yes, there will be considerable uptake of the basic Web services technology during 2002.
- Yes, Web services technology is already appearing as a "tick-box" extension to product ranges – either as partner plug-ins or as an intrinsic component.
- No, Web services won't slay everyone's development dragons.
- And no, I can't see the realization of self-connecting, self-organizing applications based on runtime discovery of other people's Web services, at least not during 2002.

Forward-looking developers will welcome the ability to expose internal application components on the Internet or intranet. They will look for infrastructure vendors who provide open, standards-based technology components that can easily be slotted together so users are not locked into vendor-specific, proprietary frameworks. This will help to limit the cost of adopting the technology and will ensure that your architecture can be extended to meet new challenges as they arise.

Related Links

- *Web Services and the Move to Loosely Coupled Computing*: http://e-serv.ebizq.net/wbs/bals_1.html.
- *It's a Web Services World*: http://e-serv.ebizq.net/wbs/otoole_1.html.
- *Web Services: Enabling the Collaborative Enterprise*: http://e-serv.ebizq.net/wbs/donato_1a.html.
- *Web Services: The Next Evolution of Application Integration*: http://e-serv.ebizq.net/wbs/wong_1.html.



SOAP & SECURITY

It doesn't implement security, but it helps

Anne Thomas Manes

Anne Thomas Manes is CTO of Systinet, a Web services infrastructure company. Anne is a recognized industry spokesperson and has published on a range of technology issues.

ATM@SYSTINET.COM

Based on the number of questions I get on the subject, quite a few people think that SOAP isn't secure. It's a bit hard to answer these questions because SOAP is neither secure nor insecure. It's not within the scope of SOAP to implement security. SOAP is simply a mechanism to package information to send between two applications. Even so, it's easy to secure SOAP messages, and SOAP provides an extensible mechanism that allows you to convey security information in your messages.

Security is a complicated topic, so let me start by explaining the basic goals of security when dealing with distributed computing.

- Message integrity ensures that it isn't modified in transit.
- Message confidentiality ensures that the message can only be read by the intended recipient.
- Proof of origin provides proof to the receiver that the message indeed came from the sender.
- Mutual authentication allows the client to verify the identity of the service and the service to verify the identity of the client.
- Authorization controls access to the service.

The most common mechanism used to implement message integrity, message confidentiality, and mutual authentication is a transport-level security system, such as the Secure Sockets Layer (SSL) or Transport Layer Security (TLS). SSL/TLS uses public key encryption to protect messages between two points. If the sender encrypts the message using its private key (as opposed to the receiver's public key), SSL/TLS also supplies proof of origin. All authentication and encryption actions occur at the transport layer, so SSL/TLS security is completely transparent to the communicating SOAP applications.

For simple communications, SSL/TLS is often sufficient, but as things get more complicated, additional security measures are needed. SSL/TLS only protects messages as they are transferred between two network ports. In many cases, a message may need to be routed through one or more intermediaries (such as a firewall or an auditing service) before reaching its final destination. So sometimes you need to use application-level security. Application-level security gives you end-to-end security control. It allows you to establish a separate identity for each service running on a server. It allows you to delegate or propagate security information across multiple hops. And it allows the service to implement authorization controls.

When using application-level security, you need to pass security information, such as user IDs, permissions, and security tokens (X.509 certificates or Kerberos tickets) within the message. A SOAP message normally passes this information in a header element. As the message travels through the routing path to its final destination, each intermediary can prepend additional security information to the security header element to indicate its progress through the path. You can still use SSL/TLS to encrypt these messages, but in some cases you might want to encrypt or digitally sign only certain parts of the message. (You might want to make only certain information available to each intermediary.) In this case, you need to use an application-level encryption service rather than SSL/TLS.

The W3C XML security standards can be used for this. XML Signature provides a mechanism to digitally sign all or part of a message. It relies on Canonical XML to normalize the XML message before encryption, and XML Encryption provides the encryption process. Signature information then needs to be specified in the security header element.

The OASIS Security Assertions Markup Language (SAML) is another handy standard. SAML is designed to support single sign-on operations. It provides a standard format to exchange security information, including authentication assertions, qualifying attribute information, and authorization decisions.

For example, a SAML assertion might specify that my corporate LDAP directory service asserts that I am Anne Thomas Manes, employee of Systinet Corp. This assertion is based on a password challenge that occurred at 11:20:22 on 04/16/02, and I am permitted to submit a purchase order to Acme Supplies for an order not to exceed $5,000; this assertion is good for 30 minutes. Once I obtain this assertion, I can plug it into a header element in a SOAP message containing my purchase order for Acme Supplies. Assuming that Acme Supplies trusts the Systinet LDAP directory service (we have a pre-existing trust agreement), it should allow me to make the purchase without requiring me to sign on to the Acme system directly.

As you can see, the basic technology is in place to support end-to-end Web services security. The only issue still at large is one of interoperability. If two applications are going to exchange security information, they must first agree on how to represent the information within SOAP messages. SAML provides a standard way to represent the security assertions, but we still need to define a standard format for expressing digital signatures and partial encryptions.

IBM, Microsoft, and Verisign recently published a WS-Security specification describing a set of SOAP conventions that can be used to exchange security information and to digitally sign messages. This specification focuses on direct authentication rather than single sign-on and doesn't include support for SAML assertions, but it does address the issues of digital signatures and partial encryptions. Unfortunately the copyright notice at the beginning of the specification makes it clear that it is not available to the general public.

I'd like to see the W3C set up a new working group to define a standard royalty-free Web services security specification. The sooner the better.

written by Greg Heidel

Web Services Standards Update

Ready for the foreseeable future

Web services has the potential to solve some of the most difficult technology and integration problems that have plagued IT departments for decades. Isolated systems, redundant code, extended development cycles, and vendor dependence have essentially been accepted as inherent side effects of enterprise computing. If Web services is to alleviate these problems, a complete, broadly accepted set of standards must be realized.

Greg Heidel is an independent consultant in Web services. His experience includes architecting and designing systems on both the J2EE and .NET platforms and researching the future directions of Web service technologies.
GREG@DISTRIPUTE.COM

In an earlier article (*WSJ*, Vol. 2, issue 1), I provided a broad look at the Web services standards landscape. At the time, XML and SOAP had reached fairly widespread acceptance and there was great optimism about the flurry of activity in other critical areas, such as service description and discovery. In this article we'll look at what changes have taken place in the major standards organizations. Then we'll take a peek at how far existing standards have progressed and what new standards have emerged over the past six months.

Standards Organizations

One of the keys to the success of developing and promoting acceptance of a standard is the organization defining the standard. Standards organizations have a huge responsibility to not only create and foster standards but also to manage their processes in order to quickly and efficiently release these standards. In addition, they have a broader responsibility to work together to deliver a unified, complementary set of standards rather than introduce specifications that create standards fragmentation and jeopardize interoperability. The work of these organizations will have a broad impact on vendor compliance and customer acceptance. Let's take a look at the leading standards organizations that define the Web services landscape today.

The World Wide Web Consortium (W3C)

The W3C is the premier Web services standards organization. Created in October 1994, the W3C is working to develop a set of technologies in order to bring Web services to its full potential. A new development within the W3C is the creation of the Web Services Activity, which consists primarily of three working groups focused on Web services:

The Web Services Architecture Group

This working group is responsible for identifying, designing, and documenting a coherent architecture for Web services. This group is scheduled to have a working draft architecture document available by June of this year.

The XML Protocol Working Group

This working group is chartered with creating a layered system of protocols (primarily XML and SOAP). The goal of these protocols is to meet the needs of applications with simple interfaces and be extensible in order to provide the security, scalability, and

robustness required for more complex application interfaces. Their public schedule called for their work to be done as of April of this year, which has come and gone.

The Web Services Description Working Group

This working group is tasked with defining a standardized way to define Web services interfaces. It will review the scope of the WSDL 1.1 specification as part of the interface component design task. In fact, the group is chartered with making only agreed-upon improvements to the WSDL 1.1 specification, rather than arbitrary changes. They are scheduled to have a draft document in June of this year.

This group is a welcome and necessary addition. The W3C has been under criticism for the lack of movement of the WSDL specification and specifically for not having a working group assigned to this area.

Organization for the Advancement of Structured Information Systems

OASIS is a nonprofit consortium founded in 1993 and dedicated to the promotion of open specifications for the interchange of structured data. OASIS is driving several key Web services standards in the areas of security, transactions, and interactive Web services, in addition to sponsoring the ebXML specifications.

Interactive Web services is a relatively new area driven, at least in part, by the portal industry. Interactive Web services typically involve a person interacting with a Web service in some capacity. OASIS has two working groups actively working in this area, WSIA and WSRP.

Web Services for Interactive Applications

WSIA is chartered with creating an XML- and Web services–centric component model for interactive Web applications. This group is driven, at least in part, by the predecessor specification WSXL (Web Services Experience Language) and earlier work by Epicentric and divine. The two main goals of WSIA are to:

- Enable businesses to distribute Web applications through multiple revenue channels
- Enable new services or applications to be created by leveraging existing applications across the Web.

Web Services for Remote Portals

WSRP is defining an XML and Web services standard that will allow the plug-and-play of visual, user-facing Web services with portals or other intermediary Web applications. It is focused on improving content delivery via portlets by using a common set of APIs.

Both of these groups are early in their work and neither had generated a draft specification as of the writing of this article.

> "One of the keys to the success of developing and promoting acceptance of a standard is the organization defining the standard"

UDDI.org

A group of companies identified on the www.uddi.org Web site is developing a set of open specifications for a service registry. Their goal is to create a platform-independent, open framework for describing services, discovering businesses, and integrating business services using the Internet. At some point, hopefully in the near future, this work will most likely be incorporated into the work of the W3C.

BPMI.org

The Business Process Management Initiative (BPMI) is a group formed to define a standard way to model business processes. Its goal is to promote and develop the use of Business Process Management (BPM) through the establishment of standards for process design, deployment, execution, maintenance, and optimization.

Currently this area is fairly fluid, with Microsoft, BEA, Sun, and OASIS all active in this space. IBM, which authored the Web Services Flow Language (WSFL) specification, is also a member of BPMI.org.

Web Services Interoperability Organization

WS-I is a relatively new organization committed to promoting interoperability among Web services. The group was formed in early February in an effort to create testing tools and standard documentation to enable competing vendors to ensure compatibility between Web services regardless of vendor or implementation. The documentation is planned to include a set of Web services profiles to assist organizations with the adoption of and support for key Web services standards. The WS-I has set a third-quarter release time for the first set of industry recommendations and example applications.

The WS-I hasn't evolved without its share of controversy. First, there is the noticeable absence of Sun from the WS-I list of members. Sun, feeling scorned for not being invited as a founder, has decided to distance itself from the WS-I at least for now.

There is additional concern that the WS-I will infringe on other organizations, specifically the W3C. I, for one, hope that the WS-I at least nudges the W3C to start cranking up their standards engine a couple more notches.

Specifications and Standards

XML, SOAP, and WSDL compose the current base standards for Web services. These specifications are widely accepted, and companies are implementing solutions based on these standards today. However, there is somewhat of a dilemma here. Vendors need time to provide implementations of these standards and get companies to accept and utilize the them in their enterprise. At the same time there are critical pieces that are missing or in need of enhancement. Balancing these aspects is a key challenge facing the standards organizations. These standards have been essentially idle lately as XML, SOAP, and WSDL have not undergone any published updates this year.

Service Description

The area of service description has been quiet recently. WSDL, while not yet a W3C Recommendation, is nonetheless in wide use. However, there has been one new specification proposed, the Web Services Endpoint Language, and with Microsoft and IBM behind it, it does merit consideration.

Web Services Endpoint Language

WSEL is an XML format for the description of nonoperational characteristics of service endpoints, like sequencing of operations, quality-of-service, cost, or security properties. These characteristics are necessary for composing Web services into larger business processes. This is a relatively new specification (developed primarily by IBM), which has made little progress since its announcement.

Service Discovery and Registration

An XML registry is an enabling infrastructure for building, deploying, and discovering Web services. The preeminent specifications for XML registries are the ebXML Registry and Repository standard and the UDDI specification. One new specification in this area is the Web Services Inspection Language jointly developed by Microsoft and IBM.

ebXML Registry and Repository

The ebXML Registry and Repository provides for both the storing and sharing of information. This is different from UDDI, which doesn't support the storing of documents. ebXML Registry and Repository version 2.0 was approved in January of this year.

UDDI

UDDI is an industry specification for description and discovery of Web services. UDDI is itself a SOAP/XML Web service designed for use by developer tools and applications. UDDI is currently in version 2.0. There is a version 3.0 in progress and security will likely figure prominently in this version. Version 3.0 is intended to be the final version before the UDDI community submits the XML business registry specification to a standards body, probably the W3C, for approval.

> "The good news is that the base Web services standards are generally agreed upon and significant work is happening in the area of security"

Web Services Inspection Language

WSIL is a new specification that defines the ability to inspect a site for available services. WSIL will enable developers to easily browse Web servers for XML Web services. While this may seem orthogonal to the aforementioned UDDI, WS-Inspection complements UDDI by enabling the discovery of available services on Web sites unlisted in the UDDI registries, which defines most Web sites offering Web services today.

Security

Security is a crucial piece of Web services architecture that has been intentionally lacking from early specification efforts such as SOAP, WSDL, and UDDI. Security is frequently cited by companies as the most critical piece missing from the Web services story and has been an area of high activity so far this year.

The W3C is developing a set of security specifications that are crucial to public acceptance of Web services. These include XML Signature, a standard for digital signatures that is now a Recommendation; XML Encryption (a Candidate Recommendation), a set of standards for encrypting and decrypting XML documents and data; and XML Key Management (a Working Draft), which enables retrieval of key information from a Web service.

Security Assertion Markup Language

OASIS is developing an XML-based security standard for exchanging authentication and authorization information. The Security Assertion Markup Language (SAML) currently has a great deal of momentum, and there are several implementations available in a number of products including those from Netegrity and Systinet.

WS-Security

Once again industry giants IBM and Microsoft have taken the initiative in driving Web services standards, this time in the area of security. WS-Security was a joint announcement by Microsoft, VeriSign, and IBM. WS-Security defines a set of SOAP extensions and describes how to exchange secure and signed messages in a Web-services environment. Microsoft has stated that this work will be delivered to a standards organization, but no specifics were provided. In addition, Microsoft and

IBM announced plans to deliver other security specifications. In particular, six specifications have been identified.

WS-Policy, WS-Trust, and WS-Privacy

The first three specifications address security policies: WS-Policy will define how to express the capabilities and constraints of security policies; WS-Trust will describe the model for establishing both direct and brokered trust relationships (including third parties and intermediaries); and WS-Privacy will define how Web services state and implement privacy practices.

WS-Secure Conversation, WS-Federation, and WS-Authorization

The last three specifications involve the sending and receiving of messages between Web services. WS-Secure Conversation will describe how to manage and authenticate message exchanges between parties, including security context exchange and establishing and deriving session keys; WS-Federation will describe how to manage and broker trust relationships in a heterogeneous federated environment, including support for federated identities; and WS-Authorization will define how Web services manage authorization data and policies.

Resource Provisioning

Resource provisioning is software that enterprises can use to centralize and manage the process of supplying – or provisioning – users with access to corporate systems and data. The challenge of resource provisioning only becomes more complex when you consider emerging B2B scenarios, in which a user might come into a system from outside the firewall. Even more complex are emerging Web services architectures, where not only users but also other bits of code may need access to corporate systems as part of a composite application.

Service Provisioning Markup Language

Emerging to address this problem of distributed provisioning is the OASIS standards group, which last year convened a new Provisioning Services Technical Committee. The group is defining an XML-based framework for exchanging user, resource, and service-provisioning information, dubbed Service Provisioning Markup Language (SPML). A major goal of the group is to define the way provisioning works in a Web services environment.

Data Access

Quick and easy access to data is essential to Web services integration efforts. Database vendors are continuing to evolve their products based on standards such as XML Schema, XML, XSLT, and XPath. Two areas that have been getting increased interest are the XML Query and SQLX specifications.

> **"Web services is here for the foreseeable future. Let's hope that the standards process will proceed in an efficient manner and continue to produce great standards"**

XML Query

XML Query or XQuery is a W3C specification that provides a vendor-independent method for query and retrieval of XML data. A key component of XML Query is XPath, another W3C specification. The data model that XQuery uses is based on that of XPath and defines each XML document as a tree of nodes. XML Query has been moving through the W3C rather slowly. There have been a total of eight working drafts delivered to date, but the main document was last issued in December of last year.

SQLX

SQLX defines SQL mappings to XML, as well as mappings from XML to SQL. The intent is to integrate XML and SQL and to make the SQL language capable of handling XML data and making XML extensions, or XPath expressions, part of the SQL language. SQLX is sometimes also referred to as SQL/XML.

Conclusion

This has been a relatively quick tour of the standards landscape. The good news is that the base Web services standards are generally agreed upon and significant work is happening in the area of security, which has been high on everyone's list of concerns. The concern is that most of the early work was accomplished during a period when there was a comparatively small core of companies driving the specifications. This is certainly not the case anymore, and many of the big players will not be content to follow only Microsoft's and IBM's lead in the Web services space.

Noticeably missing from this article were new developments in other areas such as transactions and business process management. It's not to say that work isn't occurring in these and other areas, but it hasn't led to new public specifications. For now, beyond the core, established standards of XML and SOAP, Web services is somewhat of a mixture of unofficial standards, such as WSDL and UDDI, and vendor-specific implementations of prominent specifications such as SAML, WSFL, XLang, and ebXML.

Web services is here for the foreseeable future. Let's hope that the standards process will proceed in an efficient manner and continue to produce great standards and that the vendors will abide by these standards and provide great implementations. There is still a great deal of work to be done but the potential payoff is huge.

written by Jim Culbert

Web Services Technologies You Can Use Today

Laying the foundation for tomorrow

Web services hold the promise to revolutionize the way architects build systems and how software is delivered and sold. While the full realization of these advances will take years to play out, most of these benefits are rooted in technologies that are available today. This article will look at what can be achieved today using Web services technologies to prepare yourself to take advantage of Web services as the standards and tools mature. We will discuss five things that you can do to take advantage of Web services core technologies and point out trends to pay attention to in the near future.

Jim Culbert, vice president of technology at MetraTech, is responsible for technology choices and architectural issues. He joined the company in 1998 as vice president of enginneering, directed architecture, and design, and helped launch the first version of MetraTech's billing solution. He has more than 12 years of experience in Internet application development from the network layer through complex Web services.
JIM.CULBERT@METRATECH.COM

Five Things You Can Do Today

XML for Data Interchange

If your system can export and import data as XML then you can take advantage of many of the benefits of existing XML technologies. One critical problem faced by IT managers is the blizzard of data formats that their systems generate. Whenever two systems need to communicate with one another, software needs to intervene and convert the syntax and meaning of data from one system and convert those data into the language and dialect understood by the other. While this is a fairly straightforward programming assignment, it consumes time and money, and can introduce spectacular errors into your operations.

XML technologies can help a lot here. XML and XML Schema provide a framework where architects can unambiguously define data formats. The format definition, written in XML Schema, is human-readable but is specifically designed to be processed by computers. This contrasts with the current state of practice where data formats are described in a format that a programmer can understand (e.g., Word document, PDF). Using this definition, a programmer can write software to process the data. With XML formats, data are described in XML Schema which can be interpreted and acted upon very easily by XML components that are available today. The benefit is that a broad collection of data-processing tasks can be automated. Schema-aware software can read a schema document and automatically create code to help the programmer convert and process disparate data formats. In some cases, programmers will be able to handle data parsing and integration tasks without writing a single line of source code.

Developers can realize several benefits by standardizing on XML as a data interchange format.

1. Consistent rules for creating formats that can be enforced across an organization
2. No need to create custom data parsers. Off-the-shelf parsing technologies from major vendors are abundantly available.
3. XML transformation technologies (XSLT) can be used to solve common data translation and transformation tasks with minimized coding.
4. Tools from major vendors (Microsoft, IBM, Open Source) can read schema files and auto-generate code. And they are able to greatly accelerate tasks that involve manipulating XML-formatted data.

XML as a New Interface Type

Even without strict adherence to emerging SOAP and WSDL specifications, your applica-

tions can use XML and HTTP as a sophisticated new interface type that gives you the loose coupling benefits of Web services. This approach can be easily migrated to native protocols when your organization is comfortable with their maturity. For many years Web developers have leveraged the plumbing that supports Web browsing to also implement distributed systems. By leveraging HTTP as a readily accessible transport (heck, everyone has a Web server) and SSL as a secure pipe, developers have implemented secure conversations among distributed applications for a number of years. Web services and technologies such as SOAP and WSDL formalize some of the ad hoc mechanisms that we've used for years, but the underlying mechanisms are the same.

Brew Your Own

Whether you adhere to the spiffy new protocols or not, you can easily gain the same benefits in your system designs by leveraging HTTP as a transport and XML as your payload. You won't automatically interoperate with emerging Web services technologies such as WSDL, but you may in fact get a better understanding of how to program in this model without getting overly confused by the acronym soup of Web services. As many Web developers know, there are plenty of tools that, in less than a dozen lines of code, allow a developer to post an arbitrary text message to a Web server. By placing XML in this message you now have an extensible mechanism where one program can talk to another. Sure, there's a Web server in between but this "protocol handler" simply strips the XML payload out of the message and passes it on to a program (cgi, xSAPI, dll, jsp, asp, etc.). A program receiving one of these messages can read the message, act on it, and send information formatted in XML back to the sender in the HTTP response. Tada! You now have a Web service.

Now, to the Web services purists out there I'm a DBWS (Dirty Blaspheming Wayward Soul) for speaking like this. Why am I advocating anything other than "The Way"? Here's why. One of the fundamental benefits of Web services is its support for a distributed, loosely coupled architectural model that is interoperable. The main enablers of this interoperability are the ubiquitous deployment of HTTP servers and XML as a data standard. While SOAP formalizes the packaging of messages sent among machines and WSDL formalizes a way to advertise the nature and availability of a service, these are secondary to the core benefit of being able to implement loosely coupled distributed architectures using a single, standardized data-modeling language (XML Schema).

> **"One final sacrilegious note: the software industry risks killing the Web services golden goose"**

The industry has developed so much HTTP and HTML processing software that all major programming environments (Java, C++, C#, Perl, Python, etc.) have rich, capable tools for moving data around over the Web. Implementing a home-brew Web service requires less than 50 lines of server and client code. Finally, once you've adopted the Web services design paradigm, making the hop to SOAP and WSDL (and whatever other helper protocols and technologies may emerge in the future) is trivial.

One final sacrilegious note: the software industry risks killing the Web services golden goose. One of the largest factors that drove adoption of HTML and HTTP was that they were exceedingly simple to implement. A developer could open vi, tap in a handful of HTML tags, and in a few minutes create globally available, hyperlinked content. Once developers became comfortable with HTML and simple editing, we began demanding more sophisticated features from our tools and the protocols. We evolved these technologies based on the intimate understanding we gained through real-life experience. With today's Web services activities many tools vendors, ISVs, and protocol developers are trying to jump right to the end of the game. By providing developers with all-encompassing developer tools, protocols, and products, we're potentially robbing the industry of the opportunity to become familiar with and to truly understand the underlying technologies. OK, enough said.

Decouple Your GUIs

Good design practice recommends that when we build applications, we separate business logic from UI logic. We figured this out pretty well in client/server applications but regressed substantially when we moved to Web technologies. For various reasons, it was difficult to build a well-separated Web application. Decoupling your Web-presentation logic from your business logic brings significant benefits to your Web application user interface design.

Whether you use Web services protocols like WSDL and SOAP or just brew your own XML messages and transports, one of the most important places where Web services can impact system architecture is at the GUI. By splitting the presentation tier from the business tier(s) using Web services, you derive two important benefits. First, the model enforces the separation of business code and display code. Despite best intentions, many implementations end up with undesirable coupling between interface code and business code. The result is not only inelegant but also can prevent a Web interface implementation from being easily scaled using Web farming techniques. Web services avert this issue by placing XML messaging between display logic and business logic.

Another benefit of exposing the business tier to the presentation tier via Web services is that Web services are presentation technology agnostic. While you may choose to implement all your presentation code in server-side JSP, a partner or customer may want to display some of your information in their Web-based GUI that uses another technology. If you use JSP on Apache and your partner has a Microsoft IIS/ASP-based site, seamlessly integrating your Web portal with your partner's can be a challenge. Typically, you have to play games with frames or you have to coordinate closely with your partner so that the look and feel of both sites is comparable.

If your business tier exposes its information to the presentation tier as XML, however, then a partner can either choose to use your Web pages (and play around with frames or risk discrepancies between sites) or they can consume the XML that your business layer generates, massage it however they want, and present it "natively" within their user interface.

One final benefit is that for many applications, XML transformations (XSLT) can be used to format the XML data for display as HTML. This can be highly efficient and can lead to elegant and easily managed presentation logic.

XML for Data Modeling

XML Schema provides a sophisticated environment for modeling data of any type. It can be used to capture data models across an application including code modules, wire formats, and database tables. Today, there are few tools (outside of formal modeling tools) that can help us unify the different islands of data design that exist in any system implementation. For instance, on any given project a database designer may design tables using DDL, a C++ programmer designs data in header files, and the people responsible for data import/ export define data in Word documents. The same data may be represented in multiple formats within a single implementation. Nothing guarantees that these formats are compliant with one another and nothing guarantees that if the "master" format is changed the other formats are updated.

XML Schema holds a lot of promise to act as a single data-modeling environment from which environment-specific instances can be created. Consider the case where a customer contact data format is required. This format would be defined in schema. The schema would then be used to create the database tables to store the format, the C++ objects to manipulate the data in memory, and the wire format to exchange the data among multiple programs. Leveraging XML to unify data modeling gives system architects a common ground for data design.

Cool, Fast New Tools

I realize that I have just finished warning of the dangers of relying on tools without understanding the underlying bits. However, once you have the fundamentals under your belt, then the new tools and protocols, once they settle down, will offer significant developer productivity. For those ready to tackle these new tools, there are important new offerings on the market from vendors such as Microsoft and IBM. While many of these tools are still under development, others, such as Visual Studio .NET, are available and provide powerful tools to take advantage of XML and Web services protocols. These tools can shave weeks off of complex integration projects.

What Next?

Full SOAP and WSDL Compliance

Once you've become comfortable with core Web services technologies such as XML, you can graduate to other technologies without significant reengineering. SOAP and WSDL provide structure for XML conversations that are powerful as you extend your Web services to integrate with other XML-based systems.

In reality, the state of the world today is such that implementing SOAP and WSDL does not guarantee that you will be able to automatically work with another Web services–compliant tool. It's a bit like a transmission where the gears don't quite line up. Things are getting close to interoperability but there's still a lot of grinding going on. This should smooth out within the next 18–24 months as more real applications begin to use the tools and protocols and the kinks get worked out.

UDDI Integration

UDDI provides a directory for you to advertise your services. Though you may think you don't need this function unless you're a service provider, UDDI will be a very useful tool for intranet and extranet developers who embrace Web services. Watch out for the marketing hype however. The world today is neither ready for, nor in need of, a global registry where companies can expose the guts of their enterprise for casual browsing and integration (never mind close competitive scrutiny or outright hacking). Where UDDI will be most useful initially is within the enterprise as a mechanism for IT to publish services company-wide (think of things like benefits information, phone lists, conference room schedulers, etc.). Once IT has sorted things out on the intranet, extranet use will follow. UDDI (administered "behind the firewall") will be used to expose limited access to business functions (e.g., inventory, payments) to partners.

Web Services Interoperability Organization (WS-I)

It's not certain how the Web services landscape will eventually look. One place to look for some idea, however, is the Web Services Interoperability Organization, formed by Microsoft, IBM, BEA, and Intel. It aims to guarantee the consistent application of Web services in the industry. You can expect that this group will "bless" protocols as well as promote those under development by member organizations. You can expect to see numerous protocols that are currently under development at Microsoft under the aegis of GXA (Global XML Architecture) appearing as recommendations from this group. The issues addressed by the group will cover both the consistent adoption of Web services technologies as well as the identification/development of technologies missing today. Key among these will be Web services protocols and best practices for implementing security, message routing, and transactions.

Conclusion

The utter din of Web services proclamations has put off many a rational IT manager. Most people are taking a wait-and-see approach and are not attempting to build anything using these technologies today. This is unfortunate since core technologies like XML, HTTP, SSL, and other bread-and-butter Internet technologies can be leveraged in the Web services pattern to create better system architectures as well as to prepare IT organizations to leverage Web services when the tools and protocols fully mature.

written by Murali Janakiraman

Building Blocks

An overview of Web services technology

In the past two years, we have witnessed an explosion of Web services and XML communication technologies. While WSDL , SOAP, and UDDI have become the accepted bases of Web services, there are even more standards in the making.

This article is the first of a two-part series that examines the Web services technological space in order to provide an overview of some of the major Web services standards now in progress in various organizations and consortiums across the country.

Murali Janakiraman is Rogue Wave's software architect for the XML Products team. He has been a developer, senior developer, and tech lead on almost all of Rogue Wave's database products, including DBTools.h++, JDBTools, DBTools.h++ XA, and RW-Metro. For the past five years, Murali has focused on databases, distributed transactions, and object-relational mapping.
MURALI@ROGUEWAVE.COM

General Classifications of Web Services

Web services technology can be broadly classified into three main groups, as shown in Figure 1.

The description stack deals with a wide range of technologies that describe Web services in order to facilitate their common use for business process modeling and workflow choreography in B2B collaborations. The discovery stack deals with technologies that allow for directory, discovery, and inspection services. The wire stack consists of technologies that provide the steam for the runtime engines of Web services.

Figure 2 breaks these stacks into their subcomponents. Many of the available Web services technologies can be mapped to these stacks, although not all stacks have a corresponding specification or technology.

Functional Classifications of Web Services Technology

Another useful way to organize the Web services technology space is according to function. We are interested in understanding why these technologies matter, what their purpose is with regard to solving business issues, and what their relationship to one another is. If we want to create a basic Web service, for example, what kind of technologies exist that can help? If we want to upgrade our basic service to a mission-critical Web service, what steps must we take?

The list below shows the various functional areas of the Web services technology space. This list is not comprehensive, but it does cover most of the available technologies.

- Basic service
  – Service description
  – Communication protocols

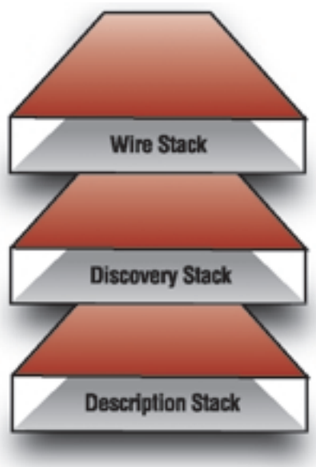


FIGURE 1 | Classification of Web services technology

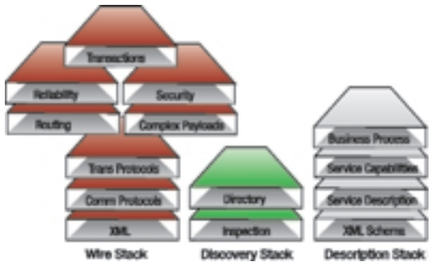


FIGURE 2 | **Web services stacks**

– Transport protocols
- Complex payloads
- Discovery
  – Inspection
  – Directory services
- Enterprise strength
  – Transaction
  – Security
  – Reliability
  – Routing
- B2B collaboration
  – Process modeling and orchestration

In the remainder of this article, I'll address each of the functional categories of Web services technology and discuss the standards that apply.

Basic Service

There are two main groups of technologies we must consider to create a basic Web service: service description and communication protocols. Transport protocols aren't specific to Web services, and hence aren't covered here.

Service Description

Service description standards specify what a service is about, what actions are supported by the service, what input and output parameters the service takes, and how the service deals with error conditions. In 2000, IBM and Microsoft came out with competing technologies: NASSL and SDL, respectively. Fortunately, they were soon merged into WSDL (Web Services Description Language). As of today, WSDL stands as one of the widely adopted technologies for describing Web services. WSDL 1.1 has been submitted to W3C, which has recently started a working group to ratify it.

WSDL takes a two-step approach to describing Web services. The first step is to provide an abstract definition of services and the data format; the second is to bind this abstract definition to concrete protocols. This two-step process permits reuse; it's possible to have many similar Web services based on one abstract definition with each implemented using different protocols.

WSDL is independent of any network and communication protocols, although it does define default binding to HTTP, SOAP, and MIME. Similarly, WSDL isn't tied to any type of system, although it does use XML Schema. WSDL is designed to be extensible to work with different types of systems and other network and communication protocols. Listing 1 demonstrates the simple WSDL grammar used to describe services. *Note:* This example isn't complete and won't parse; more namespaces need to be defined.

In Listing 1 the message element, along with the part element, defines the data in abstract terms. The operation element defines the action supported by the service. WSDL defines four basic operations: one-way, request-response, solicit-response, and notification. The portType element acts as a container for a set of abstract operations. In this example, we define a portType element, "StockQuotePortType", with a single operation, "GetLastTradePrice", which takes an input message, "GetLastTradePriceRequest", and gives an output message, "GetLastTradePriceResponse".

These abstract definitions are then bound to concrete protocols using the binding element. The port element captures the communication endpoint details, and the service element contains a list of related ports. The types element (not shown) acts as a data container holding various data type definitions. In the example, the operations in "StockQuotePortType" are bound to SOAP and HTTP.

WSDL enjoys the support of many tools. Some help generate WSDL from existing Java and C++ classes, and others generate Java and C++ classes from WSDL documents.

Communication Protocols

Standards in the communication protocols area deal with message format and serialization details. In order for a receiver to correctly parse and digest a message, the format of the message must be known. In contrast to the service description area, many protocols have been published in the communication area. These protocols include XML-RPC, SOAP, the ebXML messaging specification, WDDX, and Jabber.

XML-RPC is an XML-based RPC protocol based on HTTP POST with a simple data model that came from Userland software in 1998. Compared to SOAP it is simple; in addition to RPC, SOAP provides much richer processing semantics, an enhanced data model, and support for messaging. SOAP has garnered a great deal of attention and a huge user base.

The ebXML messaging specification, built on top of SOAP, is one part of a set of ebXML specifications. WDDX, an effort from Allaire, is focused on providing a simple, lightweight data exchange mechanism for Web programming languages such as ColdFusion, ASP, Perl, and PHP. Though RPC semantics can be layered on top of WDDX, it isn't as widely adopted as SOAP for RPC purposes. Jabber is an open-source protocol that enables exchange of structured information in a near-real-time manner between two or more end points. Jabber is used in the instant messaging areas.

Let's look at SOAP in detail, since it is the protocol of choice for most Web services.

SOAP, the Protocol of Choice

SOAP has a come a long way since its 0.9 release by Microsoft in 1999. SOAP is now handled by the W3C, which was close to publishing a last-call working draft of SOAP 1.2 at the time of this writing.

SOAP is a lightweight XML-based communication protocol for the exchange of information in a decentralized, distributed environment. SOAP is neutral with regard to language, platform, and programming model, allowing both the sender and the receiver to operate in their environment of choice. SOAP documents can be exchanged over many transport protocols.

The SOAP specification can be broadly classified into four main parts:

- A framework for describing the content of a message and how to process it
- A simple data model and a set of encoding rules for serialization
- A convention for representing remote procedure calls and responses
- A binding to HTTP

The SOAP "grammar" can be best demonstrated by a SOAP message, as shown below.

```
<env:Envelope
xmlns:env="http://www.w3.org/2001/09/soap-
envelope" xmlns:app="www.rwav.com" >
  <env:Header>

<app:transactionId>010001</app:transaction
Id>
  </env:Header>

  <env:Body>
    <app:getStockQuote>
      <app:ticker>RWAV</app:ticker>
    </app:getStockQuote>
  </env:Body>
</env:Envelope>
```

In this example, the SOAP message is identified by the namespace-qualified root element "Envelope". The Envelope namespace determines the version of the SOAP specification to which a SOAP message conforms. The header element is optional; it is typically used to carry out-of-bounds information, such as transaction or security information. The header can contain any number of namespace-qualified XML elements, called entries or blocks. The above example contains one header entry named "app:trans actionId". The body element contains the essence of the message intended for the endpoint. Unlike the header element, the body element must be contained in every SOAP message; the body element can contain one or more namespace-qualified XML elements, called *entries* or *blocks*. The above example contains one application-defined body entry named "app:getStockQuote". SOAP defines one body block, called Fault, to represent errors.

As part of its encoding rules, SOAP defines a simple data model consisting of simple types, compound types similar to structs in programming languages, an array type, and an ID/HREF type that represents references. The encoding rules define a particular serialization rule for this data model. SOAP data model and encoding rules are optional. SOAP defines an "encodingStyle" attribute under the "env" namespace, which can be used to specify a particular encoding rule in effect for a specific element or group of elements.

Like encoding rules, the RPC conventions defined by SOAP are optional. In SOAP, both the request and the response of an RPC call are modeled as structs; they can also be modeled as arrays, according to recent changes in the SOAP specification. The name of the struct represents the name of the method being invoked. The parameters of a request or the results of an invocation are modeled as named accessors inside the struct. Our example message is an RPC request defined according to SOAP-RPC conventions. Though SOAP has defined a set of conventions for RPC, SOAP is not RPC-centric. It can be used for any general-purpose messaging.

SOAP can be exchanged over many transport protocols, but the SOAP 1.2 specification defines a binding to HTTP and provides an e-mail binding.

The W3C working group on SOAP is expected to publish their recommendation around August 2002. To participate or follow their progress, go to www.w3.org/2000 /xp/Group.

Complex Payloads

So far we've looked at technologies that help create a basic XML Web service. The data exchange format and the message format in all these technologies is XML. But not all of the world's data is in XML. We have legacy systems, EDI systems, images, and many more formats. How can we use these new technologies for non-XML data? Converting all this data into XML is inefficient and time consuming; also, XML may not be the best representation for all kinds of data. For example, JPEG may make better sense for images. Even sending arbitrary XML could be a problem. We cannot simply take one XML document, insert it into another, and expect to end up with a valid XML document. Even to carry arbitrary XML in XML-based protocols such as SOAP, we need help.

There are at least two technologies that address this space:

- SOAP with Attachments
- DIME

SOAP with Attachments

SOAP with Attachments (SwA) was an effort by a group of individuals to combine the existing SOAP and MIME technologies to facilitate carrying arbitrary data in SOAP. The W3C has published SwA as a W3C note.

SwA doesn't introduce any new technology. Rather, it uses the referencing facilities in SOAP (HREF attribute) and Multipart MIME (RFC 2045) to make it possible to carry arbitrary data. The whole message is constructed as a multipart MIME message with the SOAP message as the root part. The MIME message can have any number of MIME parts, and the SOAP message can refer to any of these parts using the HREF attribute. In addition, the specification places a few more constraints (such as content-type and start parameter), and makes some recommendation on how the reference URIs in the HREF attribute can be resolved using existing RFCs.

Listing 2 shows a SOAP 1.2 message with an attached facsimile image of a signed claim form (claim061400a.tiff).

Until recently, SwA was the most popular way to carry arbitrary data in SOAP; now DIME seems to be shifting the balance. The W3C has not yet started any work on SwA.

DIME

DIME (Direct Internet Message Encapsulation) came from Microsoft and is published as an Internet-Draft by the Internet Engineering Task Force (IETF). DIME is a packaging protocol for multiple binary records with a fixed format and a variable record length. DIME allows for chunking, a process in which data is streamed out without having to be held in memory to calculate the maximum length. DIME has "begin record" and "end record" boundaries so the records can be assembled in

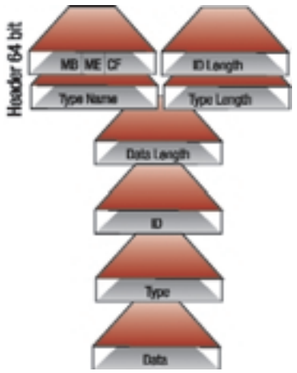


FIGURE 3 **ID, Type, and Data starts on 32-bit boundary**

order at the receiving end. Figure 3 provides details of the DIME record structure.

The MB, ME, and CF fields are bitmasks indicating the "begin", "chunk", and "end" of records. The Type Name field is a 3-bit field indicating the structure of the value of the type field. DIME provides a numeric value mapping for different media and MIME types. The ID field is used to give an identifier for each DIME payload. The maximum size of the data field is limited to 4GB.

Microsoft has also published a companion Internet-Draft that shows how SOAP messages can use DIME to send arbitrary data. Using SOAP with DIME is somewhat similar to using SwA. In both cases, the SOAP message is wrapped in a compound structure with the SOAP message as the root or first message, and the referenced parts as the second. DIME specifies rules for resolving the URIs referenced through HREF attributes in SOAP messages; they are similar to SwA rules (RFC 2396 and 2557). DIME also adds on to SOAP-HTTP binding semantics by specifying the content-type as application/dime, rather than the default text/xml specified by SOAP.

It's important to note that in both SwA and DIME, the SOAP message itself travels as either a MIME or DIME message with respect to the carrier or transport protocols.

SwA Versus SOAP with DIME

DIME is designed for simplicity, with SOAP and XML Web services in mind, while MIME offers great flexibility. DIME makes data handling a bit easier, as it requires that the data length be specified. DIME also makes parsing easier, since it's easier to identify the boundaries of different records using data length, rather than scanning for the string separators used in Multipart MIME to separate data records. The compulsory inclusion of data length may also help in heap management. DIME does not require encoding of binary data, and hence may be faster. MIME on the other hand is very flexible and well-understood, with many implementations supporting it.

Discovery

If programmatic discovery of Web services needs to be supported, there are a few technologies that can help:

- WS-Inspection (WS-I)
- UDDI
- ebXML-Registry and Repository Specification

UDDI has garnered as much attention as SOAP and WSDL; the three together are now considered the basic building blocks of any Web service. Though ebXML has a registry specification of its own that can be used as a standalone specification, it hasn't attracted much attention. WS-I is a companion technology to UDDI that addresses a specific purpose in the area of service discovery. I'll focus on UDDI and WS-I.

UDDI

UDDI (Universal Description, Discovery, and Integration) is an effort by a group of companies that hasn't yet been submitted to any other consortium or standards body. UDDI consists of an XML Schema that allows a user to provide a description of a Web service along with its business information. The description of a service in UDDI schema takes a business-centric view, whereas WSDL takes a functional view. In addition, UDDI also defines an API specification that allows a user to publish Web services and to query and obtain information on other published services. UDDI publish/query is based on SOAP messages.

The other face of UDDI is the repository itself. A repository implements the API specification with which users can publish or discover services. UDDI repositories are logically centralized and physically distributed. As of this writing, there are four node operators running UDDI registries: Microsoft, IBM, SAP, and HP.

The UDDI repository implementations are open source; users could get them and run their own in-house UDDI repositories. Though UDDI was initially touted as the technology that would open the gates for dynamic discovery of Web services and dynamic collaboration, it is more and more frequently used for Intranet and in-house repository needs.

WS-Inspection

WS-I was a joint offering from IBM and Microsoft released in 2001. While UDDI involves going to a central place to publish and query about services, WS-I involves going to a site offering Web services and seeking information about the services offered at there.

WS-I defines a simple grammar to aggregate service description documents of various services offered at that site. The service descriptions can be in any format, such as WSDL or UDDI. There can be many service descriptions per service, and many services can be defined in a single WS-I document. WS-I also defines an extended binding grammar for both WSDL and UDDI that provides hints about what may be found in the referred service description documents.

WS-I makes some recommendations on how its documents may be made available to users, so they are easily found. WS-Inspection documents may also be placed within a content medium such as HTML.

Conclusion

So far we have looked at technologies in service description, communication protocols, complex payloads, on-site inspection, and general discovery. In the next part of this series, we will look at technologies that deal with enterprise-strength issues, such as transactions and security, and technologies that cover routing and process orchestration.

Resources

- *W3C Note on WSDL:* www.w3.org/TR/ws dl.
- *W3C Working Draft on the SOAP Messaging Framework:* www.w3.org/TR/ soap12-part1.
- *UDDI home page:* www.uddi.org.
- *IETF Network Working Group Request for Comments:* www.ietf.org/rfc/rfc261 6.txt.
- *IETF Internet Draft on Direct Internet Message Encapsulation (DIME):* http://search.ietf.org/internet-drafts/draft-nielsen-dime-01.txt.
- *W3C Recommendation on XML Schema, Part 1:* www.w3.org/TR/xmlschema-1.
- *XML-RPC Specification:* www.xmlrpc. com /spec.
- *OASIS ebXML Messaging Services Technical Committee:* www.oasis-open.org/ committees/ebxml-msg.
- *The Web Distributed Data Exchange:* www. openwddx.org.
- *Jabber Software Foundation:* www.jabber.org.
- *W3C Note on SOAP Messages with Attachments:* www.w3.org/TR/SOAP-attachments.
- *IETF Interment Draft on Encapsulating SOAP in DIME:* http://search.ietf.org/internet-drafts/draft-nielsen-dime-soap-00.txt.
- *Web Services Inspection Language (WS-Inspection) 1.0:* www-106.ibm.com/developerworks/library/ws-wsilspec.html.
- *OASIS ebXML Registry Technical Committee:* www.oasis-open.org/committees/regrep/.
- *W3C Web Services Workshop position papers:* www.w3.org/2001/03/wsws-popa/.

Listing 1: Simple WSDL grammar

```
<definitions name="StockQuote" >

 <!…Assumes GetLastTradePrice is defined in XMLSchema …>
 <message name="GetLastTradePriceRequest">
  <part name="body" element="xsd1:GetLastTradePrice" />
 </message>
 <!-- similarly define a message for "GetLastTradePriceResponse" -->

 <portType name="StockQuotePortType">
  <operation name="GetLastTradePrice">
   <input message="tns:GetLastTradePriceRequest" />
   <output message="tns: GetLastTradePriceResponse" />
  </operation>
 </portType>

<binding name="StockQuoteSoapBinding"
          type="tns:StockQuotePortType">
  <soap:binding style="document" />
  <operation name="GetLastTradePrice">
   <soap:operation soapAction="http://my.org/GetLastTradePrice" />
  </operation>
 </binding>

 <service name="StockQuoteService">
  <documentation>My First Stock Service</documentation>
  <port name="StockQuotePort" binding="tns:StockQuoteBinding">
   <soap:address location="http://my.org/stockquote" />
  </port>
 </service>

</definitions>
```

Listing 2: SOAP with attachments

```
    MIME-Version: 1.0
    Content-Type: Multipart/Related; boundary=MIME_boundary;
type=text/xml;
          start="<claim061400a.xml@claiming-it.com>"

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <claim061400a.xml@claiming-it.com>

<?xml version='1.0' ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
..
<theSignedForm href="cid:claim061400a.tiff@claiming-it.com"/>
..
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--MIME_boundary
Content-Type: image/tiff
Content-Transfer-Encoding: binary
Content-ID: <claim061400a.tiff@claiming-it.com>

...binary TIFF image...
--MIME_boundary—
```

Download the code at

sys-con.com/webservices

Reviewed by Joseph A. Mitchko

About the Author:
Joe Mitchko is a lead engineer with Nekema, Inc., a leading Web services technology provider for the insurance industry sector. Joe is also the product review editor for ***Web Services Journal***.
JOE@SYS-CON.COM

BEA WebLogic Workshop

A CASE for the 21st Century

Once upon a time, back before the turn of the century, there was a buzzword in the industry called CASE – computer-aided software engineering. In a nutshell, CASE would take the various models and requirements gathered by software analysts and automatically generate production-ready application code. At the time, the concept had me scared to death. The idea, of course, was that if you could get the software to write all the code, you wouldn't need any programmers. Luckily, the idea fizzled out like most of the crazes to hit the industry. Or did it?

Software has become so complex that it's increasingly difficult to develop and maintain the literally hundreds of modules and configuration settings that comprise today's sophisticated Web application. Try to deploy the application as a Web service, and the problem compounds itself with all the SOAP-related XML protocols. It seems as though we mere mortals cannot grasp the entire picture alone, and those of us who have managed to get such a system into production spend a good part of our time plugging up the holes in the software dike. Luckily, help is on the way, in the form of software tools and products that not only assist us in the overall design and development of Web services, but do most of the heavy lifting for us when it comes to configuration and deployment. This is what BEA Workshop is all about.

Architectural Overview

In order to fully appreciate the power behind Workshop, you need to know a bit about Java Web Services (JWS), an up-and-coming standard in the J2EE world. Similar to how Java code is embedded in a JSP file, Java code contained within a JWS file is interpreted by the application server and deployed as a fully functioning Web service.

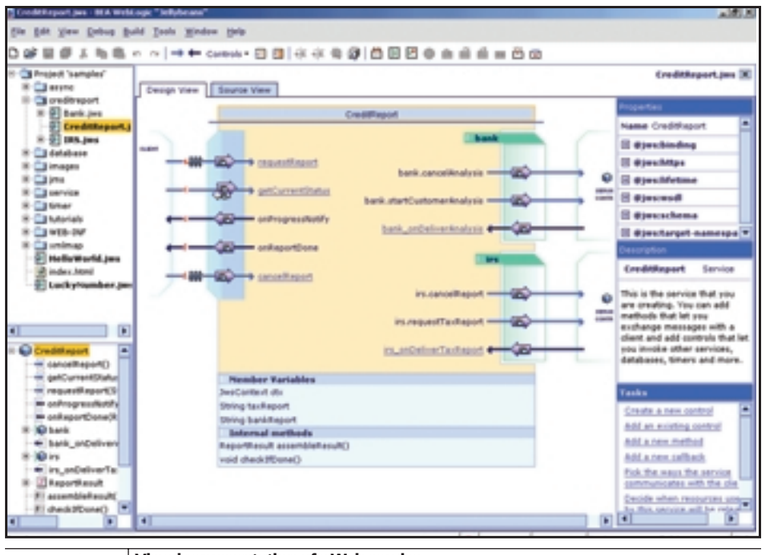


FIGURE 1 | **Visual representation of a Web service**

CONTACT:
BEA Systems, Inc.
2315 North First Street
San Jose, CA 95131
Tele: 1.800.817.4232
Web: www.bea.com
E-mail: sales@bea.com

DOWNLOAD INFO:
http://commerce.bea.com/downloads/weblogic_workshop.jsp

TEST ENVIRONMENT:
OS: Windows-XP
Hardware: Dell Inspiron 8000

JWS allows you to take a standard method call in a Java class and, by adding one or more JavaDoc-based annotations, instruct the application server to expose the method as a Web service port, taking care of all of the details.

Another important feature of the JWS standard is XML mapping. Unlike some Web service integration products that do not expose the XML too readily, Workshop allows you to bind an element in the SOAP message directly to a method parameter. This allows the service to maintain its public contract (the underlying SOAP interface) while making changes to the implementation. For a good introduction, see the article "JWS: Web Services in Java" in the April 2002 issue of ***WSJ*** (Vol. 2, issue 4).

Features

Workshop comes with an integrated development environment – a Design View – that contains a visual representation (see Figure 1) of the Web service, a runtime framework, control architecture, XML mapping facility, integrated testing and debugging, and more. Using the underlying JWS architecture, Workshop allows developers to create and deploy Web services just by creating and configuring objects in a "painter-like" interface. Within the Design View, you set up one or more public interfaces for the Web service and connect various control interfaces to EJB components, database objects, etc., to the service. The underlying Java code you write integrates the various control interfaces into a functioning Web service (see Figure 2).

The beta version I used for this review came bundled with a pre-release version of the BEA WebLogic Server 7.0 and WebLogic Builder, which is a graphical tool for configuring and deploying J2EE application modules.

Building a Web Service

Building and deploying your Web service is easy and seamless. Initiate the build and you're only a few seconds away from testing the interface in the test harness. If all goes well when you compile the JWS file (i.e., no coding errors on your part), the process of building and deploying a service works each and every time. Not bad for a beta version.

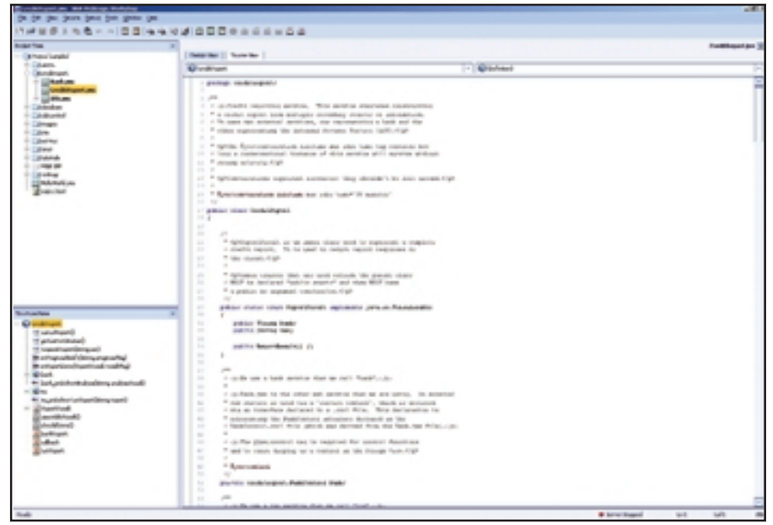


FIGURE 2 | **Source View**

Long Live the Transaction

Out of the box, Workshop provides you with the ability to easily set up and manage long-lived, asynchronous transactions. Transaction state and management is maintained by the Workshop JWS framework, so there isn't much you need to do to set one up. To start a conversation, just set up the appropriate property values in the Design View. Parts of the Web service need to run single threaded? No problem. Using the properties pane, just configure a JMS messaging queue into the SOAP operation and you're done. The underlying JWS framework will do all the work for you.

First Impressions

The first thing that hits you when you look at the Design View is how clean and visually appealing the GUI design is. It's one of those products that cause you to mutter "cool" under your breath when you first see it. The GUI design is well organized and very intuitive to use. The complete development cycle is quick and seamless in operation. You can easily run through a complete test cycle in under a minute.

Conclusion

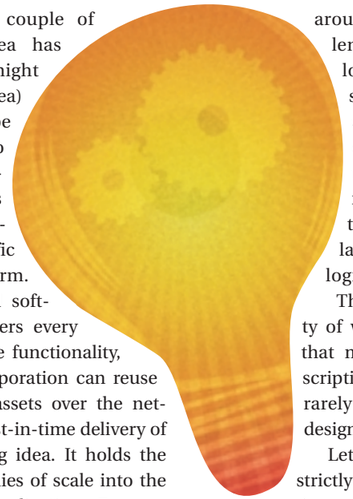
Workshop has the potential to be a very powerful tool in the development and deployment of large and complex Web services, where you can literally see how a Web service fits together and works. Combined with the new release of WLS, it becomes a very impressive platform indeed.

What's the Big Idea?

Make the Internet interesting again

Jeremy Allaire

Jeremy Allaire is the CTO of Macromedia, where he is instrumental in guiding Macromedia's product direction and is the company's primary technology evangelist, responsible for establishing key strategic partnerships within the Internet industry. Jeremy was the cofounder and CTO of Allaire Corporation, which merged with Macromedia in March 2001.
JEREMY@MACROMEDIA.COM



Over the past couple of years, an idea has emerged (some might argue it's an old idea) that software will be transformed into being used as services, rather than as monolithic applications tied to a specific machine or platform. Rather than install software onto computers every time we need some functionality, an end user or corporation can reuse other application assets over the network. The idea expands into the notion of just-in-time delivery of applications. "Software as services" is a big idea. It holds the promise of introducing radical new economies of scale into the manufacturing and distribution of software applications. From an end-user perspective, it's the idea that users can access information and applications anytime, anywhere, and from any device.

This is a powerful set of ideas. It's the idea that the value that we create in software can be freed from any single delivery platform. It's the idea that rich applications can be used easily across platforms and devices. How close are we today to realizing this set of ideals?

The Web Services Industry Today

The phenomenal focus on Web services over the past year is interesting in contrast to the actual set of technology available to fulfill this vision. In comparing the broader vision for software as services to the world of Web services technologies, such as protocols like SOAP and formats like WSDL, it looks like we're perhaps 25% of the way toward fulfilling that vision.

Today, Web services are primarily talked about through the lens of specific protocols – SOAP and WSDL, perhaps UDDI (though I have yet to find a customer who's really using UDDI in any significant way). This early focus and discussion is great – finding the holy grail of a common protocol for exchanging objects and data between platforms is an incredible achievement, and the fact that the industry is so focused on it gives us hope for the future of interoperable and open software.

But even the current crop of technologies implemented around these standards has problems and challenges. In the Web services model, application logic is well-structured, often as component services. The need to deliver well-structured applications has largely eluded Web application development. In fact, we all know that close to 75% of Web applications are not well structured – they're conglomerations of dynamic pages, including scripting languages, database code, and presentation logic all together.

There is a huge disconnect between the necessity of well-structured Web services and the reality that most Web applications are built using 4GL scripting languages, are usually unstructured, and rarely meet the requirements of thoughtfully designed, object-oriented systems.

Let's assume for a moment that Web services are strictly defined as the middleware standards for messaging and object marshaling – e.g., SOAP and WSDL. Even in that world, the majority of Web application developers aren't well positioned to both create and consume Web services without becoming full-on system programmers using complete object-oriented environments such as Java or .NET. The back-end world of Web services needs to evolve to make it simpler for RAD or scripting-level developers to easily create and consume Web services. It would be a huge victory for the industry if those 75% of Web applications built with scripting languages could also share their value and data with other applications.

However, the focus on Web services through this lens largely relegates the topic to discussions of classic back-end middleware, rather than a more holistic view of how to deliver software as a service. Strikingly absent from the discussion is any notion of what the end-user experience is of these "software services." In part, this is due to the type of vendors thinking about Web services – e.g,. traditional enterprise middleware companies – but it is also because of the hard work required by interoperable messaging and object protocols – these are necessary conditions to any future for software as services.

Getting back to the original vision of "software as services," it's apparent that we need a more holistic discussion and framework for thinking about Web services, one that actually includes end users using them. What would an end-to-end model for Web services look like?

The Missing Piece: Rich Clients and Web Services

Many of the early visions of Web services centered on the idea that, in the future, software could be used as a service, rather than as monolithic applications that needed to be installed and used on our desktop computers. While the Web has made progress on delivering applications easily to browsers, the world of Web services will deliver the experience of desktop-quality software that can be consumed as a service. Visions of using "productivity applications" as services were common interpretations by the industry. Examples such as Hotmail and Salesforce.com were cited as leading indicators – in this world of software services, end users could easily access rich applications from any desktop computer. These applications would always have their personal information, would be interconnected with back-end systems, and could even be portable across devices. This new world, however, would leave behind the document-based or page-based model of the Web for one that was much richer in terms of application capability, and that extended beyond the browser onto desktops and devices.

So, what is the user experience of Web services? What's the model for combining rich interfaces with back-end middleware to deliver exceptional new value for end users and the companies that serve them?

I believe that the perfect complement to "Web services as middleware" is the emerging category of rich clients. Indeed, it may well be that rich clients and Web services are two sides of the same coin – combining to enable this world of "software as services." What are these rich clients, and what do they require to transform the user experience and provide the logical front-end to back-end Web services?

Rich clients should combine rich content, applications, and communications in a single client environment. By rich content, I mean richly formatted text, graphics, audio, and video. By applications I mean rich, complex user interfaces – the kind we expect from a modern desktop computer – as well as application logic and data deployed in the network. And by communications I mean the ability for end users to interact with each other through these clients – to share data, text, audio, and video, in real time and non-real time. Rich clients should provide these capabilities in an integrated manner, where the applications that can target them far exceed what is possible in the world of HTML documents.

Rich clients should allow applications to run not only in browsers, but also as stand-alone applications on desktops and laptops. They should also support running on devices. To support these new Internet-connected application types, they should handle offline data storage, enabling occasionally connected devices and applications.

Most importantly, rich clients should anticipate the emerging world of back-end Web services by using a services-oriented architecture for integrating business logic and data across the network, whether that's simply contained in an application server, or actually a distributed Web service exposed through a protocol like SOAP.

Rich clients and Web services combined hold the promise of fulfilling the broader vision that the industry has for deploying software as services. The combination will enable rich, business-connected productivity applications. It will transform what's possible on the Internet today.

Software as Services: Make It Real

My company, Macromedia, is passionate and excited about using software as services. We believe in a holistic view that encompasses both the front-end user experience and the back-end integration layer. We hope that by making Web services approachable and affordable we'll all be able to take advantage of what they have to offer, not just the legions of advanced programmers and classic middleware developers. And remember, building for the next generation of the Internet should be fun – let's make the Internet interesting again.

WebServices .NET J2EE XML JOURNAL

PUBLISHER, PRESIDENT, AND CEO
Fuat A. Kircaali fuat@sys-con.com

CHIEF OPERATING OFFICER
Mark Harabedian mark@sys-con.com

CHIEF FINANCIAL OFFICER
Bruce Kanner bruce@sys-con.com

VP, BUSINESS DEVELOPMENT
Grisha Davida grisha@sys-con.com

ADVERTISING
SENIOR VP, SALES & MARKETING
Carmen Gonzalez carmen@sys-con.com
VP, SALES & MARKETING
Miles Silverman miles@sys-con.com
ADVERTISING DIRECTOR
Robyn Forma robyn@sys-con.com
ADVERTISING ACCOUNT MANAGER
Megan Ring megan@sys-con.com
ASSOCIATE SALES MANAGERS
Carrie Gebert carrie@sys-con.com
Alisa Catalano alisa@sys-con.com
Kristin Kuhnle kristin@sys-con.com
Leah Hittman leah@sys-con.com

SYS-CON EVENTS
VP, EVENTS
Cathy Walters cathyw@sys-con.com
REGIONAL SALES MANAGERS, EXHIBITS
Michael Pesick michael@sys-con.com
Richard Anderson richard@sys-con.com
CONFERENCE MANAGER
Michael Lynch mike@sys-con.com

CUSTOMER RELATIONS/JDJ STORE
MANAGER, CUSTOMER RELATIONS/JDJ STORE
Anthony D. Spitzer tony@sys-con.com
CUSTOMER SERVICE REPRESENTATIVE
Margie Downs margie@sys-con.com

WEB SERVICES
WEBMASTER
Robert Diamond robert@sys-con.com
WEB DESIGNERS
Stephen Kilmurray stephen@sys-con.com
Christopher Croce chris@sys-con.com
Catalin Stancescu catalin@sys-con.com
ONLINE EDITOR
Lin Goetz lin@sys-con.com

ACCOUNTING
ASSISTANT CONTROLLER
Judith Calnan judith@sys-con.com
ACCOUNTS RECEIVABLE
Jan Braidech jan@sys-con.com
ACCOUNTS PAYABLE
Joan LaRose joan@sys-con.com
ACCOUNTING CLERK
Betty White betty@sys-con.com

SUBSCRIPTIONS
SUBSCRIBE@SYS-CON.COM
1-888-303-5282
FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS, PLEASE SEND YOUR LETTERS TO SUBSCRIPTION DEPARTMENT
COVER PRICE: $6.99/ISSUE
DOMESTIC: $69.99/YR (12 ISSUES)
CANADA/MEXICO: $99.99/YR
ALL OTHER COUNTRIES: $129.99/YR
(U.S. BANKS OR MONEY ORDERS)

SYS-CON MEDIA

What's Online

www.sys-con.com/webservices

Join the fourth wave of technology and become part of the newest paradigm in software development!

Web Services Edge East 2002
International Conference & Expo
Plan to join us at the Jacob K. Javits Convention Center in New York City, June 24-27, for the largest Web services conference and expo of the year.

Conference tracks will offer invaluable information on Web services, .NET, XML, and Java.

WSJ2.com
Check in daily for breaking news from the *WSJ* News Desk on Web services products, industry events, developments, and happenings. Be the first to know what's going on!

WSJ Industry Newsletter
Subscribe now for more in-depth Web services coverage. The most up-to-date coverage of companies producing products and technology that are at the forefront of this paradigm. Brought to you every month by industry leaders.

Join the Web Services Discussion Group
Web Services Journal proudly announces WSJList, the NEW Web services mailing list community at the center of discussions, technical questions, and more...

Join the WSJList now to monitor the pulse of the Web services industry!

Digital Edition
Don't have your print edition? Can't wait to read the next issue? Our digital edition is just what you need. As long as you have your computer with you, you can read *Web Services Journal* anytime, anywhere.

Written by John Fou

Collaboration and Web Services Orchestration

A chance to redefine business processes

Collaborative commerce (c-commerce) is the name given to commercial relationships carried out over a collaborative framework to integrate enterprises' business processes, share customer relationships, and manage knowledge across enterprise boundaries. The ultimate aim of initiatives is to maximize return on intellectual capital investment, business agility, and the quality of the customer experience. C-commerce is far more crucial than basic B2B e-commerce, which is designed to construct a virtual link for a predefined community of trading partners to buy or sell goods and services. Even after the fall of the dot-com era, corporate strategists and venture capitalists are embracing c-commerce as the next generation of e-commerce and an evolution of the traditional supply chain process.

In Web services, c-commerce may have found an innovative way to redefine its business model. With this technology, c-commerce is given a solid platform to enable effortless and seamless integrations. Web services is going to reinvent c-commerce to offer new products, services, and multidimensional collaboration, bringing global enterprises a step closer to realizing the promise of increased velocity in supply chains and efficient interenterprise processes.

Lacking an open, reliable platform, traditional c-commerce vendors couldn't develop flexible, sharable, drag-and-drop modules among their products. Consequently, traditional c-commerce is a transaction-focused, one-dimensional application. The majority of applications reside in the direct or indirect procurement area. So far, the key factor behind most current exchanges is price. There is no opportunity to negotiate enhancements to products in order to have them match unique customer needs and requirements.

Web services–enabled c-commerce will, potentially, fully integrate the trading partner's intellectual library and the customer's demanding knowledge base to solve the above issues without human interaction. Web services orchestration is the infrastructure that assembles loosely coupled components and coordinates the multiple asynchronous conversations over coarse-grained communication protocols. C-commerce needs orchestration to deliver its value.

The first step of c-commerce is procurement. Our observation of this trend is based on the real practice of high-tech manufacturing, airlines, and the telecom industry.

Figure 1 details the purchase order flow process. The procurement diagram captures the interactive process between buyers and suppliers. The PO process statuses include PO create, PO change, PO cancellation,and PO fulfillment. The process is constrained as linear and synchronized by existing technology infrastructure.

Procurement Process Infrastructure

Most organizations use technology that is labor intensive and expensive. Typical mishaps include:

- Nonstandard one-to-one point integration
- A thick glue among suppliers and buyers
- No separation between strategic "business critical" purchases and tactical "business support" purchases
- Long latency between policy creation and process enforcement
- Developers having to hard-code all business exceptions
- Difficulty in scaling performance and functionality
- Lost opportunities for both customers and vendors

Web services orchestration takes these challenges to help businesses achieve their ultimate goals:

- Standards-based, multipoint, any-to-any integration
- Up-to-date supplier performance metrics
- Decision support that provides "business critical" information
- Asynchronous business process
- Elimination of errors and reconciliation costs
- Enforcement of policy deployment
- Ease of scaling and adapting functionality to new requirements

Benefits of Web Services Orchestration

Open Standards

Web services orchestration leverages your existing investments and skills in Java

AUTHOR BIO:
John Fou is the cofounder of an application solution company focused on telecom, financial services, and the supply chain area. He is authoring a book on Web services and supply chain with Prentice Hall.
JFOU@EDYNAMIC.CC

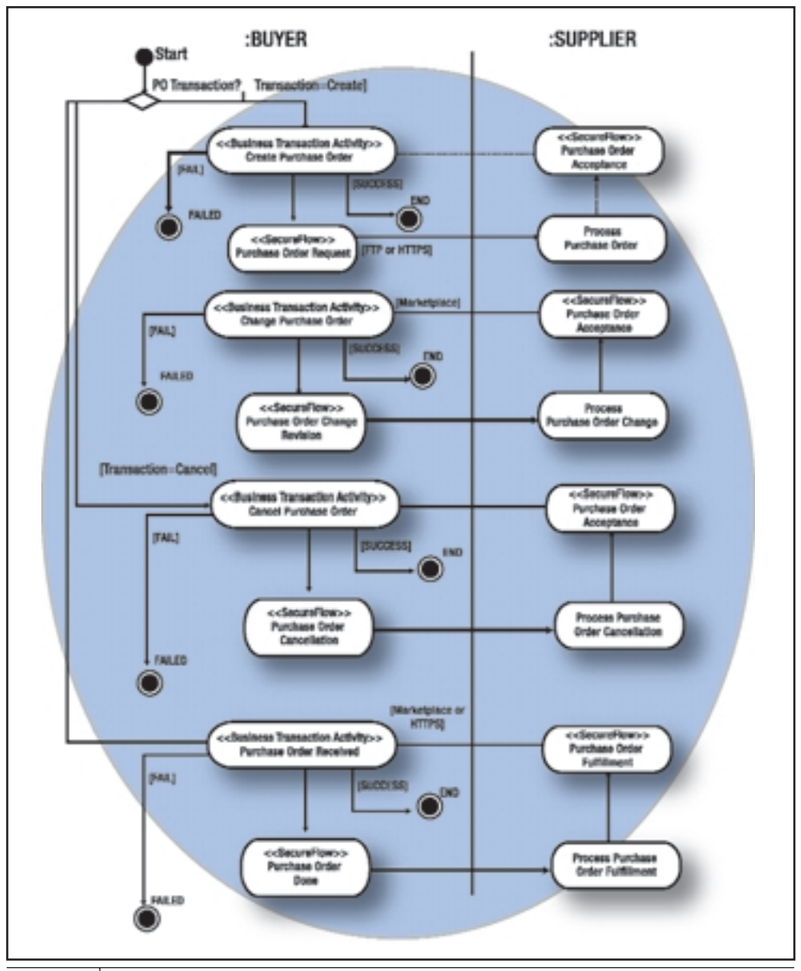


FIGURE 1 | **Purchase order flow process**

and J2EE. It's the platform that marshals conversation into SOAP, XML, and JMS messages so that the other side of the communication is independent of your implementation. The future procurement system will be presented as a Web service that can be integrated into any trading communities and partners.

State and Context Management

Purchase requests normally involve extensive negotiations, credit valuations, and a marketplace buying-power collaborative process. This entails an asynchronous conversation among communities. Web services orchestration will correlate these negotiations and provide trading partners with a set of actions.

"With Web services orchestration as an engine of integration, we can finally realize the value of c-commerce"

Loosely Coupled Services

A marketplace can easily handle millions of dollars in transactions. Web services orchestration simplifies business acquisitions and business spin-offs for technology departments. A loosely coupled services architecture provides the flexibility for business modules to be plugged or unplugged without disrupting the normal operation of transacting business.

Parallel Processing

When a large purchase order is received, the marketplace will be intelligent enough to split these purchase items into different suppliers, depending on pricing, delivery location requirements, and contract obligations. Web services orchestration provides a parallel-processing platform that optimizes the execution of the overall transaction.

Exception Management

Web services orchestration can handle business exceptions, such as an unacknowledged purchase order – for example, a PO that has not been confirmed, rejected, or fulfilled after four days. Other examples may include a PO delivery date that is earlier than the PO receiving date or purchase items that have been ordered but are no longer available.

Event and Notifications

Web services orchestration can provide business notification if one of the suppliers has merged with another supplier. It can also provide notification if one supplier has outsourced their product to a third party and the customer has to change the PO headers. Event notifications are particularly useful while conducting long-lived asynchronous conversations with external service providers, where prompt handling of logic in response to received events is essential to the execution of the business transaction.

Scalability

Web services orchestration provides the flexibility to scale up the procurement system. The transaction volume of procurement can pick up based on marketing promotions, distributor incentives, and seasonal fluctuation to provide for adequate quality of service. In addition, the procurement system can scale up in

"Web services–enabled c-commerce will, potentially, fully integrate the trading partner's intellectual library and the customer's demanding knowledge base"

functionality while keeping complexity growth at bay.

Business Transactions

Web services orchestration provides a compensation mechanism to undo certain steps in the transaction due to failures of interdependent activities.

Distributed Administration

Web services orchestration can provide a platform to handle multiple phases of purchase-order change. It's very common that purchase orders need to be changed partially or fully. An asynchronous platform that allows this flexibility so suppliers can consolidate the purchase order state is in great demand in the business process. Web services orchestration allows monitoring of purchase order status by all parties, such as suppliers, buyers, and distributors.

Business Visibility

Web services orchestration provides up-to-date supplier performance metrics, strategic outsourcing benchmarks, real-time customer satisfaction on the purchase order fulfillment rate, and elimination of errors and reconciliation costs.

Version Control

Web services orchestration can provide correlated version control on catalogs, contracts, and pricing promotions associated with the purchase order system. In addition, the system needs to allow for graceful upgrading of functionality and ensuring consistent behavior for users running the old version as well as those running the new version.

Audit Trailing

Web services orchestration can provide financial auditors with a complete history of the purchase order, including negotiation, request to purchase, and purchase order changes. It's also a digital backup for legal records of trades.

Conclusion

Web services orchestration is a core technology integration infrastructure for collaborative commerce. It gives c-commerce a chance to redefine every process of the business cycle: design, plan, source, execute, and fulfill. With Web services orchestration as an engine of integration, we can finally realize the value of c-commerce.

Web Services Journal

FOCUS ON Standards

written by Klaus-Dieter Naujok

ebXML: The Missing Ingredient for Web Services?

Neutral technology can help Web services attain interoperability

Web services has the potential to transform e-business into a plug-and-play affair. Not only will Web services simplify how businesses interconnect, they will also enable businesses to find each other.

As chief scientific officer for IONA Technologies, Klaus-Dieter Naujok is responsible for the company's overall standardization efforts and for technical leadership of its B2Bi direction. He directs IONA's involvement in OASIS and its inclusion of ebXML technology within the Orbix E2A Web Services Integration Platform.
KLAUS-DIETER.NAUJOK@IONA.COM

One reason for the increased interest in Web services is the promise of interoperability. However, complex standards are needed to achieve true interoperability, not only at the messaging and transport layer, but also at the business (application) layer. The success of Web services will depend on how easily businesses are able to engage interoperability at all levels.

There have been many efforts at standardizing Web services, but none of them provides the required features for e-business transactions. Web services standards only address the infrastructure side, but ebXML can provide the standards for interoperability at the business layer, making it the standard solution for Web business services. In other words, ebXML is the missing ingredient for Web services.

ebXML: What Is It?

In the summer of 1999 the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and the international technology business consortium called Organization for the Advancement of Structured Information Standards (OASIS) joined forces to produce a global XML framework for electronic business, called ebXML. At the start more than 120 companies and standards bodies signed up for the "ebXML Initiative." Over the next 18 months more than 2,500 participants across the globe worked on the development of several interrelated specifications. At the final ebXML Phase One meeting in May 2001, they ratified the first generation of ebXML specifications.

The reason for the successful creation and approval of the ebXML infrastructure specifications was that nothing new was invented. The project teams evaluated proven technology to be used as the baseline for all specifications. The editing teams leveraged as much existing technology as possible, including the World Wide Web Consortium's XML Schema, XML Linking Language, and the XML Signature Syntax and Processing specification. In addition, several references from the Internet Engineering Task Force's Request for Comments were considered. New initiatives that were launched well after the ebXML project started were carefully examined, including Security Services Markup Language (SSML), Simple Object Access Protocol (SOAP), and Universal Description, Discovery, and Integration (UDDI). SOAP was successfully incorporated into the ebXML Message Service Specification.

The ebXML infrastructure as documented in the suite of approved specifications provides the only open, out-of-the-box, standards-based solution ready for use. So what sets ebXML's solution apart from the rest? Many commercial solutions are available, however, none simultaneously supports all the business verticals. Enterprises electing to use one of these commercial solutions may not be able to participate in a truly global business environment.

> "Without this functionality, Web services won't be used for industrial-strength and mission-critical applications"

ebXML Infrastructure Elements

The ebXML framework comprises five major work areas, three of which are infrastructure support services that facilitate registration of a business entity, discovery of business partners, configuration of business partner trading agreements, and exchange of business information.

The Registry/Repository component supports all of these functions, making it a major piece of the infrastructure. The ebXML registry provides a set of distributed services that enables the sharing of information. This simplifies business process integration between business parties. The Registry provides the interfacing access services by means of the registry information model and reference system implementation, while a repository provides the physical back-end information storage. For example, an ebXML registry may retrieve configuration information for a particular business entity from the Repository in response to a query, or the repository may contain document type definitions or schemas that are retrieved by a registry query.

Collaborative Protocol Profiles (CPP) are another key element of the infrastructure. The CPP specification is based on the Trading Partner Agreement Markup Language (tpaML) work begun by IBM. The IBM work was enhanced by the efforts of the ebXML Trading Partner Agreement project team to produce a method for defining the transport, messaging, and security environment of a specific business entity. Also, the team defined methods for dynamically exchanging the CPP data between business entities and negotiating message-exchange agreements between the parties. These profiles may be maintained by the individual business entities within the ebXML business domain or may be stored within an ebXML repository.

Information packaging and transport mechanisms, specified in the ebXML Message Service Specification, are the third critical component of the ebXML infrastructure. A protocol-neutral method for exchanging electronic business messages is defined in this specification. Enveloping constructs are specified that support reliable, secure delivery of business information. These flexible enveloping techniques permit ebXML-compliant messages to contain payloads of any format type. This versatility ensures that legacy systems using traditional syntaxes (i.e., United Nations Electronic Data Interchange for Administration, Commerce, and Transport (UN/EDIFACT); ANSI X12; or Health Level 7 (HL7)) can leverage the advantages of the ebXML infrastructure along with users of emerging technologies. Interestingly, both IBM and Microsoft were instrumental in persuading ebXML to adopt SOAP as the foundation for its message services.

The other two areas of work contributing to the ebXML framework are Business Process and Information Modeling (BPIM) and Core Components; both relate to the content and context area. The business process team defined a metamodel that described business transaction patterns used to achieve business goals. In simple terms, these atomic business processes prescribe the detailed interaction of business documents and business signals among parties, called *choreographies*. ebMXL processes define activities such as "order goods" or "deliver goods," as compared with traditional EDI documents that are electronic versions of paper documents such as purchase order or ship notice.

Core components are reusable data elements found in business documents. They're semantically neutral objects; their actual meaning in business documents depends on their context, provided by the business domain and industry in which they are applied. Core components can be single elements or aggregates, defined as natural collections of core elements. A telephone number, for example, may contain a country code, city or area code, and number that when strung together constitute an aggregate. Core components provide the means for industries to continue using their own terminology in business documents, and at the same time relate their terminology to common business processes and neutral identifiers provided by ebXML. As long as trading partners can relate their own terminology to neutral ebXML core components, businesses have a basis for achieving interoperability.

> "The ebXML infrastructure as documented in the suite of approved specifications provides the only open, out-of-the-box, standards-based solution ready for use"

ebXML: Phase Two

After the completion of the infrastructure specification, UN/CEFACT and OASIS signed an agreement to continue the ebXML work by assigning the infrastructure support services to OASIS and placing the busi-

ness components within UN/CEFACT. This enables each organization to advance the work within its own area of expertise, ensuring continuing rapid progress.

UN/CEFACT is continuing the work started under ebXML Phase One, that is, to advance ebXML development as related to business processes, core components, and e-business architecture.

UN/CEFACT has adopted a business process and information modeling methodology, referred to as the UN/CEFACT Modeling Methodology (UMM), that provides the framework under which the ebXML project teams concurrently develop technical specifications that fit seamlessly together with sufficient detail for ebXML-conformant implementation.

"As with any standard, success is measured by its acceptance and implementations based upon it. We're seeing both"

An ebXML business process and information model draws from reusable components such as

- Common business process models as provided for in reference libraries (imported from various levels of business process models, i.e., transactions, collaborations, processes)
- Simple "best-in-class" business collaboration patterns
- Pieces of collaboration patterns, e.g., patterns of how commitment categories are specified, resources are described, etc.,
- Business transaction patterns as established in the UMM
- Business entities, defined as business information objects having a life cycle that transitions through defined states in the fulfillment of the commitments in a collaboration
- Core components specialized for business context.

The business process and information modeling and core components work carried over from ebXML Phase One has come to fruition in Phase Two with the benefit of much iteration of revisions and comments. The modeling information required to enter and determine successful execution of a business collaboration or transaction, i.e., states of business entities, is benefiting from the CC library as a reference for conceptual information entities. Business entities are then elaborated in the model of "on-the-wire" business documents as normalized business information entities.

The UN/CEFACT e-business Architecture provides the umbrella specification that covers the work of all the UN/CEFACT e-business projects. As such, it elaborates on the Phase Two ebXML projects and shows how they relate to the other e-business activity in UN/CEFACT.

Future of ebXML: What's Required for Its Success?

As with any standard, success is measured by its acceptance and implementations based upon it. We're seeing both. To succeed we must complete the work and not get sidetracked by other activities that may seem like they are competing. Remember also that on the content and context side (UN/CEFACT's ebXML responsibility) there is no dependency on technology. In other words, it could be called ebWS (e-business Web services) since the business process and information specifications can sit on top of any infrastructure, not just ebXML's messaging service. This technology and protocol-neutral work will survive many different infrastructure solutions. However, for it to be accepted as such, implementers must have the vision and patience to complete the work. Rushing it will lead to failure since it takes extra time to develop a framework with future-proof details, i.e., not tied to any specific technology. The past has shown that it's simpler and quicker to develop proprietary solutions based on a single technology. However, as technology changes, technology-specific specifications become obsolete and implementers are forced to migrate to new solutions. UN/CEFACT's content and context work (BPIM and CC) promises that businesses won't have to translate their interactions (collaboration) as the infrastructure solutions progress over time.

ebXML and Web Services Standardization

Many deficiencies must be addressed in order to standardize Web services. From the ebXML perspective, Web services do not have the features required for e-business transactions; its standards only address the transport layer. However, ebXML provides the standards for interoperability at the business layer, making it the standard solution for Web business services.

As pointed out, there are some business-critical functionalities (collaboration, security, etc.) missing from the current narrow definition of Web services (SOAP, WSDL). Without this functionality, Web services won't be used for industrial-strength and mission-critical applications. Efforts are currently under way in organizations, consortiums, and standards bodies to fill these gaps with specifications that use the same language from and are aligned with the current Web services definition. Most of those efforts compete directly with various components of ebXML. It is perplexing that some organizations start new efforts such as WS-I immediately after having contributed to a similar effort in ebXML Phase One. WS-I's founding companies were key players in that phase, but instead of acknowledging that ebXML solved most of the problems, they're trying to readdress them, ignoring their own prior work.

"So what sets ebXML's solution apart from the rest? Many commercial solutions are available; however, none simultaneously supports all the business verticals"

Since the ebXML framework is a neutral technology, why not provide mappings from the ebXML specifications to Web services technologies? Why not use the ebXML content and context work as a standard way to define the business semantics in Web services to achieve interoperability at all layers?

The ebXML framework is the missing ingredient for Web services. The ebXML development effort could fail to seize this collaborative opportunity, and Web services vendors could "standardize" on one of their preferred proprietary solutions. However, if ebXML does grab this opportunity, its vision for a global economic exchange standard will be brightened indeed. Current Web services are just a stopgap measure. However, if the ebXML work can engage Web services initiatives in a tactical alliance, the broader UN/CEFACT objective of Web resources as business entities engaged in collaborations will be realized. Remember, the XML implementation aspect of ebXML is just the next generation of EDI. The following generation (within five years?) will likely be different from both the current Web services and ebXML.

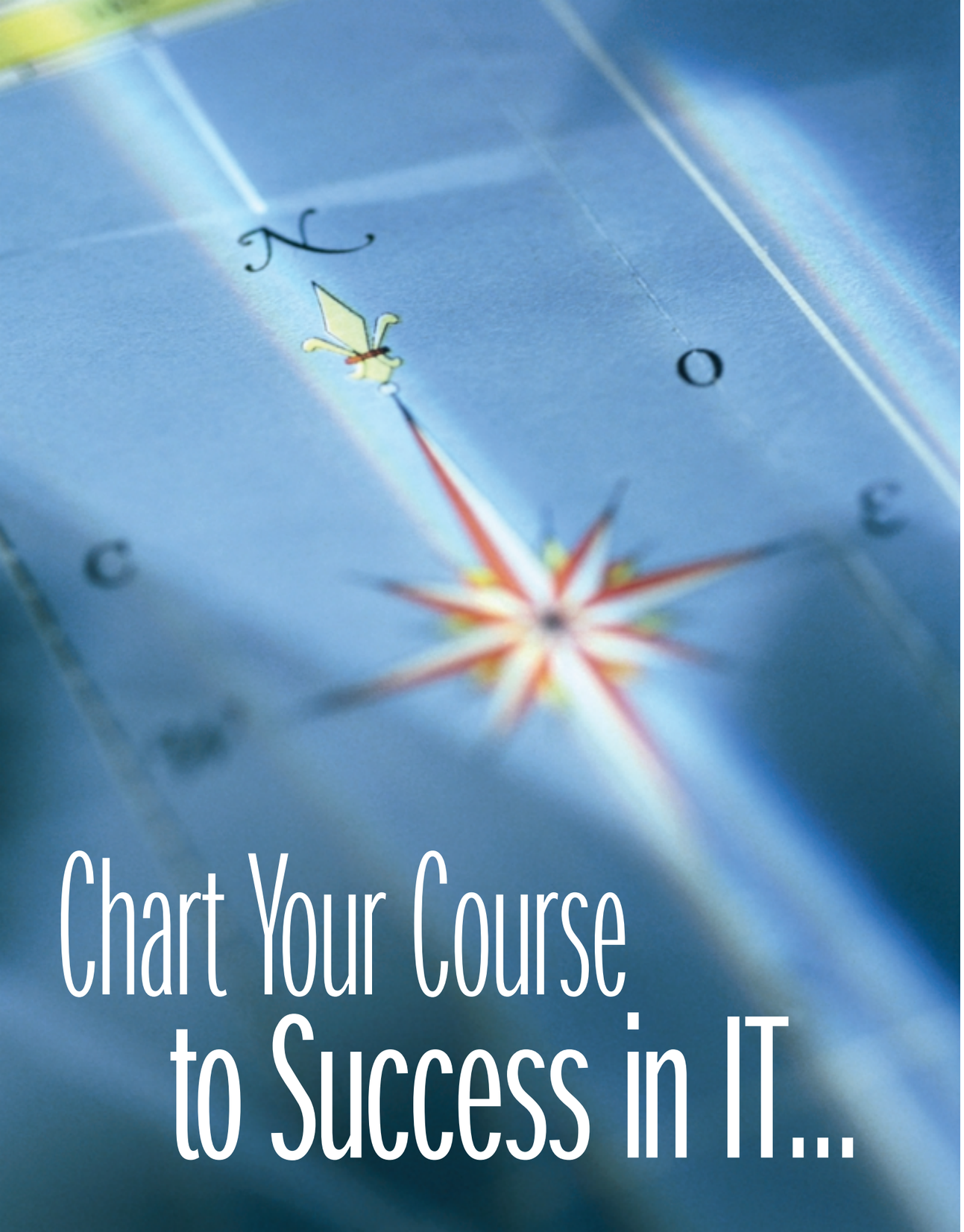


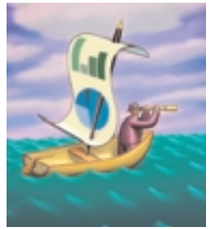
Chart Your Course to Success in IT...

...order your copy of

Java Trends: 2003

Available July 1*

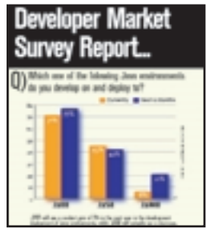
Don't go astray. In the vast sea of Internet technology, market conditions change constantly. Will Java remain the hot platform it is today? Will C# rapidly gain ground? What are the world's foremost Java developers aiming their sights toward? Which companies come to mind when planning their next project? How can their thinking direct your efforts?



EnginData Research has studied the IT industry extensively, spotting many key trends and alerting industry leaders along the way. In the first quarter of 2002, they embarked on their most challenging mission ever: the most in-depth study ever done of the Java development marketplace.

After months of analyzing and cross-referencing more than 10,500 comprehensive questionnaires, the results are now available.

EnginData's research is invaluable to leading IT vendors, integrators, Fortune 500 companies, trade-show organizers, publishers, and e-business organizations worldwide.



Here's just a sample of the critical data points the Java report delivers...

- ✔ Current and projected usage of languages and platforms
- ✔ Multiple rankings of hundreds of software vendors and tools
- ✔ Types of applications being developed
- ✔ Databases being used
- ✔ Purchasing patterns
- ✔ Company size
- ✔ Development and purchasing timelines
- ✔ Perceptions and usage of Web services
- ✔ Preferred Java IDE
- ✔ J2EE, J2ME, J2SE usage comparisons

As an IT specialist, **EnginData Research** focuses exclusively on three leading drivers in IT today – Java, Web services, and wireless development. Coming soon, our Web services survey will deliver such critical knowledge as:

- ✔ Time frame for developing Web services – enabled apps
- ✔ Percentage of apps with Web services today
- ✔ Sourcing and hosting Web services
- ✔ Perceived leaders in Web services tools and solutions

Navigate your company through the challenging IT marketplace with the trusted knowledge and intelligence of **EnginData Research**. Order your copy of the 2002–2003 Java market study by calling Margie Downs at 201-802-3082, or visiting our Web site.

engindata RESEARCH

www.engindata.com

*A limited number of preview copies will be available at our booth (#624) at JDJEdge International Java Developer's Conference & Expo, June 24–27, at the Jacob Javits Convention Center in New York.

Written by Paul Fremantle, Donald F. Ferguson, Heather Kreger, and Sanjiva Weerawarana

UNDERSTANDING THE WEB SERVICES VISION

Delivering on a new service-oriented model of computing

AUTHOR BIOS:

Paul Fremantle is a Web services architect in IBM's Hursley Laboratory. Paul has been working with Java, XML, and Web technologies since 1994, and is one of the architects of IBM's Web Services Invocation Framework and the Web Services Gateway. Paul is the coauthor of The XML Files, *an IBM Redbook, as well as technical papers on WebSphere. He has an MA in mathematics and philosophy from Oxford University and an MSc in computation.*

Donald F. Ferguson is an IBM Fellow, member of the IBM Academy of Technology, chief architect of the WebSphere Platform, and chairman of the AIM Architecture Board. He holds a PhD in computer science from Columbia University.

Heather Kreger is the lead architect for Web Services in Emerging Technologies. She is responsible for identifying new Web services trends as well as helping Web services integration and adoption across IBM and is the colead for JSR109, which specifies Web services support in J2EE environments.

Sanjiva Weerawarana is a research staff member in the Component Systems Group at IBM's TJ Watson Research Center. He is one of the coauthors of the WSDL and WSFL specifications, and a codeveloper of Apache SOAP, WSTK, WSDL Toolkit, WSIF, and WSGW. He holds a PhD in computer science from Purdue University.

Web services is the latest trend in distributed computing. Based on sending XML messages that are transported over the HTTP protocol, the initial work has created a distributed computing model that is simple, easy, and lightweight. Most importantly, it works over the Internet.

- ***Simple Object Access Protocol (SOAP)*** has brought interoperability between disparate systems over the Internet, providing distributed computing based on open Web standards. This complements existing approaches to interoperability, for example CORBA or messaging standards.
- ***Web Services Description Language (WSDL)*** provides a framework for describing Web services based on XML and XML Schema. WSDL allows different systems and tools to understand and communicate with services provided by other technologies and organizations.

Two discovery technologies allow Web services to be discovered on the Internet or on a network:

- ***Web Services Inspection Language (WSIL)*** allows a server to publish the set of services available for a particular domain.
- ***Universal Description, Discovery, and Integration (UDDI)*** allows businesses to publish information about their business and the services it offers, and others to search the directory for businesses and services. Organizations can also use UDDI within their enterprise.

Overview

Web Services Motivation

The Web today is fundamentally designed around human-to-machine interactions, with little semantic description. Now there is a requirement for a Web-based technology that supports application-to-application communication. This is driven by a number of needs, including:

- ***The "Portal" model:*** Many companies have developed highly customized Web sites where users can access the majority of information, content, and function that they require from a single consolidated Web page. This model requires Web sites to communicate with each other.
- ***The growth of electronic business-to-business transactions (B2B):*** Many standards exist for B2B, including EDI, RosettaNet, cXML, and others. These traditionally solve well-defined, systematic approaches to B2B. The growth of the Internet has put pressure on the technology to provide lightweight, inexpensive Web-based communications standards for B2B, with special focus on dynamic discovery and self-description of services.
- ***Enterprise Application Integration (EAI):*** Many companies have put in place integration of applications across a wide variety of technology. In order to create efficient and effective channels to market, it is often necessary to connect together systems as diverse as mainframe systems, customer relationship systems, databases, accounting systems, and Web application servers. The push to e-business has meant that businesses are exposing their processes and systems directly to customers over the Web.

Benefits

We believe Web services offers a number of core benefits that allow application-to-application communication over the Web or intranet.

XML and Open Web Standards:

XML and HTTP are important because they offer:

- A technology-neutral approach
- Universal availability
- An appropriate document model for loosely coupled systems
- A vendor-neutral approach
- Support for existing Web infrastructure like firewalls.

Services-Oriented Architecture

Web services focuses on describing the interactions between parties rather than the provider of the service. In particular this is important for distributed heterogeneous communications, where the implementations may vary significantly, but the interface should be constant.

Description and Discovery

The combination of WSDL and UDDI allows someone to search and locate a service, and create a client to that service – without requiring access to code repositories, common technologies, or even the same tools as the service provider. This organizational loose coupling has been a major driver to the growth of the Web. For example, many Web sites benefit from links, both to and from their Web sites, which did not involve any organizational involvement between the Web site owners.

Messaging and Request-Response Models

Typically, there has been a split in the technologies between tightly coupled request-response (RPC) behavior, and loosely coupled messaging systems. Both SOAP and WSDL accommodate both models in a single set of standards, making them suitable for both client-server and integration (B2B and EAI). This unifies the programming model for cross-Internet integration as well as supporting existing intra-enterprise optimizations.

In this article we present a wider view of Web services as the basis of a services-oriented architecture (SOA). In particular, we show how Web services can be used in the portal, B2B, and EAI models, and how the notion of description and discovery can be extended to provide a rich model for interaction across multiple technologies.

Building and Using Web Services

The Web services standards only describe the outside interface of a service. Those interfaces imply some internal structure and semantics but the requirement to implement a Web service is very minimal. A number of technologies can be used – for example, simple Java classes; Enterprise JavaBeans; scripts such as VBScript, Python, or JavaScript; or C/C++ programs or objects can all be the implementation of a Web service. In fact, many other technologies, such as relational database stored procedures or legacy

transactions can also implement Web services. There are three common models.

Bottom-up Development

An existing application function, such as an Enterprise JavaBean component or Java class, is exposed as a Web service. A tool examines the structure of the component, and generates the Web services description (WSDL file), which may be published using WSIL or UDDI. Often the tool will also generate deployment information, which configures a server to support and provide that service – to configure the SOAP support and other protocol enablement, and to "route" incoming requests to the underlying original component. The component does not need to understand XML, SOAP, or other formats, as the server transforms the SOAP request into the invocation mechanism of the original component.

Tools which support this include the IBM WebSphere Studio Application Developer, IBM Web Services Tool Kit, and the Axis Java2-WSDL. Once the component is deployed, users can download the published WSDL description and use any WSDL-aware tool to create a stub or proxy object. That object gives them an interface in the language they are programming in, such as a Java class or .NET business object, which can be invoked and will generate the SOAP message across the network.

Meet-in-the-Middle

In this model, the service definition exists already. For example, a parcel delivery firm wishes to implement a Web service that allows tracking of parcels. There is an existing common interface, which is published in UDDI as a tModel. The developer of the Web service downloads the WSDL from the Internet, and uses a tool to create a *skeleton*. The skeleton is a component that has the structure but no business logic. The developer then creates the business logic that accesses the existing parcel tracking system. The new component calls the existing business logic, but exposes it with the common Web service interface. The service is published in UDDI with the same tModel as other parcel tracking systems. Existing parcel tracking clients, written to that tModel, are able, to invoke the new service.

There is a third model – top-down – which we will address later. This model is similar to meet-in-the-middle, except that new business logic is necessary to implement the service.

Implementing Web Services in a Java Environment

Recently, there has been a great deal of work on Java standards for Web services in the Java Community Process, creating standards for using, developing, and deploying Web services in the Java and J2EE environments.

3.1 JAX-RPC

The Java API for XML-RPC (JAX-RPC) is a standard which defines to invoke and interact with Web services. JAX-RPC allows a user to access a stub or proxy for the service, or to use a Call interface that offers more direct access to the underlying SOAP invocation. JAX-RPC also defines a standard mapping from WSDL to Java interfaces, allowing tools to generate standardized interfaces for accessing Web services; in addition, the specification defines a standard mapping from Java interfaces to WSDL descriptions. At the time of writing, the JAX-RPC specification was about to become final. The Apache Axis project is currently building an implementation of JAX-RPC.

JSR109 – Implementing Enterprise Web Services

JSR109 is the ongoing standards activity on how to implement Web services in a J2EE environment. Included in the standard are:

- How to implement Web services in a J2EE environment consistent with the J2EE programming model
- How to deploy Web services implementations in a J2EE environment
- How to locate Web services from within a J2EE environment
- How to use Web services as a client consistent with the J2EE programming model
- How to protect and secure Web services using J2EE security technology

JSR109 adds Web Services deployment descriptors to support deployment of Web services. It builds upon the view of Web services provided by JAX-RPC and supports the deployment of JAX-RPC clients and service implementations. At the time of this writing, JSR109 was about to enter Public Review.

Other Standards

There are a number of other Java specifications and open-source contributions available or being developed. Some of the more significant ones for Web services are:

- ***JWSDL:*** A simple, extensible API for reading, writing, and creating WSDL documents, backed by an open-source implementation.
- ***JAX-B (XML Binding):*** Defines how to translate from Java classes to XML schema and back again.
- ***JAX-R (XML Registries):*** Defines an API for accessing service registries including UDDI and ebXML Registry and Repository.
- ***J2ME Web Services Specification:*** For use of Web services on pervasive devices; this is just getting started.
- ***XML Digital Signature, XML Digital Encryption, and XML Key Management:*** JSRs that are developing Java bindings for their respective standards in the W3C's SOAP-SEC standard.
- ***UDDI4J:*** An open-source API used to access UDDI directories.

So far we've seen how the base Web-services standards fit together, and how they are implemented in a real environment. Now let's take a look at how we can use this technology, both in a B2B environment and in an EAI environment.

Using Web Services in a B2B Context

There are a number of initiatives for doing business-to-business electronic commerce. In particular, Electronic Data Interchange (EDI) has been very successful amongst large companies improving their supply-chain management. RosettaNet is another standard that has had success – especially in the electronic component marketplace. However, both of these initiatives, while successful in their own niches, have very limited success relative to the evolution of the Web and broad, cross-industry acceptance of HTTP, XML, etc. The result is that many companies are looking to leverage Web standards to build B2B solutions for e-commerce and e-business. We see the use of Web services developing very rapidly in B2B.

Challenges

There are two issues around this area.

- Security and management
- Process and conversation management

One approach to addressing security and management is the use of a gateway. A gateway is a control point for external interactions. The gateway fills a role for B2B similar to what firewalls provide for more traditional uses of HTTP. For example, the gateway can provide an extra level of security and authorization, and logging of both incoming and outgoing service interactions. In addition, a Web services gateway can provide a layer of indirection and protocol switching, which adds security and allows enterprises to use existing protocols and still provide Internet protocols for external users. IBM has made available a Web Services Gateway Preview.

The second issue with using Web services in a B2B scenario is that currently Web services are strictly "one-shot" interactions, whereas typical B2B interactions are ongoing, bi-directional, long-running, and correlated. However, there have been some proposals on how to manage a set of interactions as part of a process. In particular, XLANG from Microsoft and the Web Services Flow Language from IBM both address this issue. It is likely that a common cross-industry standard will emerge which will allow standardization of the processes between companies using Web services.

Using Web Services in an EAI Context

A number of vendors and companies have seen Web services as an approach to support Enterprise Application Integration. Many first Web services projects and pilots have been based on connecting Microsoft COM systems to Java and CORBA systems: for example, using J2EE as their application server and using COM-based Windows tools to build the clients. SOAP and WSDL allow them to create simple proxy objects on the client, which can invoke services running on the server in an effective manner.

This approach also ties into the work many organizations began in 2000 and 2001 to standardize their internal integration on XML languages. Many companies built integration on top of industry-standard XML grammars, and this architecture fits very readily into the Web services paradigm. SOAP and WSDL support *document*-style interaction: in this model, the SOAP envelope is used to transmit an XML document that is defined by an existing XML Schema, allowing SOAP to be used as a messaging format for existing XML languages.

A number of packaged application vendors have identified this as an approach to open up their systems to simple lightweight access, and this is likely to increase the use of Web services technology as a basis of EAI.

There have traditionally been two very different approaches to integration. One approach is generally based on asynchronous one-way messaging, which is used to integrate widely divergent systems across large geographies and different technologies. This approach is typically based on *document* interchange. The other ap-proach is based on tightly coupled synchronous integration, for example using remote procedure calls or CORBA and its description language (IDL) to call objects directly. This is typically used in single locations, with high-bandwidth, highly available systems, and is typically based on strictly typed interactions. Both SOAP and WSDL include both of these concepts: RPC-based interactions using typed interfaces and document-based one-way messaging. This gives Web services a very powerful approach to subsume and support both approaches.

Challenges

Web services need to meet the existing level of expectation in EAI technologies. There are a number of high-quality (and high-cost!) EAI technologies that offer essential facilities such as transformation, adaption, process management, reliability, high performance, monitoring, and metadata repositories. Web services offers a different set of benefits: low cost of entry, open standards based, and truly heterogeneous.

Moving Beyond SOAP to a WSDL-Centric View

WSDL forces a true separation of interface and implementation. This allows a separation of the business logic and the infrastructure – and is probably the most important aspect of the SOA. In general, application developers should concentrate on the abstract interface defined by the interface, and the deployers and administrator should concentrate on the infrastructure.

As stated earlier, SOAP is the key component for building systems that can interoperate across the Web, but the real benefit of Web services is for building distributed systems where the service implementation is unimportant to the user of the service. What is important are the interface and its description. This motivates us to move our attention from SOAP to WSDL, and in particular to developing our models based on the interface or *PortType* of a service.

There are a number of approaches and technologies that are being developed that take this approach.

Top-down Development

Earlier we described *bottom-up* and *meet-in-the-middle* development. The other approach is to do *top-down* development. In this model, the business analyst builds a UML model or business process. This is an abstract description of the interaction or the process. From this the WSDL PortType can be generated. Outside of Web services, this model is used very successfully as it tends to produce systems that closely match the requirements and the business model. From a Web services viewpoint it also works well because it fits the Web services model, and tooling exists to then create implementation objects from the WSDL definitions. This scenario describes an approach known as Model-Driven Architecture.

Business Process Management

Once sets of services are available via SOAP and described and published, it becomes very desirable to be able to build new processes using those services. Typically, programmers in the past have used programming languages to code the flows of a process, in which it is very easy to capture the "success case" – the normal handling of a particular process. However, often the fault cases are complex. In particular, if the individual activities in the process are distrib-

uted and loosely coupled – for example provided by another organization across the Web – then we need to have smart compensation logic in order to handle the fault cases, because traditional two-phase commit protocols are not appropriate in loosely coupled environments.

A different model is to describe the process using a business process description language. These "flow" languages capture the relationship between services and the steps in the business process. An engine then runs this process, and if faults occur, the engine takes appropriate compensation action. Work is ongoing to develop a standard Web services language for describing business processes, where the process is described completely in terms of the interfaces/PortTypes of the individual services, and the bindings are defined at deployment time. Building business processes becomes a matter of composing existing services.

Web Services Invocation Framework

Earlier, we described how WSDL was inherently extensible, and independent of implementation type. Adding *extensibility elements* to WSDL allows the creation of descriptions of other service implementations than SOAP. Typically, Web services are implemented using existing application components – such as Java classes, Enterprise JavaBeans, or COM objects. These components are also sometimes *wrappers* onto legacy systems, such as mainframe transaction systems. We can capture the dependence relationship between these by extending WSDL to describe the existing component models, the exposed SOAP-available service, and the underlying component. In effect, we are adding the description capability of the SOA to the existing component model.

WSDL is not just a description layer – it has a real implementation in tools that can use WSDL descriptions to generate stubs that can access the services. So if we add descriptions of non-SOAP systems, we are in danger of losing that benefit. The Web Services Invocation Framework (WSIF) addresses that. Effectively, it is a pluggable framework, that allows *providers* to be plugged in. A *provider* is code that supports a WSDL extension and allows invocation of the service through that particular implementation.

This means that the client code is independent of the implementation. WSIF also allows late binding, where an existing client can use a new implementation at runtime – even over a new protocol or transport. WSIF also allows the infrastructure and runtime to choose the best implementation on the basis of quality-of-service characteristics or business policy.

Provisioning

Provisioning is the ability to host services and charge clients for their use. For the Web services vision to become real, it will be necessary to be able to sell services and to identify who is using them, and how often. Provisioning is not based on the underlying SOAP message, but instead on understanding the range of services and how they fit to contracts. In order for a service to be provisioned, there must be a contract with a given identified user or organization, and then each use of the service must check if there is an applicable contract and then be metered. From the metering, a rating engine can identify the cost under the right contract and generate an invoice for the usage.

The Web Service Hosting Toolkit (WSHT) is a toolkit that offers the ability to provision services, irrespective of how they are implemented and deployed.

Conclusion

In this article we have seen a progression, from the base Web services standards bringing point-to-point interoperability using XML and HTTP, all the way to dynamic invocation and composition of services using higher order description languages. Our vision is of an SOA that:

- Encompasses both messaging and RPC approaches
- Supports discovery and description of services
- Enables both business-to-business communications and enterprise application integration
- Allows composition of services into business processes
- Enables the metering, monitoring, and contracting of services

In our vision of Web services, there is a clear distinction between the business logic process and interfaces and the underlying infrastructure that provides them, through meta-data and XML descriptions such as WSDL and process descriptions.

We recognize that there is still a long way to go in the Web services technology for the greater marketplace – in particular, there are requirements to address and standardize:

- Composition and choreography
- Complex end-to-end security models
- Support for reliability
- Business transaction models
- A framework for business trust

These efforts are underway, but that shouldn't stop the technology from being used today. The production environments exist to allow Web services to be deployed, described, published, and accessed today. Early implementations exist that show how Web services can be exposed to partners, composed, and provisioned. Finally there is a vision and a roadmap that is shared by a number of vendors that are delivering on a new service-oriented model of computing.

References

- *Standards:* www.w3.org and www. uddi.org.
- *Java Standards:* www.jcp.org.
- *CORBA, UML,* and *MDA:* www.omg.org.
- *Web Services and Open Source Portal:* www.ibm.com/developer/W orks.
- *Open Source Web Services:* http://xml.apache.org.
- *IBM WebSphere:* www.ibm.com/we bsphere.
- *IBM Previews:* www.alphaworks.ib m.com.

FREE TUTORIAL
JUNE 27TH WITH WEB SERVICES EDGE 2002 EAST REGISTRATION. LIMITED OFFER!

TO REGISTER: www.sys-con.com or Call 201 802-3069

web services world tour EDGE 2002

$195 REGISTRATION FOR SYS-CON SUBSCRIBERS
BEST EDUCATIONAL VALUE FOR THE HOTTEST TECHNOLOGY SKILLS!

Take Your Career to the Next Level!

SEATING IS LIMITED. REGISTER NOW FOR THE CITY NEAREST YOU! WWW.SYS-CON.COM

Learn How to Create, Test and Deploy Enterprise-Class Web Services Applications

TAUGHT BY THE INNOVATORS AND THOUGHT LEADERS IN WEB SERVICES

EXPERT PRACTITIONERS TAKING AN APPLIED APPROACH WILL PRESENT TOPICS INCLUDING BASIC TECHNOLOGIES SUCH AS SOAP, WSDL, UDDI AND XML, PLUS MORE ADVANCED ISSUES SUCH AS SECURITY, EXPOSING LEGACY SYSTEMS AND REMOTE REFERENCES.

RECEIVE 4 FREE 1-YEAR SUBSCRIPTIONS ($345 VALUE!)

SHARPEN YOUR PROFESSIONAL SKILLS.

KEEP UP WITH THE TECHNOLOGY EVOLUTION!

"Presented an excellent overview of Web services. Great inside knowledge of both the new and old protocols. Great insight into the code piece."
– Rodrigo Frontecilla

"Very articulate on the Web services SOAP topic and well-prepared for many questions. I've learned a lot from this seminar and I appreciate this seminar for my job. Thank you!"
– Kenneth Unpingco, Southern Wine & Spirits of America

"I liked the overview of Web services and the use of specific tools to display ways to distribute Web services. Good for getting up to speed on the concepts."
– B. Ashton, Stopjetlag.com

Echoed over and over by Web Services Edge World Tour Attendees:

"Good balance of theory and demonstration."

"Excellent scope and depth for my background at this time. Use of examples was good."

"It was tailored toward my needs as a novice to SOAP Web services – and they explained everything."

WHO SHOULD ATTEND:

- Architects
- Developers
- Programmers
- IS/IT Managers
- C-Level Executives
- *i*-Technology Professionals

IF YOU MISSED THESE...

BOSTON, MA (Boston Marriott Newton) SOLD OUT!
WASHINGTON, DC (Tysons Corner Marriott) SOLD OUT!
NEW YORK, NY (Doubletree Guest Suites) SOLD OUT!
SAN FRANCISCO, CA (Marriott San Francisco) SOLD OUT! CLASSES ADDED

BE SURE NOT TO MISS THESE...

Each city will be sponsored by a leading Web services company

...COMING TO A CITY NEAR YOU

2002

NEW YORK AT WEBSERVICES EDGE 2002 EAST	JUNE 27
BOSTON	JULY 10
SAN FRANCISCO	AUGUST 6
SEATTLE	AUGUST 27
AUSTIN	SEPTEMBER 10
LOS ANGELES	SEPTEMBER 19
SAN JOSE AT WEBSERVICES EDGE 2002 WEST	OCTOBER 3
CHICAGO	OCTOBER 17
ATLANTA	OCTOBER 29
MINNEAPOLIS	NOVEMBER 7
NEW YORK	NOVEMBER 18
SAN FRANCISCO	DECEMBER 3

2003

CHARLOTTE	JANUARY 7
MIAMI	JANUARY 14
DALLAS	FEBRUARY 4
BALTIMORE	FEBRUARY 20
BOSTON	MARCH 11

REGISTER WITH A COLLEAGUE AND SAVE 15% OFF THE LOWEST REGISTRATION FEE.

TOPICS HAVE INCLUDED:
Developing SOAP Web Services
Architecting J2EE Web Services

On May 13th, the San Francisco tutorial drew a record 601 registrations.

SYS-CON *i*-TECHNOLOGY EDUCATION

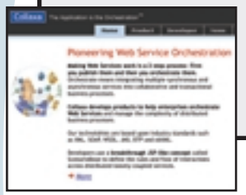
REGISTRATION FOR EACH CITY CLOSES THREE BUSINESS DAYS BEFORE EACH TUTORIAL DATE. DON'T DELAY. SEATING IS LIMITED.
NON-SUBSCRIBERS: REGISTER FOR $245 AND RECEIVE THREE FREE ONE-YEAR SUBSCRIPTIONS TO *WEB SERVICES JOURNAL*, *JAVA DEVELOPER'S JOURNAL*, AND *XML-JOURNAL*, PLUS YOUR CHOICE OF *BEA WEBLOGIC DEVELOPER'S JOURNAL* OR *WEBSPHERE DEVELOPER'S JOURNAL*, A $345 VALUE!

TO REGISTER: www.sys-con.com or Call 201 802-3069

Collaxa Announces Availability of Web Service Orchestration Server 1.0

(Redwood Shores, CA) – Collaxa, Inc., has announced that its Web Service Orchestration Server (WSOS) 1.0 for the BEA WebLogic application server is now available for public download at www.collaxa.com/developer.jsp. WSOS 1.0 for Oracle9*i* Application Server and Database is available for private download. Collaxa's WSOS is the first JavaServer Page (JSP)–like abstraction for orchestrating Java Message Service (JMS) and XML Web services, enabling mainstream Java developers to solve complex business-process integration problems using emerging standards. Over 150 developers have successfully evaluated the solution through the company's preview program since it was unveiled at BEA's eWorld.

www.collaxa.com, www.bea.com



Altova Announces Enhancements in XML Spy 4.4 Suite

(Beverly, MA) – Altova, Inc., has announced the release of its XML Spy 4.4 Suite, a comprehensive product line of tools for advanced XML application development, consisting of the XML Spy 4.4 Integrated Development Environment (IDE), the XML Spy 4.4 XSLT Designer, and the XML Spy 4.4 Document Editor. XML Spy 4.4 Suite is available as a free upgrade to current XML Spy 4.x Suite customers.

A key new feature is DocBook Editing Support. DocBook is a widely adopted XML content model for describing books, articles, and other prose documents such as technical documentation.

www.xmlspy.com

LogiXML Expands Products to Microsoft .NET Platform

(McLean, VA) – LogiXML, Inc., the creator and designer of the XML-based modeling language, LogiXML, has announced that the LogiXML modeling language is now available on the Microsoft .NET platform. LogiXML added this new enhancement to its product suite to enable customers to use and interact with Web services. Microsoft's .NET platform allows XML-based Web services to communicate and share data over the Internet, regardless of operating system, device, or programming language. By moving LogiXML to the .NET platform, the company provides developers with easy access to many .NET features for using XML-based Web services.

www.logixml.com

ScreamingMedia Launches New Version of Web Services Infrastructure

(New York) – ScreamingMedia, Inc., a provider of information, technology and tools for the enterprise, has announced the launch of the latest version of its Web services infrastructure, which will enable the delivery of more flexible, dynamic, and modular content and applications to portals.

ScreamingMedia also announced it is joining the OASIS Web Services for Remote Portals (WSRP) Technical Committee, a group of portal industry leaders charged with creating a standard that allows for the Plug and Play of content and applications from disparate sources into portals and other intermediary Web applications.

www.screamingmedia.com

FundsXpress, Epicentric Announce Deployment of 350 Financial Portals

(San Francisco) – Epicentric, a provider of business portal solutions, has announced that FundsXpress Financial Network, Inc., is delivering 350 Epicentric-powered e-banking portals to the financial services small- and mid-size markets through its Portana Dynamic Web Site offering.

The Portana Dynamic Web Site is a hosted solution that delivers highly customized portals to financial institutions and their account holders in as little as seven days. Targeted at financial institutions of all sizes, the

Portana Dynamic Web Site enables banks and credit unions with modest technical resources to enhance relationships with their account holders online. An option exists to incorporate other FundsXpress services, including Internet banking, cash management, lending, and account aggregation.

www.fundsxpress.com, www.epicentric.com.

Versant's New Release Enhances Support for J2EE

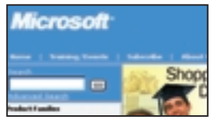
(Freemont, CA) – Versant Corporation, a provider of middleware infrastructure technology, has announced the commercial availability of Versant enJin 2.3. Versant's latest release enhances support for Java 2 Enterprise Edition (J2EE) standards, making it easier for customers to develop and deploy J2EE applications using Versant enJin with any J2EE-compliant application server.

enJin is focused on the growing segment of the J2EE applications market that uses complex object models to address business needs. Using enJin with J2EE-compliant application servers makes it feasible to implement sophisticated business logic supported by a rich domain model without the time-consuming task of object-to-relational mapping.

www.versant.com

Microsoft's "TrustBridge" to Enable Businesses to Share User Identities

(Redmond, WA) – Taking the next step toward a more connected and secure Web services environment, Microsoft Corp. has announced a new Windows technology, codenamed "TrustBridge," that will enable businesses to share user identity information between applications and organizations. "TrustBridge" technology will allow different organizations using the Windows operating system to exchange user identities and interoperate using industry-standard XML Web services protocols including Kerberos and WS-Security. Microsoft also delivered a product roadmap, the Microsoft Federated Security and Identity Roadmap, for federated security and identity management across the Microsoft product line.



www.microsoft.com

Sun Unveils Sun ONE Portal Server 6

(Santa Clara, CA) – Sun Microsystems, Inc., a solutions provider, introduced its Sun ONE Portal Server 6, the industry's first portal server solution to include fully integrated, secure identity-management capabilities to streamline the security and management of enterprise portals with a single management console. The company also unveiled its product roadmap to support several J2EE application servers: BEA WebLogic, IBM Websphere, and Sun ONE.

The new Sun ONE Portal Server provides fully integrated, secure identity management capabilities; Web single sign-on; policy and access control; service provisioning; and unified user management – via the embedded Sun ONE Identity Server.

www.sun.com

Quintessence and Cape Clear Partner to Migrate Oracle Applications

(Berkshire, UK) – Quintessence Systems and Cape Clear Software have announced they will work together to enable customers to migrate existing Oracle applications and connect them to Web services both inside the corporate network and across the Internet. Customers can use Quintessence's in2j tool to automatically migrate their Oracle legacy code into

Java. Once in Java, Cape Clear's CapeConnect platform can be used to expose the migrated Java code as Web services.

www.in2j.com, www.capeclear.com

WSJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Actional	www.actional.com/soap2	800-808-2271	9
ADOS Co., Ltd.	www.a-dos.com	81-3-5475-1551	15
Altaworks	www.altaworks.com	603-598-2582	31
Altova	www.xmlspy.com		21
CFAdvisor.com	www.cfadvisor.com	888-303-5282	
ColdFusion Developer's Journal	www.sys-con.com/coldfusion	888-303-5282	57
Covasoft	www.covasoft.com		35
engindata Research	www.engindata.com	201-802-3082	52-53
JDJ Store	www.jdjstore.com	800-303-JAVA	47
Power Builder Developer's Journal	www.sys-con.com/pbdj	800-303-5282	58
Richard Hale Shaw Group	www.richardhaleshawgroup.com		27
Sams Publishing	www.samspublishing.com		25
Sitraka	www.sitraka.com/jclass/ws		13
Sonic Software	www.sonicsoftware.com/websj		4
SpiritSoft	www.spiritsoft.com/climber		17
SYS-CON Media	www.sys-con.com	888-303-5282	37, 59
Web Services Edge Conference & Expo	www.sys-con.com	201-802-3069	60-61
Web Services Journal	www.sys-con.com	888-303-5282	41
Web Services Resource CD	www.jdjstore.com	800-303-JAVA	64-65
WebLogic Developer's Journal	www.sys-con.com	888-303-5282	51
webMethods	www.webmethods.com/ews		19
WebSphere Developer's Journal	www.sys-con.com	888-303-5282	37

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.

WebServices JOURNAL .NET J2EE XML

XML JOURNAL

THE FIRST & ONLY WEB SERVICES RESOURCE CD!

WEB SERVICES RESOURCE CD

THE SECRETS OF THE WEB SERVICES MASTERS

INCLUDES EXCLUSIVE .NET ARTICLES

MORE THAN 400 EXCLUSIVE WEB SERVICES & XML ARTICLES

EDITED BY SEAN RHODY

EVERY ISSUE OF WSJ & XML-J EVER PUBLISHED

THE MOST COMPLETE LIBRARY OF EXCLUSIVE WSJ & XML-J ARTICLES ON ONE CD!

"The Secrets of the Web Services Masters"

CD is edited by well-known WSJ Editor-in-Chief Sean Rhody and organized into more than 40 chapters containing more than 400 exclusive WSJ & XML-J articles.

Easy-to-navigate HTML format!

Bonus:

Full .PDF versions of every WSJ & XML-J published since the first issue

- XML in Transit
- XML B2B
- Java & XML
- The XML Files
- XML & WML
- Voice XML
- SYS-CON Radio
- XML & XSLT
- XML & XSL
- XML & XHTML
- 2B or Not 2B
- XML Script
- XML Industry Insider
- <e-BizML>
- XML & Business
- XML Demystified
- XML & E-Commerce
- XML Middleware
- XML Modeling
- CORBA & XML
- Data Transition
- XML @ Work
- XML & Databases
- Electronic Data Interchange
- Ubiquitous Computing
- Information Management
- Objects & XML
- XML Pros & Cons
- Java Servlets
- XML Filter
- UML Integration
- WSDL
- Beginning Web Services
- Web Services Tips & Techniques
- Frameworks
- Management
- Security
- UDDI
- .NET

3 YEARS 25 ISSUES 400 ARTICLES ONE CD

Special Limited Time Price

Now Shipping

$79 + S&H

ONLINE ORDER AT JDJSTORE.COM SAVE $40

www.JDJSTORE.com

OFFER EXPIRES JUNE 30, 2002

Beyond Web Services

The next critical step

Chris Weaver

Chris Weaver is VP of technology for Paros Software, Inc. He has created industry-leading software products in the collaborative commerce market for Polycom and VTEL and as a consultant for clients around the world. His research interest at Paros Software is real-time business process collaboration as applied to Web services software.
CHRIS@PAROSSOFTWARE.COM

One of the greatest challenges faced by Web services developers is the fact that the term "Web services" isn't well understood outside the developer community. We've all experienced requests from product managers, customers, or salespeople who had no idea that connecting two disparate systems takes more than a few lines of code. The situation will only get worse with the increased complexity of business-process, transaction, and workflow management.

The use of workflow standards to integrate Web services into a comprehensive, enterprise-wide system is the logical innovation that follows Web services. The challenge is that process management standards must catch up to Web services. The primary impediment is the lack of accepted standards distributed and adapted across the developer community. Just as standards became the foundation for the rise of Web services, they will serve the same function for workflow.

The good news is that the number of serious contenders is limited to three: WSFL, XLANG, and BPML, each with at least one advantage over its rivals. WSFL offers power, XLANG offers simplicity, and BPML offers backing from some strong industry players. Let's take a closer look at each:

- ***WSFL (Web Services Flow Language):*** An XML-based language that uses directed-graph modeling to define and execute business processes while defining a public interface so business processes can advertise themselves as Web services. Designed by IBM to be part of the Web services technology framework, WSFL complements existing Web services specifications.
- ***XLANG:*** A fundamental piece of Microsoft's BizTalk server, XLANG is a specification for describing message exchange behavior that allows for automation and tracking of processes exposed as Web services. XLANG is considerably simpler than WSFL, and questions exist about whether it has the technical completeness to handle 100% of workflow use cases. Wholly owned by Microsoft, XLANG is not sanctioned by any standards body.
- ***BPML (Business Process Modeling Language):*** Another player in the process-management landscape, BPML was submitted in March 2001 by the Business Process Management Initiative, whose heavyweight industry membership includes BEA, Bowstreet, HP, WebGain, Sun Microsystems, and interestingly, IBM.

The next critical step in the adoption of process management is the establishment of a unified standard that combines the best of these three solutions. IBM submitted the WSFL 1.0 specification to the WS-I for review. After extending the spec, the WS-I returned it to IBM, where it remains while Microsoft and IBM discuss the reconciliation of WSFL and XLANG. One potential for unification may be the OASIS work on Business Transaction Protocol (BTP). BTP is intended to be the underlying protocol offering transactional support for the application layer or for a business process management system.

While we wait for the needed integration and standardization, other important steps have been taken that are speeding the diffusion of these process management specifications. In March SilverStream announced the release of its eXtend Composer XML integration server (version 3.5) as the first software to implement WSFL for process management. Siebel Systems, Inc., will follow this summer with its Universal Application Network, which will let users model business processes as reusable objects that plug into any enterprise application integration system. Other companies are certain to quickly follow suit with support for WSFL, clearly indicating the industry's decision to adopt IBM's WSFL specification.

The end result is not as simple as the adoption of WSFL. The sheer power of WSFL adds a layer of complexity that makes it more difficult to grasp and use proficiently, putting mastery beyond the reach of the majority of VB programmers and necessitating the use of powerful tools. The relative simplicity of XLANG is an advantage in this regard, but this specification is simply not powerful enough to drive the varied demands of process management.

In the meantime, companies are adopting WSFL. With IBM in active talks with Microsoft, it's likely that their respective specifications will be melded to create a solution resembling WSFL. Choosing this specification now guarantees standards-based process management in the future without having to revisit the implementation.

The payoff for the unification of process management standards is the power to represent business processes as a workflow connecting applications within the enterprise streamlining their business processes, making daily operations dramatically more automated and efficient, and allowing the monitoring of previously unmonitorable processes. Of course, let's not forget the ultimate promise of Web services – better collaboration between business units. This is the next big opportunity, but it can't happen until we have one industry-supported standard. Thankfully, we're well on the way toward integration and standardization of these process management solutions.