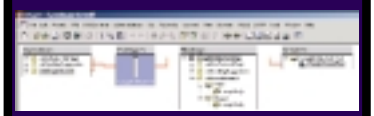
STANDARDS: A LOOK AT BPEL4WS

December 2002 Volume 2 Issue 12

WebServices JOURNAL

.NET J2EE XML

WSJ2.COM



RETAILERS PLEASE DISPLAY UNTIL JANUARY 31, 2002

$6.99US $7.99CAN

12

0 71486 03420 9

SYS-CON MEDIA

PAGE 10

BUILDING INTERACTIVE WEB SERVICES WITH WSIA & WSRP

One step toward making Web services what they were meant to be

FOCUS ON WEB SITES & WEB SERVICES

WebServices JOURNAL
.NET J2EE XML

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

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

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
TECHNICAL EDITORS
David Chappell chappell@sonicsoftware.com
Anne Thomas Manes anne@manes.net
EXECUTIVE EDITOR
Gail Schultz gail@sys-con.com
EDITOR
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
PRODUCTION CONSULTANT
Jim Morgan jim@sys-con.com
LEAD DESIGNER
Richard Silverberg richards@sys-con.com
ART DIRECTOR
Alex Botero alex@sys-con.com
ASSOCIATE ART DIRECTOR
Louis Cuffari louis@sys-con.com
ASSISTANT ART DIRECTOR
Tami Beatty tami @sys-con.com

CONTRIBUTORS TO THIS ISSUE
Andy Astor, Brian Barbash, Henry Bowers, Steve Brown, Paul Butterworth, Paul Cowles, Neil Davidson, Yaron Goland, Anne Thomas Manes, Thomas Obrey, Eilon Reshef, Sean Rhody, Thom Robbins, Mike Sick, Khoong Hock Yun

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

A Face to Remember

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

There's an expression in the industry – "putting lipstick on a pig." It's used to signify a task, or project, that has only cosmetic value and doesn't address the real issues of a situation – it merely makes the pig look a little better. But after all, it's still a pig.

Sometimes as we discuss Web services, we tend to think of the user interface as lipstick. It's an easy trap to fall into; after all, Web services are about communication between machines, so no user interface is actually necessary. Of course, that's a great fallacy.

Web services are seen by many as a mechanism for independent communication between systems, but also as a means of connecting divisions within a company, and enterprises within an industry. In many cases, Web services are an Edge technology – meaning that they exist at the boundary of an information system.

And although the primary technologies in Web services are not user interface elements, the fact that the service sits at the boundary of a system often means that it will end up as a client-facing service. And some clients want to interact directly with the service, not through a programmatic API.

Which is where Web service user interfaces come into play. We need to acknowledge the fact that many Web services are going to be acted upon by user interfaces which sit in front of human beings, as opposed to being invoked by some automated process. Some, not all. But a significant number.

And that's the basic premise of this month's issue. User interfaces, and Web services. I see interfaces as the part of Web services that gets overlooked while we concentrate on making sure that we can execute complex business logic, or create a business transaction, or ensure that our service is secured from unwarranted usage. And it's true that there is no widespread agreement on Web services interface standards (although work is underway on several). But that doesn't remove the need to address the issue. Actually, it's probably better that no one standard exists – it would defeat the machine-independent premise of Web services. But at the same time, as architects and developers, it is necessary to recognize that many Web services will require a user interface, and to understand how to use them effectively. Whether it's JSP; ASP on the Web; or Java or C# creating a thick client; or some other programming language, we need to investigate how to put a face on our service.

So in this issue we're exploring various facets of user interface interaction with Web services, including tools and technologies that make our lives easier. While it may not be a core part of Web services, it's still very necessary.

It's not just lipstick on a pig.

Putting an Interface to Your Web Services

The right plan means a better user experience

Thomas Obrey

Thomas Obrey cofounded PixelMEDIA in 1994. As chief operating officer, he is responsible for the day-to-day operations of the company, managing the development and professional services staff, and providing senior-level strategy and consultation services to the company's client base. PixelMEDIA provides strategy, design, and implementation services for Web, multimedia, and IT initiatives.

As the Web services market continues to shift and evolve in response to global accessibility needs and consumer demands, companies are continuing to extend and enhance their Internet-based services. Planning and following an integrated design process are invaluable in ensuring that a project is successful. Companies that don't possess the variety of disciplines involved in the creative process can partner with external resources that are agile and experienced in those areas.

Developing an enjoyable user experience online is an important part of building and maintaining trust with existing customers and prospects searching for specific services.

In order to create the user experience, you need to define your users and the top tasks they will perform as well as identify the potential scenarios for how they will interact with your application. Once you have this information, you can create an overall concept that provides the direction for the user experience.

Good experiences can build trust with customers, while an unsatisfactory experience can destroy a company's credibility with their existing customers and prospects. By following an integrated design approach, a company can design and develop an easy-to-use interface that creates an enjoyable experience. The following is a set of steps that can give your company insight into the process of user interface design.

First, take your user research and reference it throughout the process. User research can be gathered from conducting customer surveys, sending out questionnaires, or interviewing key users of your offering. If you have the opportunity to include real end users in your process – take it!

Next, start mapping the tasks that a user can accomplish based on the features you've defined for your application. Use your scenarios (defined based on your user research) to validate the flows and identify any incomplete paths that may have been missed.

Begin mapping out your menu systems while keeping in mind how your audience will want to interact with the service. Involve real users in the process with usability testing terms, labels, and workflows to gain real feedback and direction for how to improve their interactions before you have developed anything.

The next step in the process is designing a solid user interface. This involves identifying the different elements for each screen, areas of focus, and the features that allow users to travel through and manipulate the content. Success at this stage demands that you involve real users in the process by running task-based usability tests that reveal how a user moves through the information and how they rate their experience with the application.

Once your prototype has been successfully tested with real end users, a knowledgeable development team should implement the solution. By keeping in mind issues of security, reliable data transmission, browser compatibility, and platforms, the system can respond in a way that meets the expectations of its audience. It also affects how successful the experience is because it directly impacts how a user interacts with the design.

If it sounds like a vast array of knowledge and creative discipline are required to achieve all of this – you're right! Many companies don't possess the many disciplines required to create, develop, and maintain their Web service. They need to look outside their own firm to take advantage of companies that gather the necessary skill sets and are extremely agile in designing and developing Internet-based applications.

Finding a partner who can provide strategic direction and streamline the design process can ensure a positive user experience for your customers. It's never too early to start planning for your next release, and define how you're going to meet the increasingly high demands of your customers.

Written by Henry Bowers

Building Rules into Web Services Applications

Use past successes – and failures – to find the best path to your project's needs

The emergence of Web services is forcing sites to substantially rethink how existing applications can and should work together.

Previously, considerations about where certain functions should execute focused principally on the tier. Should the client, the application server, or the database management system perform this processing? Web services, however, force the question in a horizontal direction: Which server on this tier is best suited to handle this processing? And, even more frequently, how can this application be partitioned intelligently so the right pieces are on the correct servers? Design tools are responding to these changes with model-driven architecture and services-oriented application development, two approaches that capture and represent the salient factors in the design of loosely coupled, distributed applications.

However, most sites that have committed to pilots of Web services cannot wait for best practices to mature past these nascent technologies. These sites need to partition applications now. Thus, they are highly dependent on learning from the successes and failures of other sites working on similar challenges. From these experiences, they can begin to divine the correct path for their own projects.

This approach is hampered by the limited amount of information available about Web services projects. The lack of information is particularly acute as regards the partitioning into Web services of projects that handle business rules. This article discusses several ways in which applications can be intelligently partitioned to make the best use of business rules. The models are drawn from our client consulting experiences with this particular problem.

Using Business Rules

Ask most IT professionals what they place on their application servers, and they'll respond "business logic." If you ask for a more specific definition of business logic, they will usually reply with application details from which flow explanations of implementation particulars. Because business logic is perceived as unique at every site, little effort is made to view its elements as anything but a series of unique detail-level components. This view, however, does not generally reflect the reality. Companies in like industries generally have similar processing to perform (over and above standard accounting applications). For example, insurers all have claims and underwriting systems; they all must perform actuarial analysis of pricing, process claims more or less the same way, and issue policies using similar constraints. In addition, their regulatory reporting requirements look a lot alike.

What distinguishes one insurer from another is the criteria used for accepting an applicant, establishing the kinds of insurance the candidate qualifies for, pricing the policy, and applying policy-specific features and benefits. These distinguishing characteristics are implemented through business rules. Rules determine who qualifies for what, at what cost, and with what options. Because of this, insurance companies, mortgage lenders, financial services companies, government agencies, and hundreds of other businesses rely on rules as a central part of their business logic. In fact, it is fair to say that business rules capture the unique competitive proposition of a business. Because of this central role, the specification, enforcement, and management of rules should be a separate and specialized portion of business logic.

Transitioning Rules to Web Services

Many rules implementations today do not distinguish rules from straight data processing very well. In fact, rules are most often implemented in pages of if/else statements buried in larger C, COBOL, and Java applications. One client had more than 400 pages of if/else statements; as a result, maintaining the business rules or adapting them to a new business product was very difficult, if not impossible.

Companies that haven't carefully segregated rules from other business logic are confronted with two choices: divide the application as best they can (and one would hope with segregation of rules a key objective) or simply slap a Web services front end on the monolithic application. Given the tight budgetary constraints at most IT sites today, the latter approach is the most common. It doesn't require a lot of effort at a programming level. Such implementations generally place software such as a principal mortgage application module on an intranet as a Web service. Other applications send the module large SOAP–wrapped XML records containing all relevant applicant data for approval. When the

Author Bio

Henry Bowers has spent more than 15 years in the high-tech sector, building and managing software products for both private industry and government. He has more than seven years' experience working with rules-based systems and business rules in general. Henry is currently a product manager for business rules at ILOG (www.ilog.com). For more than 10 years, ILOG's innovative enterprise-class software components and services have helped companies maximize their business agility and improve operating efficiency.
HBOWERS@ILOG.COM

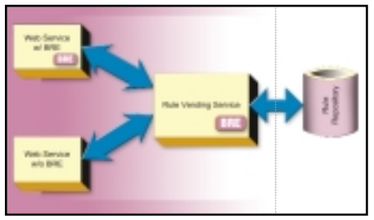


FIGURE 1 **Rule Vending: Applications call a Web service to process rules. This service calls another Web service to provide it with the freshest rules required by the application.**

software approves the application, it returns to the client the necessary text to drop into loan document templates.

One of our customers is using a variant of this approach to enable its agents to sell policies in the field. Previously, agents would fax the data to an underwriting department and wait for an answer. Shortly, they will use wireless access to virtual private networks (VPNs) and Web services to get approval and the necessary documentation back on their machines with only a slight delay. This allows the agents to close a policy sale on the spot. The advantage of this approach is its simplicity. Existing software that a site knows and trusts is unchanged but for the front-end means of access. This solution is quick, inexpensive, and effective. However, it does not make optimal use of business-logic components.

Better Granularity

The second approach is to separate business rules from the business logic. Depending on the construction of the applications, this can be fairly easy. For example, the 400 pages of if/else statements mentioned earlier are located in a handful of functions. These routines can easily be wrapped up as a Web service. This service is sent the same parameters with which the original functions were called and returns the same values after executing the rules. This straightforward division is somewhat more complex if global variables in the application are modified, but even this effort is worthwhile given the advantages of this approach.

The first advantage is that other applications can now make use of the same rules. Suppose, for example, that a mortgage insurance company wants to establish what proportion of its loan portfolio is at risk. It might obtain updated information on its customers and run the data through the qualification engine to see who would be rejected or assigned a lower credit score. To do this analysis, the company needs to access the business rules without dealing with the other processing of a loan application. It is not interested in receiving text streams for loan documents; it wants only the results generated by pushing data through the qualification rules.

Web services provide a unique opportunity to implement this kind of reusability by carving out sets of rules into callable services. In addition, they encourage the conversion of many standalone data formats into XML, which has the benefit of simplifying overall application integration (at the cost of performance, which in high-volume settings is far from negligible).

Sites that choose this path are often surprised at how quickly benefits become apparent. The separation of rules processing as Web services can be implemented progressively. Complete conversions of applications are unnecessary. Also, once rules are segregated as Web services, new opportunities for their use appear quickly. For example, business analysts can much more easily perform what-if scenarios by rolling hypothetical data through the rules.

A common implementation detail of this approach is the migration of rules from a code-based implementation to a business rule engine (BRE), an integral component of a business rule management system. A BRE applies business rules to application data, generally in a highly optimized fashion. However, its real importance is that it enables firms to specify rules in business terms rather than in code. This step permits sites to move rules formulation and testing outside of IT and into the hands of business analysts. These analysts now can formulate rules using their own vernacular, test the results, and go live when they're ready. Meanwhile, programmers are freed from the conversion of business policies into code and so can focus on other portions of the application.

By using Web services and a BRE this scenario becomes possible. The rest of the application then interacts with the rule sets, which are now flexible and highly amenable to use in one-off situations. Because of this benefit, BREs are increasingly viewed as a central best practice for the implementation of policy-intensive applications.

Toward Business-Rule Vending

As companies gain expertise with Web services and BREs, they find that deploying and reusing volatile business rules presents new management challenges. With business rules that change frequently, the task of deploying rules varies as the changes occur. Ensuring that all affected applications have been updated becomes increasingly tedious and error-prone. Companies with this predicament may avail themselves of an additional implementation model that simplifies and automates distribution of updated rules. This implementation model introduces the concept of business-rule vending.

The rule-vending service is implemented as a Web service with a BRE and an interface to a business-rules repository (see Figure 1). This vending service receives requests from other Web services and does two things with these requests: it determines what business rules in the repository are appropriate to apply, and it returns those rules in the requested format. Determining which business rules to apply and return consists of matching the context and the origin of the request with the sets of rules available in the repository. The matching logic is implemented using rules as well (these matching rules are stored and maintained with the other rules – in the repository). The rule-vending service has the ability to return an XML-based file containing the applicable business rules, a link to a Web service that applies a specific set of business rules, or the results produced by executing the business rules on the spot.

This vending service simplifies distribution of rules by providing a central (and potentially a single) location for business rule location and distribution. It also enables sites to implement even more sophisticated arrangements such as publish-and-subscribe models for business rule distribution.

The use of BREs in conjunction with an atomic implementation of rules enables sites to have extremely effective and efficient rule processing via Web services.

Written by Eilon Reshef

Building Interactive Web Services with WSIA & WSRP

WSIA and WSRP are new Web services standards that enable businesses to create user-facing, visual, and interactive Web services that organizations can easily plug-and-play into their applications and portals. This article will familiarize you with these technologies and illustrate how they can help your businesses.

One of the main promises of Web services is enabling the assembly of Web applications from functional components distributed across multiple locations. However, until now the assembly of visual, rich, interactive Web applications with a cohesive flow and look-and-feel has been a challenge. Custom programming is required to create a user interface tier for each new Web service, resulting in set-up and maintenance efforts that render business initiatives cost prohibitive as the number of components increases.

Web Services for Interactive Applications (WSIA) and Web Services for Remote Portals (WSRP) are standards for user-facing, presentation-oriented, interactive Web services intended to simplify the creation of distributed interactive applications.

With WSIA and WSRP, Web services include presentation and multipage, multistep user interaction. This lets service users plug them into sites and portals without custom development and leverage new functionality available in future versions of the service without the need for additional custom development (see Figure 1).

AUTHOR BIO:

Eilon Reshef is the vice president of products at WebCollage. WebCollage provides comprehensive software solutions for interactive web services, allowing companies to package their existing customer-facing Web applications as Web services and rapidly share them with multiple business partners.
EILON.RESHEF@WEBCOLLAGE.COM

History

WSIA and WSRP stem from parallel efforts initiated in the middle of 2001. Portal vendor Epicentric spearheaded the Web Services User Interface (WSUI) initiative to address the lack of a standard user interface layer as part of the Web services stack. IBM initiated its own effort: Web Service eXperience Language (WSXL). WebCollage, a software vendor providing a platform for integrating interactive Web applications, wanted to standardize its customer implementations based on an initiative called Interactive Web Services. In parallel, many portal vendors recognized the need to address the same problem in the context of portal toolkits: how to quickly plug remote interactive services (called "portlets") into a portal without custom programming for each remote service.

The efforts were consolidated into two working groups under the umbrella of OASIS, the organization behind ebXML and other XML-related standards. WSIA focuses on the framework for creating interactive Web services, and WSRP defines the interfaces to include portal-specific features. Today the working groups include more than 30 industry-leading vendors from different industry segments: application server vendors (BEA, IBM, Oracle, Novell, Sun), pure-play portals (Epicentric, Plumtree), interactive application integration vendors (WebCollage, Kinzan, Citrix), and enterprise application providers (Peoplesoft, SAP).

Version 1.0 of the specifications includes the interfaces common to the two groups, and is currently under review.

WSIA/WSRP and the Web Services Stack

WSIA and WSRP define a set of APIs that allow applications to leverage remote interactive services. The APIs are built on top of the existing standard Web services technologies:

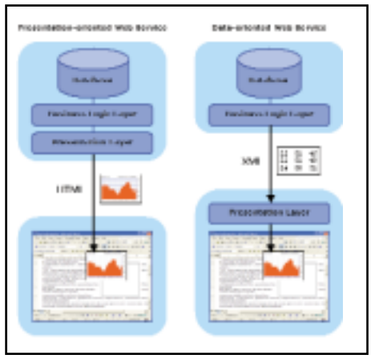


FIGURE 1 **Data vs. presentation-oriented Web services**

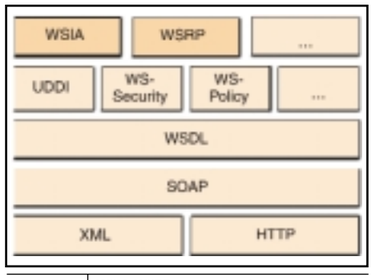


FIGURE 2 **WSIA/WSRP and related technologies**

- ***SOAP:*** For the service invocations
- ***WSDL:*** For formally describing the WSIA and WSRP service interfaces
- ***UDDI:*** For publishing, finding, and binding the WSIA and WSRP services

WSIA and WSRP will leverage emerging Web services security and policy standards as they become available (see Figure 2).

Business Scenarios

WSIA and WSRP can be used in a variety of business scenarios. Described below are three scenarios considered in the working groups.

The "Indirect Customer" Gap

One of the common gaps in e-commerce today is between manufacturers and their indirect customers. For example, a consumer who researches electronic products at a retailer site online cannot find the information and tools needed for the process because the retailer sites usually lack the advisor and configuration tools typically used for this kind of task. The consumer is forced to go to the manufacturer site to find these online applications, and then go back to the retailer site to complete the purchase.

In the ideal scenario, online distributors and resellers would be able to offer manufacturer tools and content as part of their e-commerce offering. However, the technical challenge is that many of the tools developed by manufacturers – online advisors, product configurators, product capsules – are sophisticated in nature. Thus, they cannot be easily described using XML APIs. In addition, XML APIs don't provide the simplicity needed to scale integration to thousands of channel partners because they require each reseller to develop custom code for each of the manufacturers APIs.

WSIA allows companies who sell indirectly to package their online sales and service effectiveness tools as WSIA/WSRP services. This makes it possible for their channel partners to easily plug the tools into their own e-commerce sites and offer them to their customers. It allows manufacturers to make it easier to buy their products while maintaining their traditional role in the distribution chain.

The "Corporate Employee" Gap

One of the common goals for business-to-business service providers (e.g., providers of 401(K) programs) is to transition as many processes as possible into self-service tasks over the Web to reduce operating costs. The main challenge is that the target audience for these online services is mainly corporate employees who interact primarily with their employer's intranet. To locate and use an external service, the employees must leave their familiar environment and locate the remote service. This extra effort reduces the adoption rate for the online services offered.

In the ideal scenario, employers would be able to integrate such external services into their intranets and their corporate portals, and make them easily available to employees. However, the current cost of integration makes it cost-prohibitive to all but the largest organizations. With WSIA and WSRP, providers of service applications can package their financial services applications as WSIA/WSRP services, making it easy for corporate customers to integrate into their intranets. It allows business-to-business service providers to develop a closer relationship with their corporate customers and with its employees.

The "Build Once Deploy Anywhere" Problem

Fortune 500 companies provide a variety of online services to their employees, customers, and business partners. To do so, they often create multiple portals – one for each target audience.

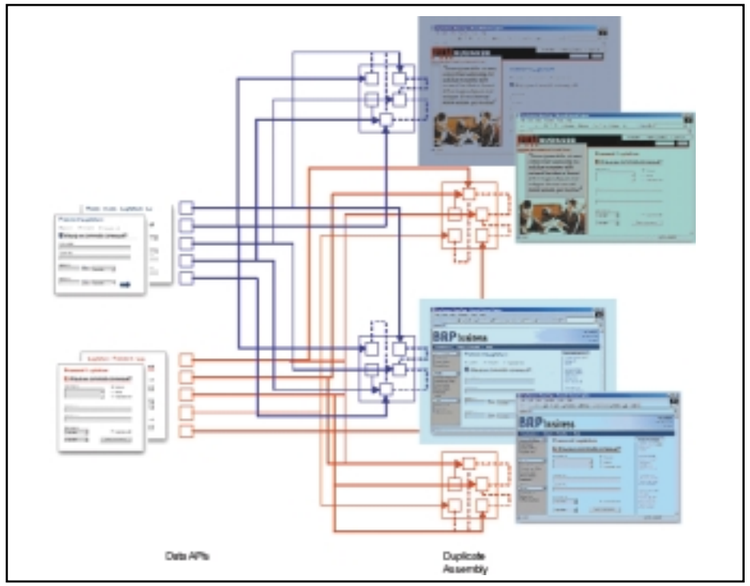


FIGURE 3 | **Assembly with XML APIs**

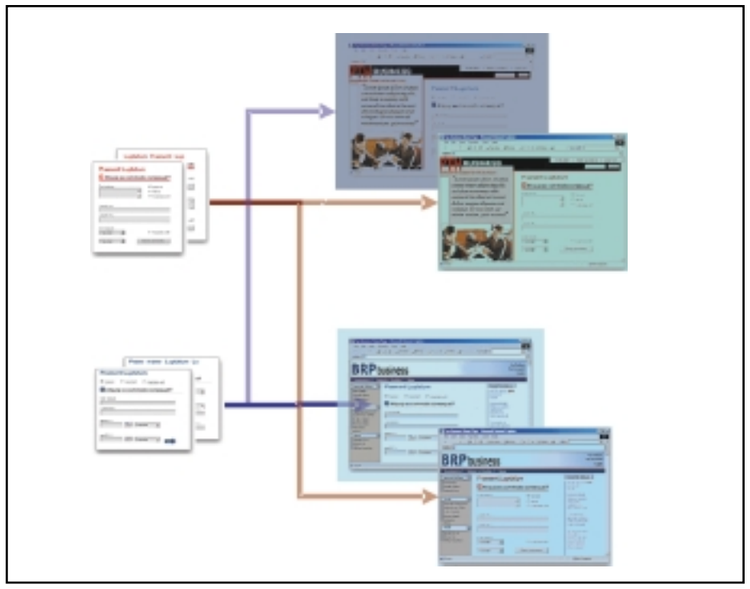


FIGURE 4 | **Plug and play with WSIA/WSRP**

One of the main challenges such companies face is the need to make a single application selectively available through each of those portals. With WSRP, companies can package their applications as "portlets" and make them available to portal administrations for integration. Portal administration can plug those portlets and make them available to the portals' end users without custom integration, while the application would still be centrally hosted and managed.

Technical Motivation

In many of the scenarios described above, WSIA and WSRP provide an alternative to data-centric Web services and simple XML APIs. When exposing only an application's business logic with XML APIs, API users incur the effort of integrating the application into their sites (see Figures 3 and 4). The approach falls short for complex interactive applications whose flow spans several Web pages because:

- Application providers need to decompose the interactive application into the underlying atomic functions, which requires a significant effort.
- API users need to recompose the underlying APIs with the workflow and the presentation into a coherent interactive application, resulting in high setup costs for the API user.
- As the APIs evolve, API users need to upgrade their application to comply with the latest interface to incorporate new functionality. This results in high maintenance costs for the API user.
- It's hard to ensure that the quality of the reassembled application meets the application provider's quality standards. The quality of the implementation depends heavily on the technical expertise of the API users, and the functions may not be assembled in the right way in each instance.

These problems stem from a single fact: instead of sending ready-to-use (customized) HTML markup fragments that the API user can embed without any further processing into their Web site, data-oriented Web services require further work, often quite complex, to transform the XML into HTML, and manage the multistep interaction. Such an approach results in code duplication, because each API user has to reimplement the same composition logic of the application.

Initial vendor "hype" in the Web services space suggested that new assembly tools would solve this problem by providing "assembly platforms" that automate the creation of interactive applications. In reality, companies realized that using XML APIs and SOAP is no easier than using APIs available today in local code libraries. To create a composite application, developers must understand the application-specific seman-

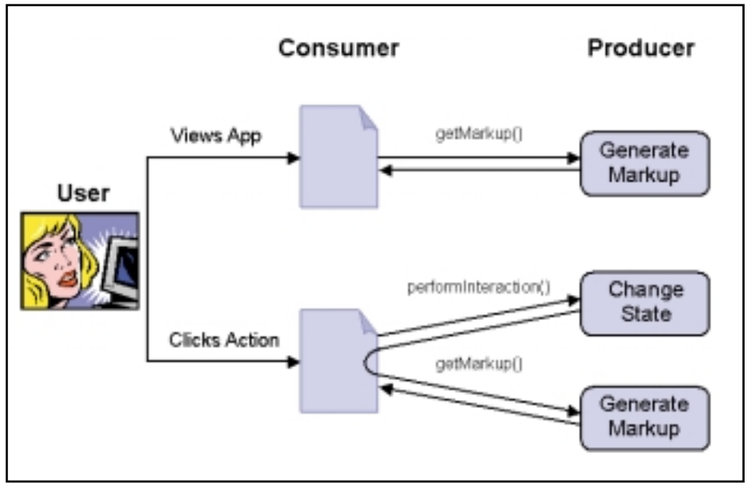


FIGURE 5 | **WSIA/WSRP Data Flow**

tics of the different functions offered, and invest time and resources to compose the different functions into a usable interactive application.

Many high-value business scenarios involve complex, interactive services that require a user interaction and a presentation model. By setting an industry standard to define this user interaction and presentation, WSIA and WSRP pave the way for providing richer and more compelling Web services for businesses and users alike.

WSIA/WSRP Technology Overview

WSIA and WSRP define a set of APIs that allow developers to produce and consume remote interactive Web services. They define three types of actors:

- ***Producer:*** Represents the service provider hosting the remote interactive Web service (for example, weather.com as a weather service provider)
- ***Consumer:*** Represents the entity integrating the remote service into its Web application, oftentimes using a portal toolkit (for example, Yahoo Weather or a corporate portal)
- ***End User:*** Represents the person that comes to the Consumer Web site to use the Producer's application in the Consumer's context

In a nutshell, WSIA and WSRP fulfill the following roles:

- Define the notion of valid markup fragments based on the existing markup languages, such as HTML, XHTML, VoiceXML, cHTML, etc.
- Provide a set of standard calls to enable a Consumer to request these markup fragments from a Producer based on the existing state
- Supply a set of calls that support the concept of multistep user interaction and preservation of state across those calls

There are four central parts to the WSIA and WSRP APIs:

- Retrieving markup fragments (encapsulated in the getMarkup() call)
- Managing state
- Handling user interaction (encapsulated in the performInteraction() call)
- Supporting customization and configuration

Generating Markup Fragments

The getMarkup() call requests a markup fragment based on the state of the Web service.

When the End User views a page that includes a remote interactive service, the Consumer invokes the getMarkup() call. The operation receives state information (see below) and returns a fragment of standard HTML code. The Consumer embeds this markup into the page, and returns the completed page to the user. In the case of WSIA, the container page can be part of any Web application. In the case of WSRP, the container page is generated by a portal toolkit.

There are certain guidelines governing the markup returned by the getMarkup() call. The most important one relates to user interaction. When the markup contains links and forms, they point back to the Consumer application, with the Web service's new state. This process, which is called URL rewriting, ensures that any further user interaction with the interactive service is routed through the Consumer application (see below). It also ensures that the next getMarkup() is invoked with the new state.

Managing State

State is critical because WSIA and WSRP services are typically interactive (for example, a three-step checkout process in an online store), requiring several calls from the Consumer to the Producer, each dependent on previous configurations, data entry, and actions.

Because WSIA and WSRP are connectionless protocols, the Producer must be able to return information to the Consumer, with the understanding that this information will be sent back by the Consumer. Two types of state information exist:

- ***Navigational state:*** Allows the current page to be correctly generated, including on a page refresh or through a bookmark. This type of state is sent from the Producer back to the Consumer, which typically stores it in the URL. It is transferred in a parameter called navigationalState.
- ***Transient state:*** While navigational state defines the current "page" of the Web service, transient state is stored on the Producer and usually related to a sequence of operations. The Consumer is sent only a "handle" to this state, much like HTTP sessions. If the Producer decides to use a transient state, the Producer returns the handle to it. The Consumer is then responsible for attaching it to any subsequent calls.

Handling User Interaction

To support user interaction, all the URLs embedded in the markup fragment returned by the remote Producer service must point back to the Consumer application. To do so, the Consumer sends a URL template as part of the invocation of the getMarkup() call. For example, the consumer may send the following URL template:

```
www.consumer.com/path?ns={navigationState
}&si={sessionID}
```

The Producer is responsible for generating a markup fragment in which all the interaction URLs point back to the Consumer. For example, the Producer may generate a link that points to the following URL:

```
www.consumer.com/path?ns=page2&si=4ABB33A
```

WSIA and WSRP also provide an alternative mechanism that allows the Producer to create URLs that conform to a predetermined pattern. The specification allows Consumers to parse the markup and to rewrite such URLs to point back to their application.

Consequently, when the user clicks on a link or submits form data, the Consumer application gains control and has access to the action carried out by the user. The Consumer application then invokes the performInteraction() call. Upon receiving the call, the Producer handles the action and returns an updated state. To redraw the complete page, the Consumer then invokes the getMarkup() call to receive a markup fragment. Because the state of the Producer has changed since the previous getMarkup() call, the markup fragment returned is typically different from the one previously returned. The End User can then perform another action, which starts a new interaction cycle (see Figure 5).

Supporting Customization and Configuration

To support a situation where a single centrally hosted service can be used across multiple consumer applications and across multiple individual users, WSIA and WSRP support multiple configurations of a single service. For example, a remote interactive product catalog may be configured to display different prices depending on the Consumer application.

WSIA and WSRP provide a set of function calls allowing Producers to expose multiple preconfigured versions of the same service. It also allows Consumers to create and manage additional configurations of the same service, as well as allowing end users to create such configurations. In the context of WSRP, such configurations are static (that is: defined in advance), whereas WSIA plans to add support for dynamically configured remote services.

The Market and Future of Interactive Web Services

WSIA and WSRP are important technologies that help bring the promise of Web services to end users by providing a standard to manage user interaction and application display. They enable business partners to integrate each other's online applications seamlessly, offering a more compelling experience to their customers.

These technologies are complex under the hood and can only thrive if vendors deliver on the promise of building tools to manage the complexity. Given the breadth of the specification, it is not expected that companies will develop WSIA and WSRP solutions in-house. It is likely that they will instead rely on tools from vendors, keeping their focus on creating the application functionality.

Although the WSIA and WSRP standards are still evolving, several vendors are already providing practical solutions in this space. Such solutions will likely migrate to the standards as they mature. You may expect advances to be made in the following directions:

- Epicentric (www.epicentric.com) and portal vendors are focusing on aggregating existing remote Web services into portals.
- IBM (www.ibm.com) and J2EE application server vendors are building tools to make it possible to create Web services from scratch that can harness the power of WSIA and WSRP.
- WebCollage (www.webcollage.com) and interactive application integration vendors are focusing on helping companies package their existing Web applications, so that they can be used remotely by customers and business partners alike.

As Gartner pointed out in a recent analysis of the Web services space, "Successful software vendors and Web services providers will find innovations in usability and user interface to be a source of competitive advantage. Better-than-average usability is one reason why Yahoo, Amazon, AOL, Google, and Palm came to dominate their respective markets."

The vision of intertwined Web services with rich user interfaces providing a complete package to the end user can be realized only if the presentation and user interaction problem is solved. WSIA and WSRP are a step in the right direction and go a long way towards making Web services what it was designed to be: pluggable application components that can be assembled into visually rich composite Web applications.

Web Services Journal
.NET J2EE XML

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

COO / CFO
Mark Harabedian mark@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-Mussa 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
Grisha Davida grisha@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
CUSTOMER SERVICE REPRESENTATIVE
Margie Downs margie@sys-con.com
JDJ STORE MANAGER
Rachel McGouran rachel@sys-con.com

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

ACCOUNTING
FINANCIAL ANALYST
Joan LaRose joan@sys-con.com
ACCOUNTS RECEIVABLE
Kerri Von Achen kerri@sys-con.com
ACCOUNTING PAYABLE
Betty White betty@sys-con.com

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

SYS-CON MEDIA

Reviewed by Brian R. Barbash

XMLSPY 5 Enterprise Edition

A valuable tool for the Web services developer's arsenal

About the Author:

Brian R. Barbash is a consultant for the Consulting Group of Computer Sciences Corporation. He specializes in application architecture and development, business and technical analysis, and Web design. BBARBASH@CSC.COM

XMLSPY Enterprise Edition version 5 provides a powerful set of XML tools and utilities to develop and test XML-based applications and Web services. Several new features that expand on its XML tool base are available in this new version, including a WSDL editor, XSL stylesheet debugger, and Java code generator. And as before, XMLSPY continues to provide powerful utilities for developing schema, DTD, and XML documents.

ALTOVA

COMPANY INFO

Altova Inc.
900 Cummings Center
Suite 314T
Beverly, MA 01915
Web: www.altova.com
Sales: www.altova.com/order

LICENSING INFORMATION

For a free 30-day evaluation copy go to www.altova.com/download

WSDL Editor

The WSDL editor is a new utility available within the main XMLSPY IDE that provides a graphical editor for WSDL documents. This tool presents a WSDL structure broken into four segments: operations, port types, bindings, and services. Figure 1 shows the editor with the Google API WSDL specification loaded.

Operations are essentially method signatures for a particular service, complete with input, output, and fault child elements. The parameters for each element may be added and removed using the context menus available while working in the WSDL editor. Parameters may be assigned types based on namespace references to existing schema documents.

The list of bindings allows the developer to create the concrete specifications for the method signatures defined in the operations panel. Each binding is mapped through a specific port type to create the association. The services panel provides the definition of the physical location of the service and the relationship to the binding that provides the structure and protocol information for the service.

The WSDL editor also provides validation capabilities to check the resulting WSDL definition integrity. Developers may make changes to the type definitions of the WSDL elements by modifying the underlying schema. Overall, the WSDL editor is a powerful utility for creating new and working with existing WSDL specifications.

SOAP and the SOAP Debugger

XMLSPY provides a set of tools to help application developers create and debug Web services and SOAP messages. The application can be set up to act as a proxy server, intercepting SOAP requests and responses and presenting them to the user for inspection and debugging. It also has the ability to create SOAP documents from a WSDL file, useful for creating test data and messages for testing individual Web services.

SOAP documents may be created manually or from a WSDL specification. To create a request from a WSDL specification, XMLSPY reads the WSDL file from a specified URL or a WSDL document already loaded into the IDE. Figure 2 shows the resulting SOAP request from the Google Search API.

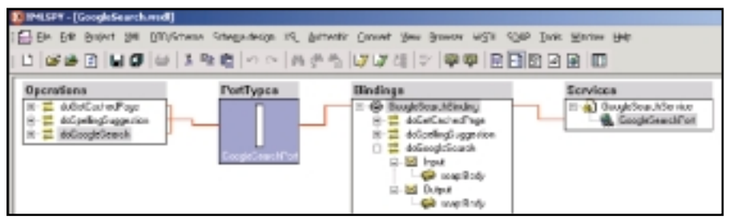


FIGURE 1 | **WSDL editor with Google API WSDL specification loaded**

Processing SOAP Responses and the XSLT Debugger

XMLSPY may be used to help develop the presentation of call results to Web services through its XSL Stylesheet authoring tool. This component allows developers to create transformations with minimal knowledge of the XSL language.

The Stylesheet Designer is a separate executable included with XMLSPY Enterprise Edition. It is a simple drag-and-drop interface for developing XSLT stylesheets. To create an XSL stylesheet, open an existing XML Schema and select those elements or attributes you wish to show in the output by dragging them into the desired position inside of an editing window. The stylesheet designer also supports conversion of an existing HTML file into an XML Schema, XSLT stylesheet, and the XML contents of the page.

Anyone who has developed XSL stylesheets can identify with the struggles of debugging their results. The new XSLT debugger brings traditional debugging capabilities to the world of stylesheet development. Within the main IDE, similar to the SOAP debugger, an XSL debugging session may be started to step through the XSL transformation.

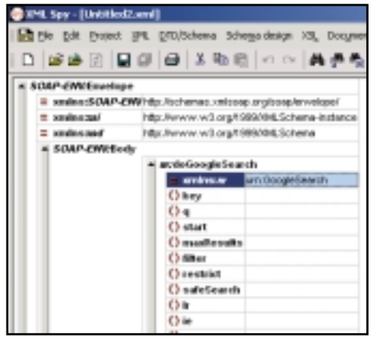


FIGURE 2 | **Generated SOAP document**

As shown in Figure 3, the developer is presented with four main panels. The top portion of the screen, from left to right, includes the XML document that is being transformed, the XSL stylesheet highlighting the current execution step, and the transformation output. The bottom portion of the screen displays in-process information including the call stack, watch list, local variables, and template definitions.

Once started, the debugger behaves like any standard debugging engine, allowing the developer to step into, over, and out of XSL transformation processes. This utility greatly eases the tasks of developing and debugging XSL stylesheets.

XMLSPY also supports the use of XSL Formatting Objects for creating high quality output documents, such as PDFs, from XML content, and includes a single-click preview window. It also supports the use of external XSLT processors, including Xalan and MSXML.

XML and Schemas

The true strength of XMLSPY is, of course, in its application to the design and development of XML documents, schemas, and DTDs. The construction of XML Schemas is intuitive thanks to its graphical representation of a document. The properties of each schema node are easily editable through context menus available when a node is selected. Here the type, occurrences, and attributes for each node, among other characteristics, may be set. Annotations for nodes are displayed as summary text below each node.

To create DTDs and XML documents, XMLSPY presents the definition in an outline format. When working with DTDs, which are similar to schemas, the properties of each element are editable through context menus. The individual entities within the document may be expanded or contracted for readability.

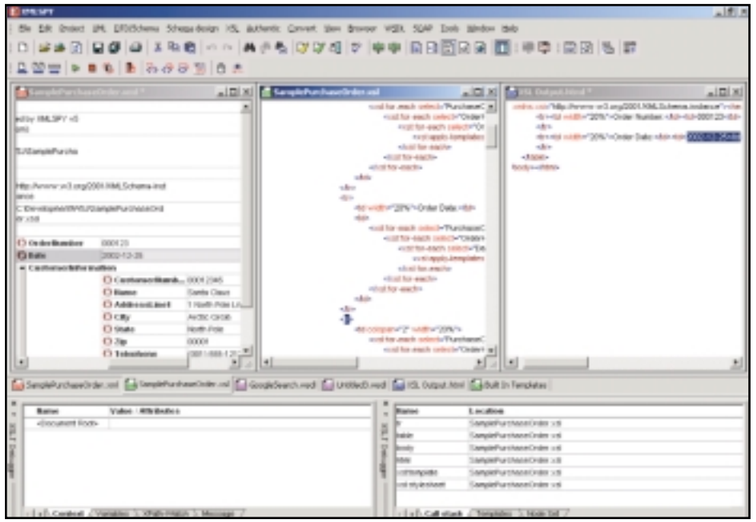


FIGURE 3 | **XSL stylesheet debugger**

XMLSPY also provides the ability to interact with databases when working with schemas and XML data. XML Schema documents may be created based on the table structure within a database. Any database is accessible via ODBC or ADO – e.g., Oracle, SQL Server, DB2, etc. Data may be imported into the application in the same manner using a single table or a select statement. As an example application, using XMLSPY's XSL stylesheet processing capability, data imported from a database can quickly and easily be converted to an alternate database schema. The results can then be exported to a database that is mapped to the alternate schema. This provides an effective environment in which to develop, test, and debug a possible data conversion utility.

A new feature in version 5 is the ability to generate program code based on a schema definition. The process builds Java or C++ classes that represent elements and data types within a schema. The Java classes are created using the JAXP API for XML while the C++ classes implement MSXML 4.0. Code generation may be customized through template files that control mappings between schema types and language primitives, the XML API used, coding conventions, and base classes. Future versions of the code generator will include support for additional languages, such as C#.

XMLSPY also provides a utility to create a set of Schemadocs for an XML schema. The output for each element includes an image depicting the element and its immediate children, links to the definitions of the children, the annotation assigned to the element, and the source code that defines the element. Schemadocs may be created in HTML or Microsoft Word format.

Summary

XMLSPY 5 Enterprise Edition continues to deliver excellent XML tools and utilities with powerful new capabilities. Its wide range of functionality allows it to be creatively applied to a Web services development environment as well as traditional development efforts using XML data. Its support of Web services standards and its ability to debug and test services and the data used in transit makes XMLSPY 5 Enterprise Edition a valuable tool in a Web services developer's arsenal.

A Natural Progression

Web Services Reaching Vibrant New Stage

Written by Andy Astor

We're at a major inflection point in the development of Web services; one that can be paralleled against the evolution of previous infrastructure technologies, including databases and programming environments. The next year will see certain vendors step forward to offer value-added components that will help fill in many of the gaps that currently exist with Web services, generating even more corporate interest than is seen today.

AUTHOR BIO:

Andy Astor is vice president for Enterprise Web Services at webMethods and is responsible for driving the company's Web services strategy and execution. Prior to joining webMethods, Andy led Customer Access Systems for Dun & Bradstreet, where his work included the development and launch of one of the earliest commercial Web services. He is on the advisory board of **Web Services Journal,** *and was IT Strategy track chair for the 2002 Web Services Edge conference in New York City.*
AASTOR@WEBMETHODS.COM

While some may be disappointed in the development of Web services to date, particularly when viewed against initial expectations, it's important to view the progress within the context of previous infrastructure technologies. In that context, Web services appear to be right on track.

Evolution

Web service–based integration is actually the evolution of an idea that's been around for 40 years – service orientation. As early as the 1960s, elementary integrations took place through the use of subroutines that called upon reusable processes. As I wrote in this magazine several months ago **(*WSJ,*** Vol. 2, issue 2**),** this concept has evolved over the years through structured programming, client/server development, object orientation, components, and on to Web services today. Web services are essentially a technique for Web-based application-to-application integration, much the same way that client/server was for non-Web–based distributed computing.

Applications for the Web can be split into two camps – those for human-to-computer interactions and those for computer-to-computer applications. Web service–based integration is all about moving the second group of applications forward. It can do for system-to-system communication what HTTP and HTML did for the browser-based Web.

So, where are we today? All signs point to our being on the cusp of a significant turning point in the development of Web services. There is a standard pattern to the development of technology solutions. It begins with a problem being encountered in multiple situations, and solutions being developed on a one-off basis to address the specific challenge. Over time, entrepreneurs deduce that a generic solution can be developed to address the problem, and they create powerful, proprietary vendor platforms to solve the challenge. As the problem domain matures, standards develop, at which point a shake-out begins and the companies that survive are those that successfully do two things: first, they embrace the developing standards, and indeed demonstrate leadership in their development. Second, they continue to innovate at the top of the stack; in other words, in related areas where standards have not yet matured.

We saw this pattern develop in databases, where Oracle took the leadership position; more recently, we saw BEA take a leadership role in the J2EE application development space. The same pattern now appears to be emerging in the integration space. As with the other technologies, the first development phase of integration platforms began with companies experiencing a problem: they increasingly needed to integrate their standalone enterprise applications. Invoicing applications needed to talk to accounts receivable applications, and CRM applications needed to plug into existing customer data warehouses. Over time, powerful integration platforms sprang up to address heterogeneous application interoperability and integration. As the space matured and became better understood, we started to see the development of Web services standards around integration in a Web-based world.

As a result, we now have agreements on a set of Web services standards that address a few key interoperability issues – standards like Simple Object Access Protocol (SOAP) for sending Web service messages, and Web Services Description Language (WSDL) for providing a simple way to describe how to invoke a Web service on the data being exchanged.

The companies leading in the integration space at the moment are the companies that recognize the importance of these standards, are leading their development, and are incorporating them into their platforms. At the same time, these leaders are plugging the gaps in the standards by providing a complete stack of integration capability, and continuing to innovate at the "high end" of this stack.

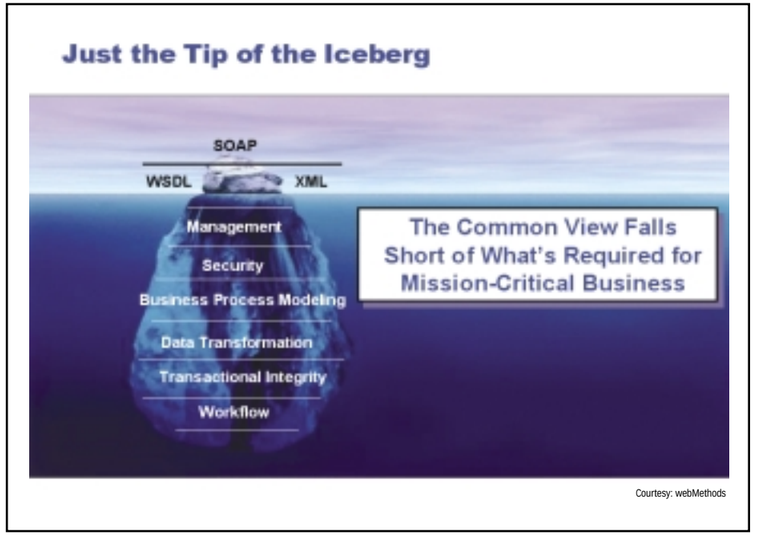


FIGURE 1 | **Web services as an iceberg**

Plugging the Gaps

While the Web services standards that have been accepted to date are an important step towards simplifying application connectivity, they address only a small portion of what is involved in true application integration. Using the image of an iceberg, Web services standards address a small number of issues above the water line (see Figure 1). The lower portion of the iceberg, hidden below the water, is far larger and ultimately more important.

Some of the issues not yet addressed by Web services standards include data transformation, adapters to other platforms, transactionality, business-process management, service monitoring, and security. Each of these issues merits a brief examination.

Data Transformation

While Web services standards make it easier to communicate with applications, they simply pass parameters back and forth in the same format and form. Integration platforms, on the other hand, have capabilities that automatically transform data to application- and machine-specific formats.

Adapters

Adapters are currently used to expose applications to the outside world. Web services are designed to do the same thing, and as standards are incorporated into major applications, adapters will become less necessary. But not all of the systems a company owns will be Web service enabled. In fact, the vast majority of organizations are populated with old mainframe systems, custom applications, and legacy databases that are not yet Web service compatible.

Transactionality

A company may create a Web service that allows its customers to change a "Ship To" address, but what happens if at the moment that information is being sent, a system failure occurs – a phone line goes down or a plug gets pulled. The customer has no way of knowing the data was not transferred. With integration platform technology, transaction controls guarantee the integrity of a transaction. There is no such provision in Web services standards.

Business Process Management

While work is certainly progressing in this area with initiatives such as BPEL4WS, provisions have not yet been built into Web services standards to manage business processes that span multiple applications and business partners, both behind the firewall and across the Internet.

Security

For communication inside or outside your company using a Web service, there is nothing built into the technology to protect against that communication being intercepted. Again, work is taking place in this area, but for now Web services standards do not address that area of security.

Service Monitoring

A company looking to use a Web service to run its business can't afford to be caught with an unreliable service. The way to protect against this today is by using sophisticated management technology to fix problems when they occur, and predict problems before they happen so issues can be resolved before they cause damage. At the moment, this is not addressed by Web services standards.

It's easy to see how all of these issues create the larger part of the iceberg, and can ultimately sink the ship. Consider the chal-

lenges involved in creating a Web service that leverages customer information. In most Global 2000 companies, customer information is stored in a wide range of systems, from mainframe and legacy systems and databases, to newer CRM and ERP systems. A company looking to build a Web service that accesses customer information might have to build up to a dozen or more connectors in order to tap into the complete customer record. As outlined earlier, there are a number of reasons why that can't be done with Web services standards today.

Vendors and members of the industry recognize that standards are just beginning to mature and are starting to address some of these issues, but we're not there yet. What that brings us to is the need for Web services–based integration platforms that support Web services fully, but also provide all the other elements that are not yet available through open standards mechanisms.

Extending Value

How do we get beyond mere interoperability, which is essentially what is provided by SOAP and WSDL, to true integration? Can we achieve an "Application Net," or the equivalent of a dial tone, where it might be possible to simply plug in and conduct business, just as we plug in for e-mail today? What are the barriers that need to be overcome to achieve true, ubiquitous integration? I see three fundamental areas to consider:

- ***Technical infrastructure***

For the most part, this is composed of the issues described above, including security, business process orchestration, and service monitoring. Excellent progress has been made in this area, including work on such standards as WS-Security, BPEL4WS, and OMI. What doesn't exist yet is a standard above these standards that is application centric at its core – in other words, a standard that defines the standards needed to have an enterprise class, trustworthy Web services-based infrastructure.

- ***Application semantics***

Every application, be it a custom application or a commercial application like SAP or Siebel, has a different way of representing the world. Interoperability does not imply semantic equality. In other words, if we solve the first barrier, technical infrastructure, we can get a dial tone. We know it's going to stay up, and we know other people cannot listen in on transmissions. But one person might still be speaking Russian and another Japanese. Numerous efforts have been undertaken over the years to achieve an overall standardized data model, but it hasn't happened yet, and it doesn't look like it will happen anytime soon.

Data models are the assets of the software vendors, and they understandably go out of their way to protect those assets. The future may very well lie in point-to-point collaborations between application vendors. But these collaborations are going to have to be standards based, or they'll simply be one-off collaborations.

- ***Implementation-specific differences***

Even if we solve the first two barriers, no packaged solution will ever be 100% complete. Customers will need to modify or customize the products they implement. Customization standards are needed that allow customers to modify or upgrade their platforms without having to rewrite collaborations that are built on the integration infrastructure. Over time, as applications are changed, the integrations and application collaborations will still work.

Until all three of these areas are addressed, complex one-off solutions will prevail.

The Task at Hand

The question, of course, is what can information technology managers do today to prepare for the future?

The first and most critical action is to get a service-oriented architecture into place. Web services are a means to integration, but the benefits cannot be realized if a services-based architecture has not been implemented in the first place. It sounds obvious, but in the rush to solve a specific business need, future technology considerations are often pushed to the side.

The second thing managers can do is start developing Web services–based integration components now, but at the same time remain pragmatic. Not everything can be, or should be, a Web service, particularly today when the standards are immature. On the other hand, it's clear that Web services are going to be the basis of integration in the future, and it will pay to begin laying the foundation.

The third action is to select vendors that lead and follow standards. The age of proprietary integration is over. As standards develop to address each of the underlying business issues, make sure your vendors offer support.

Companies are just beginning to experiment with Web services, developing prototypes and testing the technology in isolated corners of their business. The activity illustrates the general enthusiasm for Web services and the benefits it can afford corporations – from faster and less expensive integrations to the ability to reuse applications and the promise of widespread automation of business processes.

Rather than being viewed as a replacement for enterprise integration technologies, Web services are now correctly seen as a natural extension of integration technology. Companies will be able to achieve true enterprise class integration by using these new standards, while at the same time taking advantage of the strength, reliability, maturity, and scaleability that have already been incorporated into proven integration platforms.

It's true that Web services have not yet delivered to the degree that some have predicted or perhaps hoped, but that has to be considered within the context. The technology is simply following a similar development pattern to that taken by other important innovations. The fact is, the market is converging around the standards that have already been adopted, and work is proceeding on a number of fronts to address the barriers that exist today.

The next year could see important breakthroughs on a number of fronts, making this an exciting time indeed for the Web services community.

The Promise of Portable Business Processes

A look ahead to a robust standard for writing portable business processes

The Business Process Execution Language for Web Services (BPEL4WS or BPEL for short) is an XML-based programming language/execution environment intended to enable portable business process definitions for WSDL-based business processes. BPEL's goal is to make it possible to write a business process once in BPEL and then run it everywhere.

As a Turing complete language BPEL can do, well, anything. Although it limits itself to features necessary for business processes this still requires an enormous number of capabilities. BPEL currently provides:

- Two full programming models (graph and structured)
- An exception model
- Three types of storage
- A threading model
- A transaction model
- A process life cycle–management model
- A message-handling and routing model

That's a lot of functionality. This article provides an overview of BPEL. It will, therefore, be necessary to leave out a number of features and simplify a number of otherwise complex issues.

The Nature of Standards

As a new specification, BPEL has years of work ahead of it before it is sufficiently mature to be used for building portable implementations. In the meantime it is difficult to make categorical statements about BPEL as any one of its numerous open issues could cause significant changes in BPEL's functionality. As such, the reader is reminded that this article only represents the views and opinions of the author and not necessarily those of the other BPEL authors.

BPEL Execution Model: The Big Picture

The core of BPEL is the process. A BPEL process is just like any other process providing standard facilities such as storage, threads, faults, etc.

Commands given in a BPEL process are called activities and the definition of a BPEL process consists of exactly one activity. Therefore, interesting BPEL processes tend to start with either the Sequence or Flow activities, both of which are containers of other activities.

A Sequence activity contains a list of activities that are to be executed serially. A Flow activity contains a list of activities that are each to be executed in their own threads. Threads may be executed in parallel but this isn't guaranteed; the BPEL implementation may choose to serialize execution. Sequence and Flow can contain any arbitrary set of activities in any order, including other Sequence and Flow activities.

The control flow of a BPEL process is specified using an arbitrary combination of two programming models – structured and graph based.

Structured Programming

Structured programs are written using the Switch and While activities. Switch defines an ordered list of Boolean conditions, each associated with an activity. The first condition that evaluates to true has its activity executed. The While activity will loop and execute its associated activity until its Boolean expression evaluates to false.

Creative use of the Switch and While activities can create the computational equivalent of common structured constructs such as if-then-else, for-each, until, etc.

Graph-Based Programming

Graph-based programming is typically created with a visual programming tool that displays activities on a screen and allows those activities to be connected via links. Links have Boolean expressions assigned to them called transition conditions. Boolean expressions associated with each activity, called join conditions, decide if an activity should execute based on the value of the transition conditions of the links that point at it.

For example, a join condition on a destination activity could be written to specify that the activity should only execute if all the links that point to it evaluate to false.

BPEL supports the graph paradigm by providing for links on all activities. Any activity can be linked to any other activity. This works across threads, structures, etc.

The combination of links, transition conditions, and join conditions is computationally equivalent to structured programs.

Author Bio

Yaron Y. Goland is a principal technologist in the office of the CTO at BEA with responsibility for business process standards. He has spent the last eight years working on standards and related technologies. While at Microsoft Yaron was the program manager for network interfaces for IE 4, representative to the IETF HTTP working group, lead author on the WebDAV specification, and architect for UPnP's transport protocols.

YARON.GOLAND@BEA.COM

Boolean Expressions

Boolean expressions in BPEL are defined using XPATH 1.0 by default. BPEL provides a number of XPATH extension functions to enable access to information such as the value of a particular incoming link. However, the use of XPATH is a configurable option and can be overridden.

Containers & Assign

Containers are BPEL's answer to variables. They can only be defined at the global level and are globally visible. BPEL supports three types of containers – message containers, partner containers, and service reference containers. Partner containers will be discussed later; service reference containers will not be discussed in this article.

Message containers hold SOAP messages. They can either be created in a container by the BPEL process or they can be used to receive messages sent to the process. A message container's type is defined by a reference to a WSDL message definition.

The Assign activity is used to move XML between containers of the same type. XML fragments are moved between containers through the use of XPATH expressions with the Assign activity. The Assign activity can also move static XML into containers.

Scopes

BPEL uses scopes to manage faults and compensation handlers. A new scope is declared by using the Scope activity that then contains the various handlers and a single activity.

Faults

BPEL faults are a combination of a unique name and an optional container. A BPEL process can throw a number of predefined faults or faults can be thrown by the Throw activity.

A scope's fault handlers can catch faults based on the name of the fault, the type of container associated with the fault, or a combination of the two. Otherwise the BPEL fault handler provides typical features such as an optional "catch-all" fault handler, standard scope-based escalation for uncaught faults, and the ability of fault handlers to re-throw faults.

Compensation

Compensation handlers are a convenience mechanism that associates activities that "undo" the actions in a scope with the scope.

For example, if a scope causes a reservation to be made at a restaurant then the compensation handler could contain activities that would send a message to cancel the reservation.

Compensation handlers are activated by a call to the Compensate activity, which can only be called from inside the fault handler or

a compensation handler. When a compensation handler is run it must be shown the state of all containers as they were when the associated scope successfully exited. Any changes the compensation handler makes to the containers will be visible only to the compensation handler and will be discarded when the compensation handler exits.

The compensation handler's potentially unbounded memory requirements and isolated nature are open design issues that will be addressed in future versions of BPEL.

Subroutines

BPEL does not define an explicit subroutine or function call model. Rather, everything in BPEL is a Web service. Reusable functionality is packaged as a BPEL process which is then called as a Web service by other BPEL processes.

Message Processing

Messages are sent and received using the Receive, Reply, Pick, and Invoke activities. Messages are received by executing the Receive activity that takes a number of arguments, including port type, operation, partner container, correlations, and a message container.

Port type and operation refer to the definitions in an associated WSDL file. The partner container holds information about the Web service to which the message will be sent. The definition of partners is outside the scope of this article. Correlations are groups of XPATH expressions that are used to uniquely identify a message as part of a conversation. The message container is used to store the message when it arrives.

When the Receive activity is executed the associated thread will stop executing until the specified message is received. If the operation is a request/response then the Reply activity can be used to send back the response.

The Pick activity is used when one of a number of possible incoming messages may be sent. The Pick activity is the message equivalent of a Switch except that the conditions are Receives instead of Boolean expressions. Unlike the Receive activity, the Pick activity can have a timeout set on it.

The Invoke activity is used to send messages and, in the case of solicit/response, receive back a response. Invoke uses the same arguments as Receive but supports two message containers, one to hold the outgoing message and an optional second message container to handle the response. If Invoke should receive back a SOAP fault, that fault will be thrown as a BPEL fault with the SOAP message in the container.

Programming Language or Execution Environment?

BPEL's multiple programming models, terse syntax, and XML serialization do not lend themselves to direct programming. In other words, BPEL should be compared to the CLR or the JVM rather than to a programming language. It is easy to imagine writing a business process in some language/environment and then "compiling" the resulting program down into BPEL for execution.

BPEL and Java

BPEL is a procedural language for defining a message router that will enforce the choreography of a business process. But BPEL is not intended for generic data processing, UI generation, and so on. Therefore, it needs to be supplemented by Java, which is better suited for providing such behavior. At runtime BPEL will accept incoming messages and then route them to Java-backed Web services that will provide the heavy lifting. Java and BPEL form a team that enables the creation of complicated business processes in a platform-independent manner.

The Future of BPEL

The Web Services Choreography Interface (WSCI) provides a design and functionality that is almost identical to BPEL's. The most significant difference is that WSCI doesn't support graph-based programming.

It doesn't take a great leap of imagination to see WSCI and BPEL combining to form a single standard. I, and my company, believe that any official standard should be available on royalty-free terms through an open standards process.

There has been a great deal of discussion in the W3C Web Services Activity regarding the formation of a choreography working group that would take WSCI, BPEL, and possible other proposals as input. At the time of this writing, nothing final has been decided, though there certainly seems to be momentum toward this as the Web Services Architecture Working Group recommended standardization of choreography.

Conclusion

This article has only touched on the BPEL feature set at the highest possible level of abstraction. It discussed how to:

- Process activities in sequence or in parallel
- Program using a structured or graph-based paradigm
- Store and manipulate XML values in containers
- Define and nest scopes
- Define, throw, and catch faults
- Define and invoke compensation handlers
- Use Web services as sub-routines and
- Send and receive messages

I also touched on BPEL's similarity to the CLR/JVM, the relationship between BPL and Java, and BPEL's expected convergence with WSCI to form a single open, royalty-free standard.

Issues I did not discuss included BPEL's WSDL extensions (properties, property aliases, and service link types), properties in containers, serializable scopes, Wait activity, Empty activity, Terminate activity, partner definitions, service references, default compensation and fault handlers, start activities, and abstract processes.

In the years to come, as BPEL is more fully developed and standardized, the Web community can look forward to a robust standard for writing portable business process definitions.

References

- *BPEL4WS:* http://dev2dev.bea.com/techtrack/BPEL4WS.jsp
- *WSCI:* www.w3.org/TR/wsci

Acknowledgements

Thanks to Bill Cox, David Orchard, and Ed Cobb for their constructive comments.

Web Services and the SEMANTIC WEB

Making information accessible and usable

Written by Paul Cowles

As the Semantic Web gains momentum, it's important for Web service developers to keep abreast of its technologies and prepare for a change in their industry.

This article looks at how the Semantic Web applies to Web services, including what specifications are under development and how you might prepare for integration with future semantic applications.

What Is the Semantic Web and How Does It Impact Web Services?

The current World Wide Web presents large volumes of information in a format tailored for viewing by people. A person can surf from link to link, query search engines, or attempt to reach sites by domain name. While the pages retrieved are appealing to humans, to the software that processes the contents they are no more than a string of random characters.

Software programs cannot load a random document, Web page, or file and understand the contents of that document. While the software could make assumptions based on HTML or XML tags, a human programmer would have to get involved and determine the meaning, or semantics, of each tag. From a computer's perspective, the World Wide Web is a garbled mess. Luckily, there's a solution: the Semantic Web.

First envisioned by Tim Berners-Lee, the Semantic Web is complementary to the World Wide Web and consists of machine-usable information. This new Web is made possible by a set of standards being coordinated by the World Wide Web Consortium. As the Semantic Web gains momentum, an increased number of information resources will be just as useful to software agents as to humans. In other words, agents finally have the ability to achieve literacy on the Internet.

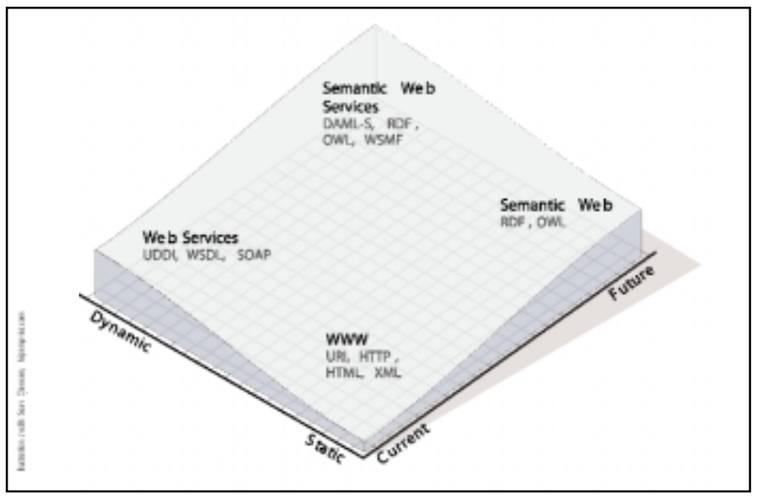


FIGURE 1 | **Evolution of the World Wide Web**

AUTHOR BIO:

Paul Cowles, VP of Business and Technology Development for Semaview, Inc., focuses on delivering products and services for businesses adopting semantic technologies. Also in charge of Semaview's partner program, Paul works with companies to identify and capitalize on new semantic business opportunities. Paul holds a B.Sc. in computer science from the University of Alberta.

PAUL@SEMAVIEW.COM

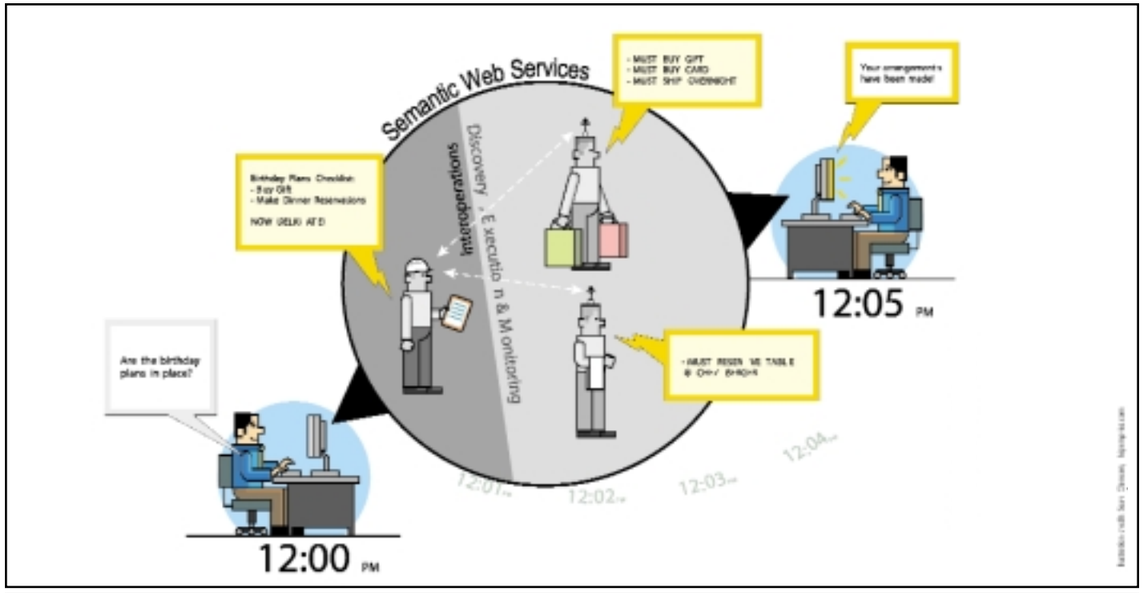


FIGURE 2 | **Harnessing the Semantic Web with personal agents**

Just as the Semantic Web is an extension of the World Wide Web, Semantic Web services are an extension of Web services (see Figure 1). At present, a programmer can build software with the ability to search ports or registries such as a UDDI server for a list of available Web services. While the software may be able to find a Web service without human intervention, it has no way to interpret how to use it or even what the purpose of the service is. Web Service Description Language (WSDL) provides a description of how a Web service will communicate, whereas Semantic markup provides details about what a service provides and why.

For Semantic Web services to become a reality, a markup language must be descriptive enough that a computer can automatically determine meaning. Following is a list of tasks such a language would require:

I Need to Find a Service (Discovery)

A program must first be able to automatically find, or discover, an appropriate Web service. Neither WSDL nor UDDI allows for software to determine what a Web service offers to the client. A Semantic Web service describes its properties and capabilities so that software can automatically determine its purpose.

I Need to Run a Service (Invocation)

Software must be able to automatically determine how to invoke or execute the service. For example, if executing the service is a multistep procedure, the software needs to know how to interact with the service to complete the necessary sequence. A Semantic Web service provides a descriptive list of what an agent needs to be able to do to execute and fulfill the service. This includes what the inputs and outputs of the service are.

I Need to Use a Few Services Together (Composition)

Software must be able to select and combine a number of Web services to complete a certain objective. The services have to interoperate with each other seamlessly so that the combined results are a valid solution. In this way, agent software can create entirely new services by combining existing ones on the Web.

I Need to Know What is Happening After I Run the Service (Monitoring)

Agent software needs to be able to verify the service properties as well as monitor its execution. Some services may take some time to complete, and the agent should have the ability to monitor the status of a process.

By providing agents with the ability to discover, invoke, compose, and monitor a service automatically without human interaction, powerful new applications can be built. Imagine an Integrated Developer Environment (IDE) that not only lists available services, but also suggests suitable combinations that satisfy your high-level requirements. Instead of scrolling through a list of services looking for one that has input parameters matching your application, the IDE provides you with only the services that are truly compatible.

Personal agents could be built to harness the Web for end users. For example, a personal agent could make arrangements for a birthday celebration with minimal input from the user (see Figure 2). An agent could combine reservation, shopping, and shipping services automatically to solve the higher-level problem of arranging the celebration. By doing this automatically, the user saves both time and money.

The Semantic Web consists of semantically marked up static and dynamic content resources. Semantic Web services are an integral part of the powerful machine-friendly Internet.

What Is the Current State of the Semantic Web?

The crucial specifications needed to build the Semantic Web are starting to come

together at the W3C. RSS 1.0 (RDF Site Summary), based on the Resource Description Framework (RDF) specification, has been in use for sharing news and events for several years. The Web Ontology Language (OWL), derived from the DAML+OIL language, is in development and working drafts have been published.

In parallel with the development of standards, applications of the technology are being released. Tools exist for creating and editing RDF, storing the semantic data, inferencing knowledge from the metadata, and visualizing the metadata. The Semantic Web has the backing of large research and development laboratories such as Nokia, BT, and Hewlett-Packard.

What About Semantic Web Services?

The standardization of Semantic Web services is following closely on the heels of the Semantic Web in general. Unfortunately, this means that Semantic Web service development is even more immature than the Semantic Web in general. Researchers and developers have proposed multiple specifications for the creation and deployment of semantically intelligent Web services (see Figure 3). Which specification will become the standard is a question that still remains unanswered at the time of this article.

RDF is the primary foundation for the Semantic Web. As such, some developers have worked toward Semantic Web services based on RDF and RDF Schema specifications. Some services, such as MusicBrainz (www.musicbrainz.org), already provide a Semantic Web service API based on RDF. While the MusicBrainz services don't satisfy all of the desirable traits of a Semantic Web service listed above, the available API is a substantial step toward the adoption of Semantic Web services.

Another specification receiving attention is the DARPA Agent Markup Language (DAML) and its Web services ontology termed DAML-S. Developed by a coalition of researchers brought together by the DARPA program, DAML provides enhanced constructs in comparison to RDF.

DAML allows for advanced properties and property traits such as equivalence and uniqueness. DAML is much more expressive than RDF, its lightweight predecessor. A specific application of DAML, DAML-S, supplies the necessary standards to create a Semantic Web service. Whereas WSDL can tell an agent how the service works, DAML-S can help describe what the service offers and why. In this way, it complements WSDL by providing additional information needed by agents to fullfill the tasks such as discovery and execution.

Using DAML as its basis, the W3C has commissioned the Web Ontology Working Group to produce an agreed-on standard for a Web ontology language. The OWL Web Ontology Language is the result and is currently in the working draft stage of development.

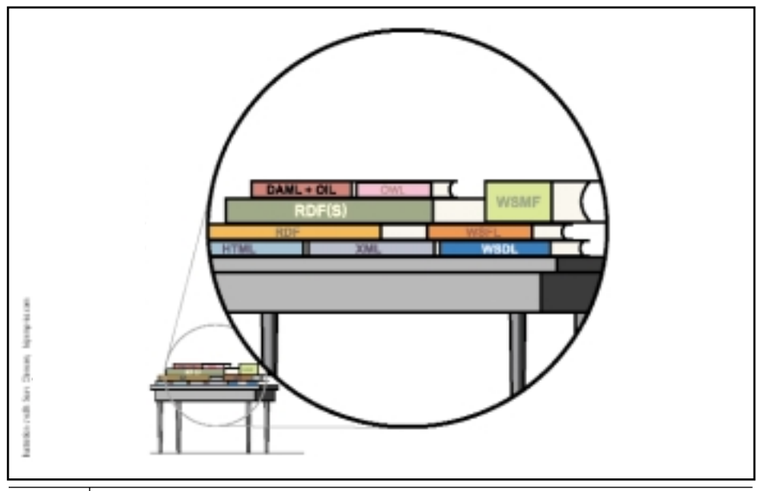


FIGURE 3 | **Semantic Web services: layers of language development**

Additional research and development has focused on another framework for building Semantic Web services; the Web Services Modeling Framework (WSMF) has been proposed to facilitate the creation of Semantic Web services. It is based in part on IBM's Web Services Flow Language (WSFL), an XML language for the description of business process–driven compositions of multiple Web services.

To date, the DAML-S specification has received the most attention in the academic and research communities. It is likely, however, that an equivalent ontology will be built using the OWL specifications and subsequently become the de facto standard for creating Semantic Web services. Presently, developers are not going to be able to leverage Semantic Web services for any practical application. Publicly accessible Semantic Web services will, however, begin to appear soon.

How Can We Prepare for Semantic Web Services?

Few developers today are actively building Semantic Web services. As the standards are completed and the demand for intelligent agents grows, expertise will be required in the field of Semantic Web services. Keeping informed, adopting early prototypes, and anticipating change is essential to stay ahead of the curve. Listed below are some recommended actions:

Stay Current

Monitoring key Web sites is the easiest and most cost-efficient way of keeping current as the Semantic Web progresses. The World Wide Web Consortium (www.w3.org/2001/sw) coordinates all formal Semantic Web specifications. Alternatively, the DARPA Agent Markup Language Homepage (www.daml.org) contains use cases, tools, ontologies, and other useful resources for creating semantic markup.

Many of the various working group members attend annual conferences dedicated at least in part to the Semantic Web. The Twelfth International World Wide Web Conference (www.www2003.org), held in Budapest, promises to have papers, talks, and workshops focused on Semantic Web services. The Second International Semantic Web Conference (http://iswc2003.semanticweb.org) is the other major Semantic Web conference

Continued on page 57 ▶

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES ONLINE

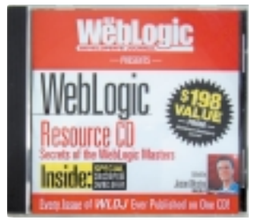
Go To www.sys-con.com/suboffer.cfm

and receive your **FREE** CD Gift Package VIA Priority Mail

Each CD is an invaluable developer resource packed with important articles and useful source code!

More than 1,400 Web services and Java articles on one CD! Edited by well-known editors-in-chief Sean Rhody and Alan Williamson, these articles are organized into more than 50 chapters on UDDI, distributed computing, e-business, applets, SOAP, and many other topics. Plus, editorials, interviews, tips and techniques!
LIST PRICE $198

The most complete library of exclusive *JDJ* articles compiled on one CD! Assembled by *JDJ* Editor-in-Chief Alan Williamson, more than 1,400 exclusive articles are organized into over 50 chapters, including fundamentals, applets, advanced Java topics, Swing, security, wireless Java,... and much more!
LIST PRICE $198

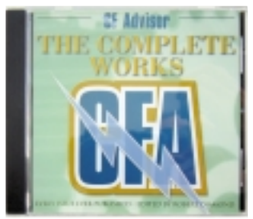


The most complete library of exclusive *WLDJ* articles ever assembled! More than 200 articles provide invaluable information on "everything WebLogic", including WebLogic Server, WebLogic Portal, WebLogic Platform, WebLogic Workshop, Web services, security, migration, integration, performance, training...
LIST PRICE $198

The most complete library of exclusive *CFDJ* articles on one CD! This CD, edited by *CFDJ* editor-in-chief Robert Diamond, is organized into more than 30 chapters with more than 400 exclusive articles on CF applications, custom tags, database, e-commerce, Spectra, enterprise CF, error handling, WDDX... and more!
LIST PRICE $198

The largest and most complete library of exclusive *XML-Journal* articles compiled on one CD! Edited by well-known editors-in-chief Ajit Sagar and John Evdemon, these articles are organized into more than 30 chapters containing more than 1,150 articles on Java & XML, XML & XSLT, <e-BizML>, data transition...and more!
LIST PRICE $198

The most up-to-date collection of exclusive *WSDJ* articles! More than 200 articles offer insights into all areas of WebSphere, including Portal, components, integration, tools, hardware, management, sites, wireless programming, best practices, migration...
LIST PRICE $198



The complete works of *CF Advisor* are now on one CD! Check out over 200 exclusive articles on topics such as e-commerce, custom tags, Fusebox, databases, object-oriented CF, WDS, upgrading CF, wireless, Verity, and more. Plus, find interviews, editorials, and source code!
LIST PRICE $198

Pick the CDs to go with your Multi-Pack order

- Pick one CD with your 3-Pack order
- Pick two CDs with your 6-Pack order
- Pick three CDs with your 9-Pack order

- ☐ Web Services Resource CD
- ☐ Java Resource CD
- ☐ WebLogic Resource CD
- ☐ ColdFusion Resource CD
- ☐ XML Resource CD
- ☐ WebSphere Resource CD
- ☐ CF Advisor Complete Works CD

Your order will be processed the same day!

AND SAVE UP TO $400 AND RECEIVE UP TO 3 FREE CDs!

Pick a 3-Pack, a 6-Pack or a 9-Pack ☑

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

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

☐ **Java Developer's Journal**

U.S. - Two Years (24) Cover: $144	You Pay: $89 /	Save: $55 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $49.99 /	Save: $22
Can/Mex - Two Years (24) $168	You Pay: $119.99 /	Save: $48 + FREE $198 CD
Can/Mex - One Year (12) $84	You Pay: $79.99 /	Save: $4
Int'l - Two Years (24) $216	You Pay: $176 /	Save: $40 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **Web Services Journal**

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

☐ **.NET Developer's Journal**

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

☐ **XML-Journal**

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

☐ **WebLogic Developer's Journal**

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

☐ **ColdFusion Developer's Journal**

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

☐ **Wireless Business & Technology**

U.S.- Two Years (24) Cover: $144	You Pay: $89 /	Save: $55 + FREE $198 CD
U.S. - One Year (12) Cover: $72	You Pay: $49.99 /	Save: $22
Can/Mex - Two Years (24) $192	You Pay: $139 /	Save: $53 + FREE $198 CD
Can/Mex - One Year (12) $96	You Pay: $79.99 /	Save: $16
Int'l - Two Years (24) $216	You Pay: $170 /	Save: $46 + FREE $198 CD
Int'l - One Year (12) $108	You Pay: $99.99 /	Save: $8

☐ **WebSphere Developer's Journal**

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

☐ **PowerBuilder Developer's Journal**

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

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS

3-Pack

Pick any 3 of our magazines and save up to $275.00
Pay only $175 for a 1 year subscription plus a **FREE CD**

- 2 Year – $299.00
- Can/Mex – $245.00
- All Other Cnty. – $315.00

6-Pack

Pick any 6 of our magazines and save up to $350.00
Pay only $395 for a 1 year subscription plus **2 FREE CDs**

- 2 Year – $669.00
- Can/Mex – $555.00
- All Other Cnty. – $710.00

9-Pack

Pick all 9 of our magazines and save up to $400.00
Pay only $400 for a 1 year subscription plus **3 FREE CDs**

- 2 Year – $839.00
- Can/Mex – $695.00
- All Other Cnty. – $890.00

Service Discovery: Perspectives on the Past, Present, & Future

A talk with Radovan Janecek and Peter Alesso

AUTHOR BIO:

Michael Sick is an independent Java architect helping clients solve complex product definition and design problems. He has more than eight years of experience in the construction of distributed information systems and Internet technology, holding positions including architect and VP of development. He holds undergraduate degrees in both geology and political science from Guilford College.

MIKE_A_SICK@YAHOO.COM

Service discovery and identification is a critical piece of the overall Web services puzzle. Existing service architectures, such as CORBA, JINI, and ebXML, have all devoted significant effort towards making service discovery available and comprehensive. If Web services is to become a ubiquitous platform for service oriented computing, then the industry will need to create standards and products that enable sophisticated discovery, identification, and composition of Web services.

This month, Radovan Janecek and Peter Alesso join *Web Services Journal* in exploring the state of service discovery today. Radovan leads development at Systinet, creating industry-leading implementations of Web services standards (UDDI, SOAP, WSDL) as part of an overall Web services platform. Peter joins us from WebIQ and Lawrence Livermore National Laboratory. He is currently engaged in a broad range of research related to software services, wireless computing, and artificial intelligence. (As a brief aside, I'd like to recommend Peter's book, *The Intelligent Wireless Web*, to all those interested in a broad and integrative view of these topics.) With both academic and commercial experts as guides, let's take a brief look at the past, present, and future of Web services and discovery.

To give us a baseline, could you both define service discovery and its major components?

Radovan: I view service discovery as the ability to provide a service identifier according to some specified metadata. What are the major components? It's hard to say since it always depends on discovery usage. My vote is that service discovery is composed of a semantic processor, a registry, a query processor, and a service locator. Additionally, I believe that future discovery services will provide semantics-based discovery.

Peter: Currently there are several methods of discovering Web services – from manual to automated. At the manual end of the spectrum, techniques include e-mail, Web browsing, phone calls, and word-of-mouth. Next, consider publishing services and service descriptions on various Web sites via formatted files, such as Microsoft's DISCO and IBM's ADS. These can be used to create standardized file formats and look up definitions.

A more sophisticated approach to discovery is that of Web services brokers (SalCentral, XMethods, and Silicon Hills Group) who actively manage services for service providers by providing a means to publish, promote, sell, test, and support their Web services. The final approach to Web service discovery is automated and dynamic service registries, such as UDDI. Like service brokers, the UDDI registry provides a Web page interface. But UDDI registries also provide application programming interfaces (APIs) for publishing, finding, and binding services. This makes UDDI registries more dynamic and complex than any other type of Web service discovery method.

Our Web Services Experts

Internet innovator, engineer, and author, H. Peter Alesso has led successful research and development programs at Lawrence Livermore National Laboratory (LLNL) for over 20 years. At LLNL, he led a team of computational physicists and engineers in a wide range of successful multimillion-dollar software development research projects ranging from software applications for robotics to parallel processing of supercomputers and clustered networks. Peter holds an M.S. and advanced engineering degree from MIT, and has published over 30 journal and conference papers, as well as two books: *e-Video: Producing Internet Video as Broadband Technologies Converge* (Addison-Wesley, 2000); and *The Intelligent Wireless Web* (Addison-Wesley, 2001).

Radovan Janecek is the VP of Engineering at Systinet. Prior to joining Systinet in 2000, he worked for the IBM T.J. Watson Research Institute, where his research and development focused on software-based speech processing engines, and multimodal control of software applications. He also worked for Borland/Inprise as a CORBA and Java consultant. Radovan was a Ph.D. candidate and served on the faculty of Mathematics and Physics at Charles University in Prague, where he was a member of the CORBA and Distributed Systems Research Group. He holds a master's degree in mathematics and informatics from Palacky University in Olomouc, Czech Republic.

Please describe your academic, research, and/or commercial history as it relates to Web services.

Radovan: I have been leading development of WASP UDDI at Systinet for two years. From the beginning we had our own vision of UDDI called the "UDDI Grid" – a federated registry of UDDI nodes with much better security support than the UDDI standard requires. As a vendor of a general Web services infrastructure, we always paid a lot of attention to the integration of UDDI with Web services. This led to the publishing of a technical paper on how to publish WSDL to UDDI.

During my Ph.D. studies, I was a member of CORBA and Distributed Systems Research Group (http://nenya.ms.mff.cuni.cz), where I worked on dynamic updates of distributed components. Discovery was also an important part of that research.

Peter: As a software researcher at www.Web-iq.com and Lawrence Livermore National Laboratory, I have been investigating Web service applications for the next generation Web architecture – the Semantic Web. This has involved three main functions: service discovery, execution, and composition and interoperation. Also, wireless Semantic Web services are of particular interest to me.

I'd like to get your 50,000-foot overview of the state of service discovery today.

Radovan: Today many discovery services are in fact only location services. One system to study is the ebXML Registry/Repository that is a part of ebXML technology and is designed to be used for miscellaneous discovery purposes. It has a generic data model and multiple APIs allowing both queries and storage of the data. UDDI, on the other hand, is a global lightweight registry used as discovery service with a strictly defined data model and API. In many cases, people are satisfied by more simple solutions based on DISCO or WS-Inspection. A critical problem of discovery services is how to register with them. Possibly the main reason that powerful discovery services are not widely used is the difficulty of the registration process.

Peter: UDDI is not the service registry itself. Instead, it provides a standard mechanism for publishing and discovering services within the registry. It is not currently an official standard, but a specification involving a collaboration of over 300 companies who formed to become the UDDI project. This project is responsible for running a UDDI business registry, which for UDDI Version 1.0 actually consists of two separate registries. Each UDDI registry has a business as an operator, which is similar to the companies that operate the service repositories for service brokerages. Currently, the two UDDI Version 1.0 registry operators are IBM and Microsoft. The agreement is that any information entered in one registry becomes replicated in all other registries as well. Version 2.0 of

UDDI has recently been released. From a technical standpoint, UDDI provides structured data and a defined way of integrating with the structured data. The UDDI project's stated intent is to have the registry support services integrate with a wide set of existing core Internet technologies, such as Java, Jini, CORBA, ebXML, RMI, and HTML.

In your opinion, what is the most advanced service discovery system in place today? Please feel free to use any example that comes to mind and don't limit your answer to traditional software systems.

Radovan: Certainly Google! (Laughs) Practically speaking, more services are discovered this way than any other method.

Peter: UDDI 2.0. The UDDI 2.0 specifications were published on Monday June 18, 2002, at uddi.org, a major version update for the Universal Description, Discovery and Integration protocol.

What are the strengths and weaknesses of UDDI? And, is it the right platform for a long-term service discovery strategy?

Radovan: I see the main strengths of UDDI as its broad vendor support and its simplicity. The weakest point of UDDI is the lack of repository abstraction – UDDI is an index only. Another weakness of UDDI is that it has no explicit support for Web services. Long term, I wouldn't be surprised to see UDDI merge into the ebXML registry. Overall, I have very high hopes for the future of UDDI under OASIS stewardship.

Peter: If you're asking if UDDI will succeed as the Web service description and discovery standard, I can only say that since Web services infrastructure is in its infancy, this question is at the very heart of the success of this new technology.

So far, business adoption of UDDI 1.0 has been tentative and has not yet fulfilled its potential. UDDI 2.0 seems about to reinforce UDDI's position as a global yellow pages, rather than upgrading its capabilities as a Web services directory.

However, the UDDI 2.0 registry operators will share data and synchronize their databases. This trend toward broader cooperation and building a future infrastructure of many UDDI registries is positive.

Why do you believe that the usage of service discovery, specifically with Web services, has been anemic to date?

Radovan: I wouldn't say anemic, but it depends on exactly what you are asking. I agree that the originally intended use of public UDDI is anemic. On the other hand, I see lots of activity with hundreds of companies using and testing UDDI in a private capacity. As for the public use of UDDI, I think the reason it is lagging is exactly the same as in the case of the CORBA Trading Service in the CORBA world: low demand.

In both cases, there is nearly nothing to discover. If you look at the biggest public registry of Web services, XMethods, you will find up to 300 services. You really don't need UDDI for such an amount of mostly anonymous and unreliable Web services. But any large company, such as a JP Morgan or General Motors, has thousands of services internally and they are usually disorganized. There is the potential for UDDI to make a difference and solve problems.

Peter: I'm unable to provide a clear commercial picture of the current state of usage of either service discovery, in general, or UDDI in particular. Both are the focus of serious growth and ongoing development, but at this early stage, difficulties in implementing and validating new services, as well as limitations due to competitive tradeoffs, make the situation somewhat difficult. Software architects will still have to spend considerable time evaluating and validating tradeoffs before making commitments to Web service vendors.

What will the next generation of discovery services look like and what are the biggest challenges to solve to make them work?

Radovan: The next generation will center on semantic-based discovery as a backbone for service management systems as well as services runtimes. Balancing reasonable registration processes against creating powerful semantics will be difficult.

Peter: The successful development of Web services will greatly depend on the ability to automate as much of the Web services process as possible under interoperable standards. Matchmaking, or service discovery, is one important aspect of Web services' e-commerce interactions and advanced matchmaking services require rich and flexible metadata that are not supported by current industry standards, such as UDDI and ebXML.

The most advanced vision for the next generation of Web services is the development of Web services over Semantic Web architecture. The Semantic Web initiative at W3C is generating technologies and tools that could fulfill the requirement for advanced matchmaking services. Support of Semantic Web–related technologies includes tools such as RDF (Resource Description Framework) and DAML-S.

This provides support for automated Web service composition and interoperability. Integral to this effort will be three main functions: service discovery, execution, and composition and interoperation. Of these three tasks, none is entirely realizable with today's Web, primarily because of the lack of a suitable markup language. Academic research on Web service discovery is growing out of agent matchmaking research. Recent industrial efforts have focused primarily on improving Web service discovery and aspects of service execution through initiatives, such as UDDI and ebXML.

And finally, put on your oracle hat and list three standards that you believe are the most critical to the advancement of Web services and discovery.

Radovan: Well, I'd say: a) RDF or a similar semantics-supporting standard and a corresponding query language; b) REST as a means for resources location and modification; and c) security, security, and more security.

Peter: The success of Web services depends on the ongoing development of open standards that ensure interoperability between different implementations. The standardization efforts for the technologies that underlie Web services include SOAP, WSDL, UDDI, ebXML, RosettaNet, BTP, and S2ML.

A Fight to the Finish for Business Process Standards?

The biggest challenge will be in giving control to the businesses themselves

Just as rules are made to be broken, it sometimes appears that standards are made to proliferate. On the face of it, this may seem to be the case in the emerging battle over business process standards for Web services.

BEA Systems, Microsoft, and IBM introduced their Business Process Execution Language for Web Services (BPEL4WS) with much fanfare during the summer of 2002. In doing so, they added another clumsy acronym to what is becoming a battle to define the way Web services are combined and deployed to accomplish business tasks. BPEL4WS is just one of a number of emergent standards in the general area of Business Process Management, also known as BPM. Others include:

- ***WSCI:*** Web Services Choreography Interface
- ***BPML:*** Business Process Modelling Language
- ***XPDL:*** XML Processing Description Language
- ***BPSS:*** Business Process Specification Schema
- ***EDOC:*** Enterprise Distributed Object Computing

A Crowded Marketplace?

The congestion in the market for standards shows that, despite the splash made by Microsoft and IBM, there is still considerable resistance from competing organizations. The Business Process Management Initiative (BPMI.org) touts BPML, with backing from BEA Systems (who also support BPEL4WS), Sun Microsystems, and SAP, among others. The Workflow Management Coalition (WfMC) offers XPDL. Even the United Nations is involved, standing behind BPSS, which is part of the larger Electronic Business XML (ebXML) initiative run by OASIS.

Of the competing standards, BPML is the most direct competitor for BPEL4WS, especially in the Web services arena. Both BPML and BPEL4WS actually form parts of rival standards' "stacks," with the familiar SOAP, WSDL, and UDDI standards as their common underpinning. Where BPEL4WS uses the (simultaneously announced) WS-Coordination and WS-Transaction standards to provide support for simple and extended transactions through Web services, BPMI endorses the WSCI standard as its Web services underpinning.

This combination and consolidation of standards is natural. BPEL4WS itself is the result of a merging of IBM's WSFL and Microsoft's XLANG. Time will tell if such a marriage of rival standards can last, but history suggests it can: the defining protocol of Web services – Simple Object Access Protocol (SOAP) – originated as a joint submission to the World Wide Web Consortium (W3C), by both IBM and Microsoft.

Decree or De Facto Standards?

With so much competition, it is clear that the struggle for standards is far from over. In fact, research from Gartner released in August suggests that the development of Web services standards won't be complete by 2007.

At the same time, enterprises are ready to begin Web services deployment today. They want to experience the decreased costs and increased integration promised by Web services and are anxiously looking for any clue as to how this battle will turn out.

This may eventually come down to a decision by the W3C. Oracle Corp. has already asked them to intervene on the issue. There have been rumors that IBM and Microsoft will submit BPEL4WS alongside the next version of the Universal Description, Discovery and Integration (UDDI) standard.

However, a decision from the W3C is not likely to come any time soon. It may take a long time for the organization to get anything to a recommendation stage. In the meantime, the hands of developers would be tied as they awaited a decision.

With that in mind, the question may be settled more immediately through the emergence of a de facto standard. Enterprises may be anxiously awaiting a formal standard, but they will not wait forever. They will soon start developing with their standard of choice. Once one standard emerges as a clear favorite among users, it will quickly outdistance its rivals. Thus, a de facto standard could emerge prior to a formal announcement from a regulatory body, and render any such decision unnecessary.

In the struggle to become the de facto standard, BPEL4WS appears to be the front runner, with Vitria and webMethods already signed up. Microsoft and IBM have a history of success in developing Web services standards. IBM already has, in Alpha, tools to build and implement processes in the language, and Microsoft can safely be expected to follow soon.

Author Bio

Steve Brown is chief technology officer of Metastorm Inc. (www.metastorm.com). Metastorm is the leading provider of Business Process Management (BPM) software to government and commercial organizations. SBROWN@METASTORM.COM

However, BPML and WSCI, whose combination presents the one direct challenger to BPEL4WS in the Web services arena, have been around a (very) little longer, and are likely to find champions among those who use neither IBM nor Microsoft. With the endorsement of BPMI.org and Sun Microsystems, the support base is very large. BPMI.org is also courting supporters of BPEL4WS by accepting the language, but positioning it as a subset of BPML (WfMC has taken the same approach with XPDL).

In both the short and the long term, the one criterion that will have the greatest effect upon the adoption of any of these standards is the availability of tools to build, deploy, and manage real systems that make use of them.

The final result, whether by decree or popularity, is still highly uncertain – which may explain why companies like BEA Systems are hedging their bets. In backing both BPEL4WS and BPML/WSCI, BEA assures itself of coming out on the right side no matter what happens. Alternatively, many intelligent companies are achieving the same goal by keeping a close eye on the struggle, while maintaining a firmly agnostic stance. This will keep them well positioned to act once a true standard emerges – though perhaps at the cost of any first mover advantage.

Beyond Web Services

Several of the proposed business process standards are not targeted at Web services. For these non–Web services BPM standards, each of the offerings will gravitate towards the areas that hold the most opportunity to exploit their advantages.

Large organizations looking to build secure, high-end B2B inter-enterprise integration in the EDI mold will look at BPSS (through ebXML) and EDOC. But it is at the lower end – the ability to orchestrate internal processes – that the real growth will be found. Here, BPML and BPEL4WS will battle for acceptance in the traditional EAI and BPM areas. XPDL is almost a non-factor, especially if WfMC gives up the fight and backs one of the other standards.

At the same time, the ability of BPEL4WS to address both Web services orchestration and traditional BPM and EAI requirements should give concern to some of these remaining players. For now, they have been content to stick to their specific niches and to leave Web services orchestration to a separate standard. However, IBM and Microsoft clearly see BPEL4WS as capable of expanding to handle both EAI and B2B processes. Proponents of niche standards may not have their market segments to themselves anymore.

> **"The major candidates for BPM standards today have one thing in common: all are XML-based"**

Time and again, the industry has chosen simple de facto standards over complex formal standards. TCP/IP was embraced; the Open Systems Integration (OSI) seven-layer communication model has largely fallen by the wayside. The same fate may one day be in store for BPSS, if and when BPEL4WS sets its sights on the high-end integration market.

Does It Matter?

The major candidates for BPM standards today have one thing in common: all are XML-based. Supporters of BPML and XPDL claim they are supersets of BPEL4WS. Therefore it should, in principle, be possible to transform a definition written in any one of these into one written in the other, using XSL Transformations (XSLT). Such a translation would lose any meaning that is not available in the target language, but to some extent the choice of standards ought not to matter too much.

But in reality, it does matter. The longer developers are left with a confusing array of acronyms and competing standards, the more difficult life becomes for vendors and customers alike.

Today, both clients and suppliers are using Web services predominantly as "intra-enterprise glue." Organizations can now see their way to linking all their legacy systems without incurring the expense of traditional EAI systems, or hand-coded solutions.

The real beauty of Web services is not only in how easy it is to call (or "consume") an existing Web service. It lies in how easy it is to generate (or "produce") a Web service interface on top of an existing application programming interface (API), whether DCOM/COM+, CORBA/IIOP, RMI, EJB, or traditional C or C++ native interfaces.

With the promise of Web services beginning to be realized, there is a danger of killing the momentum by not putting the best standards in place to facilitate the move from intra-enterprise deployments to inter-enterprise transactions. Standards development is necessary if we are going to realize the true promise of Web services.

Whether settled by the W3C, some other regulatory body, or the establishment of a de facto standard, it is incumbent upon those involved in the development of standards to consider the potential of the technology and the impact of the standard. There is a delicate balance between innovation and regulation. Reconciling the benefits of pushing the development envelope through innovation with the necessary establishment of standards will be one of the biggest challenges to the acceptance of a true standard for business processes.

But the biggest challenge of all will be in the provision of non-technical tools, based on the standard, which swing control of business processes away from technical departments and towards the business process owners themselves.

Written by Thomas Robbins

Web Services Made Easy

Integrate and enhance your existing Web sites with a Web service

A couple of weeks ago, while I was on my way home, my cell phone rang and I was greeted by one of my favorite customers, who sounded like he had had better days. He had just left a meeting with the CIO and received his annual development budget for the following year. The problem was that the CIO was unable to justify a new set of Web service initiatives around a set of just-completed internal Web sites.

AUTHOR BIO:

Thomas Robbins is a senior technology specialist with Microsoft in New England. He focuses on .NET development and implementing XML-based solutions. Thom is a regular speaker, writer, and presenter at industry events.
TROBBINS@MICROSOFT.COM

He and the upper management felt that it was too early to redevelop these sites. After all, as he explained, "the users had just been trained and were just starting to take advantage of these sites." It certainly wasn't that they didn't see the clear business and technical advantages of Web services but the business value just wasn't there. "Until we can get some return on our investments for these sites, they will stay as they are," was how the CIO later phrased it to me.

During our conversation I started to realize that we all too often forget how important it is to leverage existing assets in infrastructure and technology – and that we can use a combination of Web services and the .NET Framework to realize that. As I did for this customer, I will demonstrate how, using the built-in HTML parsing solution within .NET, you can parse existing content from a remote HTML page and then programmatically expose the resulting data in a Web service.

The development of a Web service that parses content is actually a different paradigm than traditional ASP.NET Web service development. At the core of this development is a service implemented through a Web Service Description Language (WSDL) file. The real difference is that with traditional ASP.NET development we never worry about actual WSDL generation. The framework handles this during the compilation process. With a parser-based service we actually spend our time focused almost exclusively on the creation of the WSDL. Once the WSDL file is created, then the framework provides a utility to generate the proxy class for our code. The trick, as I will show next, is that additional XML elements are added to specify both the input parameters and data returned from a parsed page. Even though additional elements are added, the end XML document must still adhere to the WSDL specification (www.w3.org/TR/wsdl). Within the WSDL file you provide both a target and a regular expression syntax to retrieve the requested parsed data. Once you have created the WSDL file, the .NET Framework provides a custom utility (wsdl.exe) that is used to generate the proxy files for ASP.NET applications. The built-in support is important to allow companies like my customer's to easily transition their existing investments in Web sites into Web services.

To demonstrate this technique, I created a simple HTML page that I will render back into a Web service callable by an ASP.NET application. There are two main caveats that I wanted to pass on. First, always make sure that you get proper permission before trying this on a site. Second, always remember that any changes to the layout of the target Web pages will cause problems within the Web service. In this article, I will show how you can retrieve both the <TITLE> and the <H1> elements of this simple document.

```
<html>
 <head>
  <TITLE>Sample Title</TITLE>
 </head>
 <Body>
 <H1>Some Heading</H1>
 </Body>
</html>
```

Creating Custom WSDL

I always like to think of WSDL as an XML format that describes the network services offered by a server. WSDL by definition is an XML-based file that identifies the services provided by the server and the set of operations within each service that the server supports.

Each operation described within the WSDL file includes a format that the client must follow to request an operation. The nature of this document sets up a requirement that both the server and the client must follow, and acts as a form of contract that both sides agree upon. The server limits its liability to only providing services if the client sends the properly formatted SOAP request. With a parsing service, both the parsing and implementation requirements are part of the WSDL document and these two combined return the requested information.

Within Visual Studio .NET, creating a custom WSDL file is fairly easy but not completely straightforward. The problem is that VS.NET doesn't directly support the creation of a WSDL file as part of its standard wizards. In order to add a WSDL file after creating an ASP.NET application, add a text file and then rename it with a *.WSDL extension. Once this is done you're ready to add the necessary XML elements.

Within the WSDL file there are a couple of basic elements. First is the <services> element. A service is a set of <port> elements that associate the physical or URL location with a <binding> element. Even though this is a one-to-one relationship, you can specify additional <port> elements within a <binding>. These are used for alternate locations. It really isn't uncommon to have multiple <service> elements within a document. This provides a couple of features, including the ability to group HTTP ports in one service and SMTP in another. This gives client applications the ability to search for the specific <service> elements they need. This also provides a built-in redirection mechanism for clients. Client applications can redirect requests to another <service> element and continue processing without any changes. For our sample I created a <service> binding that points to the local machine. Obviously, within the production application you would need to reset the URL to a valid location.

```
<service name="GetTitle">
    <port name="GetTitleHttpGet"
      binding="s0:GetTitleHttpGet">
      <http:address location="http:
//localhost/WebInfo" />
    </port>
  </service>
```

Within a WSDL document the <service> "name" attribute is used to uniquely distinguish one server from another. This becomes even more important when you have multiple ports in a service. The name attribute allows each one to become unique and distinguishable from the others.

TABLE 1: Match XML Elements

Attribute	Description
Name	Describes the class or property name of the returned data. The .NET Framework associates this with a class.
Pattern	The regular expression pattern used to apply against the parsed page and obtain the requested data.
Ignorecase	Determines whether the pattern expression is run case insensitive. By default all patterns are case sensitive.
Repeats	Provides the number of values that should be returned from the expression. The default is all matches (-1).
Group	Specifies how to group the matches found within the pattern.
Capture	Provides the index of a match within the grouping.
Type	Proxy classes generated using the .Net Framework (wsdl.exe) use this attribute as the returned class.

Within our WSDL file we also have the <message> elements. These are used to define the input and output parameters. Within this element is a <part> child element that represents the particular parameter. This element contains a name and type attribute. The name attribute contains the unique name of the parameter being passed, and the type attribute lists the data type of the parameter being passed. WSDL isn't limited to simple type only. If you want to define more complex types using XSD, they can be defined within the <types> section of the services description and then specified within the data type for the parameter. For our example I am using the simple type string and defining "Body" as the parameter name.

```
<message name="TestHeadersHttpGetOut">
    <part name="Body"
element="s0:string"/>
  </message>
```

Using Regular Expressions

Of course, all elements are important for a properly formatted WSDL document. The most important element for parsing is the <match> element. This element contains the actually parsing instruction and the data elements required by the .NET Framework to properly generate the proxy classes. The <match> element is part of the fully qualified <text> element and contains the <output> and <operation> elements of a specific <binding>. Within the <match> element there are a variety of attributes (see Table 1).

By far the most important is the pattern attribute. This contains a regular expression syntax pattern that will be applied against the parsed page and will determine the return value. By definition, a regular expression is a series of characters that define a pattern. The pattern is then compared against a target string to determine whether there is a match to the pattern in the target string.

The real power of these expressions is in the use of metacharacters to indicate character positioning, grouping and even repetition. The easiest example of a metacharacter is the "*" from the old DOS days. The .NET Framework contains a fairly extensive set of expressions that can be used when parsing pages. For more information and examples of syntax, take a look at the .NET SDK. For our example, I attempted to locate both the <TITLE> and the <H1> tags within the base HTML elements.

```
<output>
        <text xmlns="http:
//microsoft.com/wsdl/mime/textMatching/">
          <match name='Title'
          pattern='TITLE&gt;(.*?)&lt;'/>
          <match name='H1'
pattern='H1&gt;(.*?)&lt;'/>
 </text>
</output>
```

One thing I learned while writing this sample is that case sensitivity is important. So, for this example and your own code, make sure that you either turn on case insensitivity or are aware of how the HTML tags are written.

Generating Proxy Classes

The job of the service description file is to define how to communicate with the Web service. XML Web services allow communication over a network in a variety of protocols. This means that the client and Web service communicate using SOAP messages that encapsulate both the in and the out parameters as XML. It is up to the proxy class of a Web service client to handle the work of mapping parameters to the actual XML elements defined within the service description file and then sending the SOAP message over the networks.

Within the .NET Framework a proxy class is generated using the Wsdl.exe utility. This utility examines the WSDL file and creates proxy classes that can be invoked to communicate with the target Web service over the network. The service in turn processes both the incoming and outgoing SOAP messages. By default, the Wsdl.exe utility assumes SOAP over HTTP to communicate with Web services. The utility also provides the ability to generate classes that can communicate with Web services using either the HTTP-GET or HTTP-POST protocol.

Wsdl.exe is run from the command prompt. The utility supports a wide variety of switches that allow you to define such things as language type, passwords, and even namespaces. For a complete listing of the available options, run "wsdl.exe /?" from the command prompt. For my example, I was interested in creating a Visual Basic .NET–based class and a specific class name. From the command prompt I ran the following:

```
Wsdl.exe /l:vb /out:datareturn.vb
http://localhost/Webinfo/datareturn.wsdl
```

The output of Wsdl.exe resulted in the creation of a class called datareturn.vb. This file contains a proxy class that exposes both synchronous and asynchronous methods for each of the methods in the Web service. In this example the generated methods were TestHeaders, BeginTestHeaders, and End TestHeaders. The Testheaders method provides synchronous connectivity to the Web service. Both the BeginTestHeader and EndTestHeader can be used to provide asynchronous Web service connectivity.

Consume the Web Service

Once the generated proxy class is added to the project and a Web reference is set to the WSDL file, you are ready to start using the service. Within an ASP.NET Web page you can call the proxy class and return the requested parsed data from the Web service using the code:

```
Dim Getdata As New localhost.GetTitle()
Dim match As localhost.TestHeadersMatches

match = Getdata.TestHeaders
TextBox1.Text = match.Title
TextBox2.Text = match.H1
```

Summary

As I said at the beginning of this article, this is a simple example of what you can do. As I spoke with my customer over the next weeks, he started to understand the value the CIO and upper management were looking for. He developed a Web services strategy that relied on current investments and leveraged them where appropriate. His strategy was centered on a gradual transition that leveraged the full power of his existing infrastructure.

As you download the source code (located at www.sys-con.com/webservices/sourcec.cfm) provided with the article, I challenge you to do the same thing. Use existing Web sites when appropriate and integrate and enhance them with the power of a Web service.

The Case for Web Services Development in Singapore

A "living lab" for practical Web service development

When Microsoft was looking to create the world's first nationwide, community-based Web services offering, it looked to Singapore. In a report distributed by the World Economic Forum, Singapore possesses the "Most Wired Government" and, as a country, is the "Most Effective in Promoting the Use of IT Among Its Citizens."

Microsoft recently launched a strategic collaboration with Singapore to test its .NET Web-based services. A world's-first in nationwide trials, Singapore will be a testbed and living lab for this new, emerging technology. .NET MySingapore, as the initiative has been dubbed, is the first of several projects within Singapore's Web services development framework. The government's partnership with Microsoft is part of a larger, nationwide push to showcase the many possibilities offered by Web services technologies.

Singapore is one of Southeast Asia's earliest adopters of Web services, with many organizations and enterprises taking advantage of this technology. The Esplanade, for example, Singapore's new state-of-the-art theater complex, is leveraging Ecquaria Technology, a Singapore company providing Web services software infrastructure and solutions, to implement a Web services platform for integrating its internal IT systems. The Ecquaria Service Oriented Platform (SOP), a fully J2EE-compliant Web services software infrastructure, is the common platform for all of the Esplanade's online services. The SOP provides plug-and-play, horizontal Web services that enable customers to schedule and view shows through a backend integration to Esplanade's Web-based ticketing system, access to news and events information, online purchases, press center, event hiring, e-mail, and short text messaging communication services.

Ingredients of Success for Any Country's Infrastructure

A key ingredient for Web services deployments in Singapore is the existing telecommunications infrastructure and connectivity rates. Currently, the country boasts nationwide broadband access coverage of 99%, with one out of every two Singaporeans online, and seven out of every ten Singaporeans communicating wirelessly. This network of personal, regional, and international connectivity forms the base infrastructure for Web services deployment.

To Singapore's software and IT services industry, Web services are a potential engine for economic growth. Already, several technology companies have come together in an effort to agree on common standards to further the adoption of Web services, and are beginning to deliver products, services, and architectures for the marketplace. A number of multinational corporations (MNCs) are currently participating in Singapore's infocomm Local Industry Upgrading Program (iLIUP), a business development program that matches MNCs with local companies, and are developing Web services with their Singapore-based technology partners. In addition to Microsoft, this group includes companies such as IBM, Sun Microsystems, Borland, Computer Associates, BEA Systems, Progress Software, ILOG, Oracle, Hewlett-Packard, and Software AG.

Singapore businesses have been eager early adopters of new technologies. For example, the TradeNet system (www.TradeNet.gov.sg) pioneered Singapore's use of technologies for business competitiveness. Under this system, the community receives a fast, efficient means of submitting permit applications electronically to multiple government bodies for processing of import, export, and transshipment documentation. When an application is approved, a permit message will be returned electronically to the sender.

Additionally, Singapore's success in e-government, coupled with its citizens' acceptance of this program, demonstrates yet again their willingness to accept new technology. In early 2002, Singapore was recognized at the E-GOV 2002 show in Washington, DC, for the development of its Public Services Infrastructure (PSi), the platform from which all public services are brought online. PSi is based on J2EE and is the backbone for all of Singapore's e-government initiatives. These successful case histories provide a strong foundation for Singapore to embark on the development of Web services.

Author Bio

As the assistant chief executive officer for Infocomm Development Authority (IDA), Khoong Hock Yun works to establish Singapore as a dynamic and vibrant Infocomm hub. His primary role is to grow and strengthen Singapore's broadband industry, thus contributing to the nation's fast-growing digital economy. He is responsible for developing strategies to attract local and international businesses to use Singapore as a conduit for launching and testing broadband applications and services in Asia.

Recognizing this possibility, Singapore's Infocomm Development Authority (IDA) recently announced a framework that identified Web services as an emerging technology with the potential to contribute to the country's efforts in leveraging technology to improve citizens' lives and create opportunities for businesses.

The IDA, as the government agency that drives infocomm development, believes it must play a catalytic role and invest in Web services while the technology is still nascent. Besides having a good infrastructure and a history of early adoption of new technologies, Singapore has the advantage of being home to many leading infocomm companies that are actively promoting this technology. Accordingly, it is these organizations that provide the critical impetus to drive the development and implementation of Web services.

Singapore's Strategy for Implementing Web Services

In pushing for Web services to grow in Singapore, there is a need to develop manpower capabilities and intellectual property; create market proof points by encouraging the development of more flagship and trial projects; and address key industry-wide issues such as interoperability, security, network identity, semantics, management of Web services, and a public UDDI registry.

As Web services technology standards are relatively new, companies still need to learn how best to use new development tools and deployment platforms, adapt to new development methodologies, as well as develop new business opportunities and models to exploit the benefits of Web services.

The other prevailing question occupying the marketplace is not "Will the technology work?" but rather "Will it sell?" There is a need to provide a sufficient comfort level to end users by raising awareness of the use, benefits, and stability of XML Web services. At this early stage, the key benefits anticipated from implementation are the expectation of its dominance in the deployment of new application solutions for Fortune 2000 companies by 2004, and driving a 30% increase in the efficiency of IT development projects by 2005. This will bring a new level of connectivity, communications, and fulfillment of products and services between businesses, governments, and users.

Finally, industry-wide issues, such as interoperability, security, network identity, semantics, management of Web services, and public UDDI registries, have to be resolved.

Nevertheless, Singapore sees this as an opportunity since in the past year alone, businesses worldwide invested some $964 billion in IT, pursuing newer versions of operating systems and office automation software. The spending spree has left many enterprises with more selective tastes and wide open to suggestions that existing investments already put in place can be made to work better, together.

Not Just One

In addition to Microsoft, Sun Microsystems has been active in Singapore. As part of the fourth "Java Tarik" initiative, Sun, in collaboration with the IDA, Singapore Institute of Manufacturing Technology under the Agency for Science, Technology, and Research (A*STAR) and other technology partners, launched the Java Smart Services Lab (JSSL) in August 2002. This facility will drive the adoption of Java-based Web services in the local industry through research, pilots and trials, incubation, and development of the appropriate manpower. The JSSL will also help shortlist and recommend to the IDA potential Java-based Web services projects and will contribute consulting, personnel, and incubation, facilities for all pilots or trials.

IBM, which also has a large Singapore presence, is working through an alternative approach to building capabilities that support the emerging Web services cluster in Singapore. Big Blue has chosen to encourage the adoption of open standards by harnessing open Web services standards that allow more interoperability options for users. In a collaborative effort with the local tertiary institution, Nanyang Polytechnic, IBM launched the NYP-IBM Web Services Innovation Zone (WIZ) in September 2002. WIZ will promote open Web services standards such as XML, SOAP, WSDL, and UDDI, by allowing software developers to work with open standards programming models, encouraging application deployment across multiple platforms. Nanyang Polytechnic will not only develop capabilities in Web services, but will also promote the development of other innovative applications and services by recommending to the IDA suitable pilot projects that would benefit from its support.

To address these issues, Singapore will implement a series of strategic thrusts over the next three years. These include developing the correct intellectual capital to address the shortfall of capability, installing the complementing enabling infrastructure, and creating a "living lab" environment for the development of Web services.

.NET MySingapore

The .NET MySingapore project is a good example of Singapore's strategy in action. First announced in April 2002, the Microsoft trial is part of a larger drive to develop and accelerate the adoption of Web services in Singapore. Working together on this initiative are the IDA, Microsoft, Singapore Economic Development Board (EDB), Institute of Systems Science (ISS), an insurance co-operative – NTUC Income – and IT services provider – National Computer Systems.

.NET My Singapore is based on four strategic pillars:

- Creating Web services that enable Singapore's citizens to easily access and utilize community services.
- Developing a new training curriculum and the world's first program for certifying Web services professionals, as well as establishing a Microsoft .NET Professional Certification. ISS, working with Microsoft, has developed a new training curriculum for post-graduates. As of November 2002, 30 students at ISS will have access to two XML electives (Master of Technology and E-Business Architecture & Design). The project team has created a steering committee to lead the development of the world's first .NET Professional Certification Program and .NET Academy.
- Establishing a joint Microsoft-IDA overseas development program designed to give Singaporean developers better access to new Web service technologies. This will involve the creation of a position at Microsoft's research facilities in Redmond, co-funded by Microsoft and IDA. Currently in its early stages, the program's Singapore-based partners are being identified to work with Microsoft and IDA by the first quarter of 2003.
- Developing an emerging technologies lab to explore new technologies with academia and industry.

The early fruits of this collaboration with Microsoft were launched in mid-October 2002 as the first community-based Web services.

Through .NET MySingapore, NTUC Income will enhance its existing online services for its 1 million policyholders. Called www.BigTrum pet.com, it is more than a Web site and is designed to connect Singapore in an unprecedented manner, utilizing both traditional and XML Web services. The projected adoption for these services is 100,000 users; the initial phase features these key components:

- ***My Money:*** Features insurance and financial planning services
- ***My Career:*** Features job search capabilities and lifelong learning programs
- ***My Home:*** Features home services, child tutoring, and domestic helper searches and management options
- ***My Workplace:*** A schedule and time maintenance program that connects personal schedule, tasks, and contacts with the users' office, customers, and clients
- ***My Clubs:*** Enables users to set up and maintain special interest clubs

The backbone of Big Trumpet is "My Folder," which provides secure access to bank, financial, medical, and personal information. Big Trumpet lets individual users access their personal information online with any device, at any time, and with a high level of security. Additionally, users also have control over what information others can see. For example, a user looking for a job can allow potential employers to see his or her career history, and nothing else.

Singapore believes that .NET MySingapore is a great example for private developers, spurring further development in Web services. As highlighted at the .NET MySingapore memorandum signing in April, the .NET MySingapore program will provide an excellent platform on which application and service providers can create new and innovative products and community-based Web services. Singapore welcomes such partnerships, and it is part of the IDA's strategy to accelerate the adoption of Web services locally, and a testimony to the fact that Singapore is a living lab for new and emerging technologies.

Conclusion

With its focus on testing and deploying new and cutting-edge technologies, Singapore will continue to serve as a good test bed for Web services. The country will continue to be a "living lab" for exploring the potential practical applications and benefits of this emerging technology, and the possible creation of future business model applications.

Content Is King

Leveraging real-time business content is the key to realizing the promise of Web services

A rapidly growing number of organizations are turning to Web services as a means of bringing increased agility to their core business systems. A key part of this increased agility is better access to real-time business content in order to address such business requests as:

- Prompt notification of discrepancies in orders and inventory, particularly regarding high-value customers, promotional offers, etc.
- Enabling our human resources staff to access all information critical to their job functions, while enforcing access privileges to shield them from information outside of their authorization
- The ability to log transactions as they occur and proactively send messages to employees or partners if there is an application break.

Why is it easier to meet these requests with Web services than with previous generations of technology? Two things come to mind: widely adopted industry standards and an unprecedented ability to harvest business content in real time.

Web services come with a set of Internet industry standards (see sidebar, page 51) that are gaining across-the-board acceptance throughout the IT community. These standards are promoting a dramatic growth in Web services built from scratch, applications outfitted with Web services interfaces, partners' Web services, and the emerging hosted Web services available for lease. They are also enabling businesses to easily tie together Web services–based applications regardless of platform, development environment, or physical location. The results are powerful, federated, heterogeneous business systems that span organizational boundaries and provide a more comprehensive and timely record of the state of the business.

Before the advent of Web services, it was difficult – if not impossible – to harvest real-time business content unless the application had been specifically coded to provide that information. Runtime systems-management tools could observe system-level information such as the number of messages, throughput, queues, failures, speeds, and response rates. However, these tools provided no visibility into the content of the messages, such as the dollar value of an order. Web services–based applications, by contrast, can unlock access to rich payloads of information – such as the source and destination of the data, information on the person requesting the data, and the quantitative business information itself. By accessing this type of real-time information, organizations can not only address security and service issues, but can also realize new opportunities for business success.

Accessing Real-Time Business Content

As organizations run more of their business functions on Web services–based systems, they can gather real-time information about usage patterns and business content by examining the messages passed among Web services. Since these messages are written in XML, even when communicating among Web services built with different tools they provide a standard, structured format for accessing business content. This content can be accessed as messages pass into (or out of) any Web service. The standardized format makes it easier to collect business information at multiple points in the business process and easily aggregate data elements into meaningful business information.

Theoretically, real-time business information can be gathered by looking within the Web services themselves, but this approach presents several difficulties. First, it's not always easy – or even possible – to instrument the internals of a Web service in order to collect a specific piece of business data. This is certainly the case with fast-breaking business opportunities that do not allow enough time for extensive recoding. Second, an IT organization may not have access to the internal workings of a Web service. This is most often the case with a packaged application that presents a Web service interface. (According to a recent report by Forrester Research, Inc., most major commercial applications will have a SOAP interface by the end of 2002.) Third, many organizations consume external Web services that are beyond their control, such as Web services they lease or Web services they consume from partners. For example, an organization may access data from an external shipping

Author Bio

Paul Butterworth is cofounder and CTO of AmberPoint, Inc., a pioneer in the emerging field of Web services management software. Paul was formerly CTO of Sun Microsystems' Java Tools Division and was a founder of Forte Software. He also served as chief architect and director of product engineering at Ingres Corporation. He holds a BS and an MS in information and computer science from University of California at Irvine.

company whose Web service they cannot reconfigure. In all of these cases, it would be very difficult to access information from within a Web service itself. In each case, however, it remains relatively simple to monitor the XML messages that are passed among such Web services.

XML is a highly structured format for data interchange. Both requests and replies contain "tags" for the delivered data. These tags can be read as a way of gathering meaningful, real-time business information and applying it in ways never before possible.

For example, say a company uses an XML Web service for order processing. A business manager may want to be notified when one of his preferred customers places an order of 1 million dollars or more. He may want to give priority to filling such orders, even if it means giving a retroactive back-order notice to a lower-priority customer. Traditionally, this capability was either pre-coded into an application or the business manager had to wait to collect this information after the fact through summary reports or a data warehouse. Organizations deploying Web services can employ a new, more powerful model. They can quickly instrument their Web services–based systems to check the business content of each request – or selected requests – to gain insight into and provide immediate notification of new or emerging conditions that are important to their operations.

Combining Content with Context

In addition to making use of business content (such as order size), Web services systems can leverage contextual information – who's requesting the data, what are his or her access rights, and what type of client device is that person using, to name a few examples. When the user is authenticated through a security system, the Web service might call an LDAP directory for a complete user profile, including user identity, roles, preferences, and access policies, and can apply this information accordingly.

Content and context form a powerful combination. A Web service can use both types of data to adapt and reconfigure itself on the fly to ensure appropriate access to the immediate audience. Say a company has a Web service that tracks "Total Parts Ordered." It can make the application available to different types of users, each with its own requirements and restrictions. A customer might be allowed to look up the real-time status of his or her order, but no one else's. An operations manager might be allowed to track a complete record of all orders, backlogs, and shipments. A business analyst might correlate information on orders with cost of goods sold in order to understand profit contribution. A sales manager might use the same Web service to compile a regional sales report.

> **"By accessing this type of real-time information, organizations can not only address security and service issues, but can also realize new opportunities for business success"**

By using the message content in conjunction with its context, a Web services system can make XML Web services far more responsive to specific business requirements than previous generations of applications ever could.

Business Examples

The following are just a few examples of how access to the content of XML messages can add business value.

Content-Based Security

Security is the number-one concern for IT managers considering Web services. The thinking goes, if Web services allow access to anyone, what's to keep everyone from getting at business-critical data? The ability to implement content-based security should stem such concerns.

The more abundant the contextual data, the greater the opportunity for access control. By using profile data in conjunction with business content, a business system can grant user-specific, content-specific read and/or write access to a Web service. By interacting with the existing authentication system, it can dynamically tailor responses based on each user's most current security profile and the context of the request.

Here's an example. A company employing a human resources Web service can conceal salary information for all employees of an equal or higher rank than the requestor. Leveraging both context (profile information) and content (salary information), the company can secure inappropriate data on a content-specific level.

Evaluating and Maintaining Service-Level Agreements

It's necessary to track the content and performance of Web services in order to set and maintain service levels. Consumers need to verify that they're getting the service levels they requested or paid for. From a Web service producer's standpoint, it's necessary to track operations to ensure that acceptable levels of service are met. In addition, producers must be prepared to make hard choices. For example, given limited resources, it may make good business sense to dedicate system use and inventories to the highest-value orders and most valuable customers.

Insight into Business Operations

Message-stream analysis enables business managers to identify new opportunities and understand consumption patterns. By seeing who's using various aspects of Web services and how the services are being used, the organization can gauge the effectiveness of its products and promotions, calculate the return on its investments, and more intelligently plan its IT roadmap.

Content-Based Alerts

Businesses need to know about fast-breaking, exceptional conditions. For example, a company launches a "while supplies last" marketing campaign. The program manager responsible for the campaign needs to know when inventories are depleted in order to handle the respondents whose requests cannot be fulfilled – ideally while the respondents are still online. Business managers also need to know about extraordinary conditions that might affect customer satisfaction so that they can deal proactively with them – before the customer complains. A Web services system can track the content of transactions in progress and send immediate alerts whenever appropriate.

Upgrading Online Systems

The traditional change control model calls for relatively infrequent system upgrades, perhaps quarterly or annually. With many older systems, upgrades were installed over a weekend or during a holiday because the system needed to be taken down. This model, of course, doesn't work in the era of Web services, where businesses need to be online 24 x 365 and where market changes are measured in days rather than months.

Fortunately, a new version of a Web service can be installed without taking the system down by simply redirecting messages from the old service to the new one. Such changes are transparent to Web services consuming data from the updated system. Moreover, by accessing the business context and content, changeovers can be done gradually and selectively. For example, lower priority customers placing smaller orders can be redirected to a new version of the Web service until it has been sufficiently "burned in." Once the new version is robust enough to handle all traffic, the old version can be de-installed without disrupting the system.

The Future Is Now

With all the talk of Web services as "the next big thing," it's easy to forget that they've already arrived. New Web services go into production every day. Many initial Web services implementations are integration projects inside the firewall. For instance, Citigroup used SOAP interfaces to join 200 different content and data feeds into a customizable portal for its bankers and brokers. Others are linking Web services outside the firewall. Lloyds TSB Commercial Finance accesses Graydon UK's credit service to provide real-time rate calculations for short-term loans against a borrower's accounts receivables. MedUnite uses Web services technologies to provide more than 130,000 healthcare offices with centralized access to data from hundreds of insurance providers. The list goes on.

As more and more organizations migrate from early Web services experiments to production-ready use, the number and magnitude of interconnected networks of applications will grow exponentially. Business-driving new combinations of data will result, elevating the role and worth of Web services to even greater heights. Backed by standards and bolstered by their use of content, Web services will raise the business value organizations harvest from their information systems.

The 90-Day Plan

As with any new technology, the best way to get a handle on the potential reward is to create a proof of concept (POC). With a well-devised POC, you can get a jump on the competition while minimizing risks.

Month One

Start by setting objectives and identifying opportunities to dynamically create new correlations from previously unrelated data sources. Form a task force comprising both business and IT professionals. Look at systems and subsystems in your environment and identify information pairings that can solve a real business need. Look for metrics with which to gauge the success of your POC.

Month Two

Develop your POC incrementally. The POC is a learning exercise, so don't overarchitect it. Start with a relatively simple service and move forward in short iterations. You can always add more functionality later. By focusing your efforts on a single business need, you minimize costs and can then build on the knowledge you acquire. Incorporating a management layer early in the process can assist in measuring POC effectiveness.

Month Three

Measure as many quantifiable results as possible. Analyze the results – both system metrics and business content – and compare against the objectives you established at the outset of the project. If possible, use these metrics to estimate ROI for a production Web service. Use your findings to devise a Web services project plan.

Web Services Standards

Standards bodies and industry consortia such as W3C, WS-I, and OASIS have done well to guide and promote Web services standards. Here are some of the more widely adopted standards, to date:

- ***eXtensible Markup Language (XML):*** The standard for defining data interchange formats within the Internet world. XML is a set of data descriptions (tags) and rules that simplify and enhance the process of data interchange between unlike computer systems.
- ***Simple Object Access Protocol (SOAP):*** SOAP is a protocol used for exchanging structured data in decentralized, distributed environments.
- ***Web Services Description Language (WSDL):*** WSDL files provide contact information, descriptions of the Web services, their location, and specification on how to invoke them.
- ***Universal Description, Discovery and Integration (UDDI):*** The Internet standard for the description, registration, and discovery of Web services. The resulting UDDI framework is a set of databases where businesses can register their Web services as well as locate other Web services.

Increased Use of Web Services Makes Testing a Good Investment

Keep your performance in line with your business requirements

In the past, if you wanted to book a holiday over the Internet you would browse a travel agent's Web site, select a holiday, and book it. If you were lucky, you'd be able to rent a car and possibly even check on the weather. Behind the scenes, the Web server – the computer and software that the Web site lives on – would be accessing a proprietary database, or would be communicating in a proprietary fashion with back-office systems to get information about your holiday.

This all changes with the use of Web services. A Web service is a system designed for the sole purpose of sitting somewhere on a network and servicing specific requests from clients. These clients aren't real people browsing the Internet from their desks; they're other computers.

What Are Web Services?

When Web services are used to book a holiday, you still browse your travel agent's Web site, but behind the scenes something different happens. Rather than accessing a proprietary database to get flight details, the Web server talks to a Web service somewhere else on the Internet. All this Web service does, day in and day out, is respond to requests for flight information from travel agents' Web sites. What is more, these Web services can publicize their existence and communicate with Web sites and other Web services in an open, public format. The travel agent's Web site accesses different Web services for flight, accommodation, and weather information.

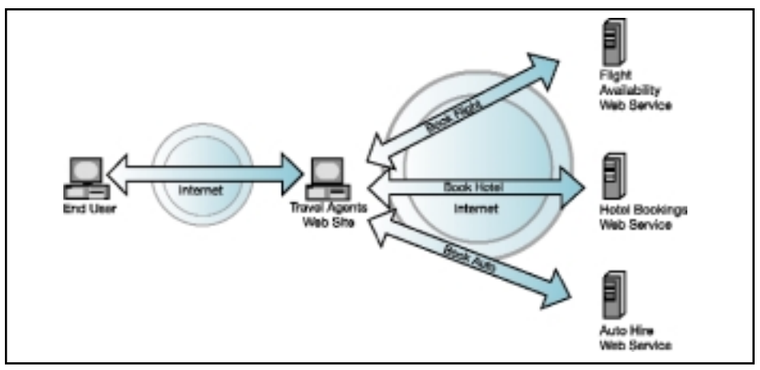


FIGURE 1 | A travel agent's Web service

Since the Web service is on the Internet and communicates in standard, published protocols, anybody can communicate with it. If you're running a conference and want people to attend, you can have flight information on your Web page. With five minutes of coding, you'll be able to let people view flight availability to your conference and book flights online, without leaving your site (see Figure 1).

Challenges of Testing Web Services

Web services are vital to a business's operation and need to be tested like any other important application. Like Web sites, they are highly visible and should be robust.

There are two broad types of Web services: those used in an intranet and those used on the Internet. Intranet Web services are used internally by organizations and are not exposed to the general public. An intranet Web service might be responsible for handling vacation requests from employees, for example. The company's intranet Web site would access this Web service and employees could request vacations. Managers could authorize the vacations and colleagues could check when other employees are on holiday. Human resources could then write a simple application in Visual Basic to make sure that help-desk employees don't all take their vacation at the same time. All this is possible without the user having detailed knowledge of how or where this information is kept.

Testing intranet and Internet Web services presents subtly different challenges. With an intranet Web service, an organization is likely to have control over who can access its Web service. Since it is on an internal network and only internal users have access to it, there is a theoretical maximum num-

Author Bio

Neil Davidson is technical director of Red Gate Software Ltd. Red Gate's latest product line is the Advanced .NET Testing System (ANTS), which provides tools for load-testing and code profiling for XML, Web services and Web sites. Visit www.red-gate.com/ants.htm for more details.

NEIL.DAVIDSON@RED-GATE.COM

ber of users, and organizations can make certain assumptions about security. Anybody can access an Internet Web service, which means that there are additional scalability and security considerations.

Another challenge in testing Web services is that they do not inherently display a user interface that can be tested. Although some development tools will build a Web page around a Web service, this is not part of the Web service itself. Visual Studio .NET, for example, will generate a page that allows users to invoke methods of Web services and view the XML returned, but this is inefficient for anything except the simplest of Web services. This lack of user interfaces means that Web services are hard to test manually, but are an ideal candidate for automated testing. Consequently, some programming skills are almost certainly necessary for Web service testers. A Web service is not the sort of application you can test by key-bashing.

Know Your Clients

It's important to know your clients when testing Web services. If you're expecting to have 100,000 clients who will only ever make one request a day, then the load is not too heavy (about one request a second). If, however, 50% of these requests will happen between 9 and 10 a.m., then that's a different story (about 15 requests a second). Another possibility is that you will only have 100 clients, but these clients will be making 10 requests a second, each and every second of the day.

You might expect, on average, to be able to service 100 requests a second, but what happens in the event of a massive peak? Will your Web service cope, slow down, or crash? Or will it simply refuse to service the 101st request? These are all things you need to know and test before you release your Web service.

You must also know how clients will be accessing Web services. It is likely that they will be making SOAP requests. In this case, you should make sure that your Web-testing tool supports this protocol. Some testing tools record scripts by browsing to the Web page representing the Web service and then recording the HTTP GET and POST requests that the browser makes. Although similar, this is not the same as using the SOAP requests that your clients will almost certainly be making.

What Service is Acceptable?

The next question is, "What is acceptable service?" Your service-level agreements might specify that you need to respond to 95% of requests within one second; in this case, you know what is acceptable. In any case, it's important to realize that a Web service is not the same as a Web site, so acceptable response times will vary. While it may be acceptable to keep response times for a Web site under 10 seconds, a Web site may be making 10 calls to different Web services, so the Web site will expect each Web service to return information within one second.

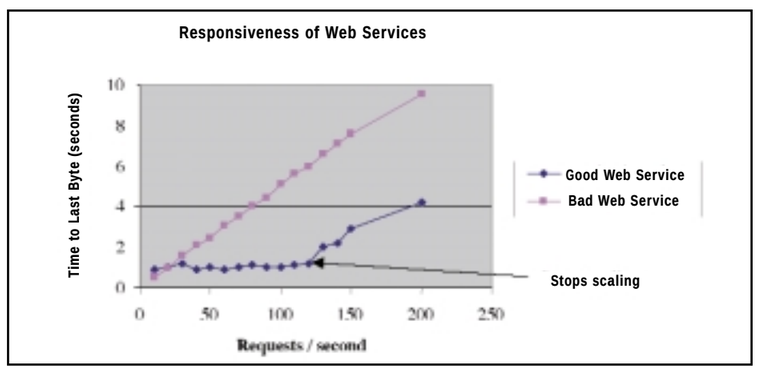


FIGURE 2 | Web services with different scalability properties

While you are carrying out your test, the tool you're using should be able to identify how the values below change as you increase the number of clients. These are all measurements of how the client experiences your Web service.

- ***Time to connect:*** The time it takes to make a connection from the client to the Web service. This should be as low as possible.
- ***Time to first byte:*** The time it takes for the client to start receiving data back from the Web service. If the Web service needs to do a lot of thinking for each request, then this time could be significant.
- ***Time to last byte:*** The time it takes for the client to receive the last byte of information back from the service. If the service needs to return a large amount of data (if it is returning maps or images, for example), then this could be significant.

Although the absolute values of these metrics are important (you want to keep these numbers as low as possible), the way they change as you increase the load on the Web service is more important. Ideally, you want these metrics to remain constant. If they increase linearly, then you're in for a shock.

Suppose that with one user, the time to last byte is a blazingly fast 10 milliseconds, for 10 users it's one-tenth of a second and for 100 users it is one second. That's still reasonable. But for 1,000 users it's going to be 10 seconds. If your Web service needs to cope with 10,000 users, then requests are going to take more than a minute. That's clearly unacceptable.

It's more likely that you'll find that your Web service scales (i.e., the response time for requests remains constant) up to a certain number of virtual users, and then it stops scaling. To troubleshoot this, you need to keep an eye on the performance of the server on which the service is running. You need to know what is causing this change in behavior: Is the CPU saturated? Is the disk thrashing? Is the network card causing the bottleneck? These are all possible causes of performance problems.

Once your system hits a bottleneck, you need to know if there's a way around this problem. If you add another processor to the server, will it double its capacity? If you add another server, will that double its capacity? If you have a Web service that will scale in this way, then you know that you will be able to cope with extra demand by adding more and more hardware. If your Web service doesn't scale in this way, then it won't perform no matter how much money you spend on expensive hardware.

In Figure 2, "Good Web Service" scales well until about 120 requests per second (the time to last byte is constant), but then stops scaling. "Bad Web Service" scales poorly. As the number of requests per second increases, the responsiveness decreases linearly.

Testing Types

As with any other application, there are different kinds of testing you can carry out on Web services.

Proof-of-Concept Testing

Since Web services are a new type of software, you need to understand if the architecture you have chosen for your Web service is the correct one. There are many different choices to make – which tool vendor to use, which programming language, and which database back end, for example. If you can clarify and resolve these issues before you start developing, or early on in the development life cycle, then you will save a lot of time and money further down the line.

Because most of the questions you need to resolve will concern scalability issues (will the architecture we're using really cope with 1,000 simultaneous users?), a proof-of-concept test is normally a cut-down load test (see below). There's no need to run it on powerful hardware or get exact answers; your aim is to answer the question, "Am I going in the right direction?"

Functional Testing

This ensures that the functionality of the Web service is as expected. If the Web service divides two numbers, does it give the expected result? If you place a zero as the denominator, does it handle this correctly? Does your Web service implement security and authentication as it is meant to? Does your Web service support all the communications protocols it is meant to? Because your Web service can be accessed by clients whom you can't control, what happens if they make requests you aren't expecting? Bounds testing and error-checking are especially important.

Regression Testing

A regression test is normally a cut-down version of a functional test. Its aim is to ensure that the Web service is still working between builds or releases and it assumes that some area of functionality was working in the past and its job is to check that it still does. If your development team has changed the method that divides two numbers, does that method still work? Does the method that multiplies two numbers still work? Is the performance still acceptable? Since regression testing is, by its nature, a repetitive task, it must be automated. This is true for traditional applications and Web sites; it is even truer for Web services.

Load/Stress Testing

The aim of load/stress testing is to find how your Web service scales as the number of clients accessing it increases. You have carried out functional and regression testing, so you know that your Web service will cope with a single user. What you need to know now is how it will cope with 10, 100, or 1,000 users. If you double the number of users, do response times stay the same? If you double the number of servers running your Web service, does its capacity double? This testing simulates a very complex environment, so it must be automated.

To obtain objective results, it's important to carry out the testing in a controlled environment. If you want to know how your Web service responds as you simulate more and more users, then you must keep all other factors (hardware and networking, for example) constant. This means that you should not carry out a load test on a live system over the Internet. Apart from the problem that bandwidth would pose, this is not a controlled environment.

You might decide that you simply cannot carry out the test in-house. If you are writing a large-scale Web service that needs to cope with 10,000 requests a second, then the odds are that you don't have the necessary hardware in-house to do this. You will probably need to consider hiring a third-party consultant or using a third-party scalability lab to carry out this test.

The ultimate aim of load testing is to reassure you and confirm that your Web service will respond acceptably for up to *x* clients making *y* requests a second.

Monitoring

Once your Web service is live and being used by real clients, you will need to keep an eye on it. Is it still working? Are response times adequate? At what times of day is it busiest? It is essential to monitor the Web service.

Testing Makes Business Sense

Web services are becoming a major part of the Internet and corporate intranets, ideally ensuring fast access to a wealth of related information. At the same time, these services place increased demands on Web applications and architectures, often resulting in poor performance that causes users to become frustrated or abandon a Web site.

Automated testing allows IT departments and contractors writing Web services to determine design options and performance trade-offs during the project development stage. This enables developers to side-step problems that could turn away customers or alienate internal users on a live site. Developers can optimize performance for specific applications, and test new and updated Web services to make certain they meet performance criteria before they are implemented.

While testing Web services can be challenging, it's a vital part of ensuring that your Internet or intranet performance is in tune with your business requirements.

WSJ News

Oracle Delivers Industry's First Personalized Java Development Tool

(Redwood Shores, CA) – Oracle Corp., the world's largest enterprise software company, has announced the release of the latest version of its Java IDE – Oracle9*i* JDeveloper Version 9.0.3. It features the new MyJDeveloper Extension Manager, which allows developers to personalize the development environment to meet their project needs.

This release integrates J2EE and Web services development to optimize Java development productivity. The entire development life cycle of Web services is now supported, including visual development, creation, publishing, UDDI discovery, testing, and debugging, as well as interoperability with .NET Web services.

Oracle9*i* JDeveloper is immediately available for free download and evaluation from Oracle Technology Network.
www.oracle.com

Westbridge Technology Offers Free Tool

(Mountain View, CA) – Westbridge Technology, Inc., a provider of security and monitoring solutions for XML Web services, has announced general availability of the Westbridge XML SOAP Monitor. The new tool enables enterprises to easily and effectively monitor their networks for all XML Web services traffic without requiring changes to the network.

The Westbridge XML SOAP Monitor is a lightweight application that plugs into any router mirror port and unobtrusively monitors all network traffic for unauthorized SOAP traffic, without affecting the architecture or performance of the network. Application architects, security analysts, and IT operations staff can easily view the content of a message for sensitive information and view the sender and receiver of each message. Filtering occurs in near real time.
www.westbridgetech.com

Rapidly Growing SYS-CON Media Garners Awards

(Montvale, NJ) – SYS-CON Media has been named one of the 500 fastest-growing technology companies in North America by Deloitte & Touche in its 2002 Technology Fast 500. The announcement came one week after SYS-CON was named one of the nation's fastest-growing private companies by Inc 500 for the third time. The 2002 Technology Fast 500 listing is available exclusively at Forbes.com.

SYS-CON Media is widely recognized in the i-technology and magazine publishing industries as the world's leading publisher of print magazines, electronic newsletters, and accompanying Web portals. The company has further solidified its dominant role in the *i*-technology space with the 2000 launch of an events business with trade shows, conferences, and educational seminars.

SYS-CON Media has achieved a record 752% growth in the past five years. The company's revenue and earnings have grown dramatically since its inception in 1994. From 1998 to 2001, revenue grew at a compounded annual growth rate of 72.9%. In 2003, the company projects its gross margin to increase 51.9%, and the contribution is projected to increase 70.4%, which will keep SYS-CON in an impressive growth pattern for 2002, 2003, and beyond.
www.sys-con.com

Systinet Announces WASP 4.0 OEM Edition

(Cambridge, MA) – Systinet, an independent provider of Web services infrastructure software, has announced a strategic initiative that includes products and services designed to make it easy for original equipment manufacturers (OEMs) and independent software vendors (ISVs) to benefit from Web services technology. As part of this initiative, Systinet announced a new version of its Web services product suite, WASP OEM Edition.

WASP OEM Edition features enhanced customization capabilities, APIs, and documentation support. This release is free for evaluation and testing, and can be downloaded from Systinet's Web site at
www.systinet.com.

SAML Ratified as OASIS Open Standard

(Boston, MA) – The OASIS interoperability consortium has announced that its members have approved the Security Assertion Markup Language (SAML) v1.0 as an OASIS Open Standard – the highest level of ratification. SAML is an XML-based framework for Web services that allows the exchange of authentication and authorization information among business partners. SAML enables Web-based security interoperability functions such as single sign-on across sites hosted by multiple companies.

SAML incorporates industry-standard protocols and messaging frameworks, such as XML Signature, XML Encryption, and SOAP. SAML complements Web services standards such as SOAP, which lack inherent security features.
www.oasis-open.org

webMethods 6 Launched

(San Francisco) – webMethods, Inc., a provider of integration software, has released webMethods 6, a standards-based, massively scalable integration platform designed to be the foundation for next-generation integration suites. The webMethods 6 release will serve as the anchor for a number of other extensions, making the webMethods integration platform the most comprehensive solution for building integration networks that enable companies to achieve an enterprise dial-tone.

Built on a 100% service-oriented architecture, webMethods 6 enables companies to leverage integration and Web services more widely and flexibly across the enterprise, allowing them to assemble solutions that meet their specific needs rather than being limited to the constraints of their original designs. Once these comprehensive business processes are created, webMethods 6 allows companies to correlate their overall IT infrastructure with the precise impact any element of that infrastructure will have on specific business processes.
www.webmethods.com

Continued from page 30

and is an excellent research and networking opportunity.

New books and research papers devoted to the subject appear daily. As developers continue to advance the field of Web services, more emphasis will be placed on the next generation of Semantic Web services.

Anticipate the Semantic Web When Designing Applications Today

Web services built today use WSDL to define their interfaces. In the future, you may need to support additional semantic interface definition languages. By anticipating the conversion and designing for extendability, the transition to Semantic Web services should be relatively smooth.

Play with the Technology

Many of the development tools available today are free, some as open-source projects. The W3C maintains links to many free tools, and commercial tools can be found at SemanticWeb.org (www.semanticweb.org).

Applications that demonstrate the business value of the Semantic Web and Semantic Web services are emerging. The TAP project (http://tap.stanford.edu) has created a demonstration semantic search application that gives a peek at the future of searching on the Internet.

Summary

The Semantic Web and Web services initiatives are closely related. Both share the high-level objective of making information on the current World Wide Web accessible and usable to software agents. Semantic Web services are the powerful combination of two evolving technologies.

References

- Ankolekar, A., et al., "DAML-S: Web Service Description for the Semantic Web," Proceedings 1st International Semantic Web Conference (ISWC 02), 2002.
- Peer, J. "Bringing Together Semantic Web and Web Services" Proceedings 1st International Semantic Web Conference (ISWC 02), 2002.
- Paolucci, M. et al., "Semantic Matching of Web Services Capabilities" Proceedings 1st International Semantic Web Conference (ISWC 02), 2002.
- McIlraith, S.; Son, T. C.; and H. Zeng. "Semantic Web Services". *IEEE Intelligent Systems*, 16(2):4653, 2001.
- McIlraith, S.; Son, T. C.; and Zeng, H. "Mobilizing the Semantic Web with DAML-Enabled Web Services." Proceedings Second International Semantic Web Workshop (SemWeb' 2001), 2001.
- Fensel, D. & Bussler, C. (2002), "The Web Service Modeling Framework (WSMF)". www.cs.vu.nl/~dieter/wsmf/wsmf.paper.pdf accessed on 8 Mar 2002.
- T. Berners-Lee, J. Hendler, and O. Lassila. "The Semantic Web." *Scientific American*, 284(5):34-43, 2001.

WSJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
.NET Developer's Journal	www.sys-con.com/dotnet	888-303-5282	37
Altova	www.altova.com		17
BEA Systems	dev2dev.bea.com/useworkshop		60
BEA WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	888-303-5282	25
DataDirect	www.datadirect-technologies.com/net		6
IBM	www.ibm.com/websphere/portalplay		2
Java Developer's Journal	www.javadevelopersjournal.com	888-303-5282	45
JDJ Store	www.jdjstore.com	800-303-JAVA	35
ObjectFocus	www.objectfocus.com	408-973-8200	53
Parasoft	www.parasoft.com/soapwsj12	888-305-0041	9
Sams Publishing	www.samspublishing.com		23
Sitraka	www.sitraka.com/jclassSS/ws	800-663-4723	13
Sitraka	www.sitraka.com/jclass/ws	800-663-4723	59
Sonic Software	www.sonicsoftware.com/websj		4
SpiritSoft	www.spiritsoft.com/climber	508-473-3227	15
SYS-CON Media	www.sys-con.com	888-303-5282	31, 47
SYS-CON Media	www.sys-con.com/suboffer.cfm	888-303-5282	32,33
SYS-CON Media	http://developer.sys-con.com	888-303-5282	41
Web Services Edge 2003	www.sys-con.com	201-802-3069	49
Web Services Journal	www.wsj2.com	888-303-5282	27
WebSphere Developer's Journal	www.sys-con.com	888-303-5282	43

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

The Secret Sauce

Anne Thomas Manes

Anne Thomas Manes is the founder and CEO of Bowlight, an analyst and consulting firm. Anne is the author of "Web Services: A Manager's Guide." She participates in Web services standards development efforts at W3C, OASIS, WS-I, and JCP. Before starting Bowlight, she was the CTO at Systinet, the Web Services Infrastructure Company.
atm@bowlight.net

How do you define a Web service? If you ask five people to give you a definition, you'll probably get at least six answers. Is a Web service any application that can be accessed over the Web, or is it limited to applications that expose a programmatic interface? Is it the code that implements the service or the interface to the code? Do you have to use SOAP? What about XML-RPC? Or RosettaNet? Or FIXML? Or some other XML protocol? And do you have to use XML? Does SWIFT qualify as a Web service?

I know that many people will disagree with me, but my basic definition is as follows: a Web service is a programmatic interface to some application function, and that interface communicates using an XML protocol – any XML protocol. As a best practice I recommend that you use a standardized or well-known XML protocol (e.g., SOAP 1.1), and describe your service using some type of standardized or well-known description language (e.g., WSDL 1.1). But as long as you're using an XML protocol, I would say that it qualifies as a Web service.

I draw the line at XML because I think that XML is the secret sauce that gives Web services technology its enormous power and flexibility. It is the Web's universal data language. Any application programming language can interpret XML. And Web services can run on any platform – from the largest mainframe to the smallest embedded device. Web services technology is the first distributed-computing middleware that is completely vendor and platform independent. This little feature is a huge win and by itself provides a very strong incentive to adopt Web services.

But there's another reason why you might want to adopt Web services: multichannel client support. If you think back to ancient history – to the days before the Web – I'll bet that you can recall discussions about three-tiered client/server computing. One of the core themes of three-tiered client/server computing was the clean separation of presentation logic and business logic. The idea was that you could build a business service, and you could access that service from both Windows and Macintosh clients. (Remember how hard it was to do something like that in 1992?)

Now you would most likely just build an HTML client that talks to the application through a Web server. But is that enough? Wouldn't it be nice if you could access your services from a wide

assortment of client environments: Windows, Macintosh, Linux, Unix, Palm OS, BlackBerry, a mobile handset, etc? And wouldn't it be nice to access them from Visual Basic, WinForms, Excel, reporting tools, business intelligence tools, WebForms, Flash, Java applets, and IVR applications? The requirement for multichannel client support is more critical than ever.

Enter Web services. Web services offer the ultimate in presentation and business logic separation. A Web service doesn't have a native human interface. It has a programmatic interface. You send it an XML message and it returns an XML message. Hence any client application that can process and interpret XML can speak to a Web service. And what's so wonderful about XML is that it's so malleable. XML can be validated and transformed on the fly. This little feature allows you to dynamically resolve any inconsistencies that might exist between a client and the service. You can also customize your messages for different types of clients.

XML can be transformed at runtime into a presentation markup language, such as HTML, WML, or VoiceXML. You can also transform the XML into binary formats to send to a device that doesn't understand XML. More and more GUI development environments, such as Microsoft WinForms/WebForms and Macromedia Flash, can consume and display the data from XML messages. The business intelligence vendors are adding inherent support for XML data sources. Microsoft's forthcoming XDocs product will capitalize on the power of XML to blur the distinction between client types, such as a spreadsheet, a document, or a form.

So if you're looking to cook up a nice little application that can be consumed by any possible client environment, don't forget the special sauce: XML. Build your application as a Web service