SYS-CON

NOW INCORPORATING XML JOURNAL

SOA Web Services Journal

February 2006 Volume 6 Issue 2

Federated Service Management

A federated approach to managing highly distributed Web services

pg. 10

Web Services Security

Don't forget security on the way out

pg. 18

Enterprise Data Integration

A critical piece of an SOA

pg. 24

Successfully Planning for SOA

Part 2 – Building your SOA roadmap

pg. 32

RETAILERS PLEASE DISPLAY UNTIL APRIL 30, 2006
$6.99US $7.99CAN
02
0 71486 03420 9


MAKE YOUR SOA SECURITY MOVES WISELY...
XWALL
WEB SERVICES FIREWALL
XRAY
WEB SERVICES DIAGNOSTICS
VULCON
VULNERABILITY CONTAINMENT SERVICE
SENTRY
SOA SECURITY GATEWAY
PUTTING TOGETHER THE PIECES FOR THE WORLD'S MOST DEMANDING SOA SECURITY SYSTEMS
FORUM SYSTEMS ENTERPRISE SOA SECURITY SOLUTIONS:
TRUSTED SOA MIDDLEWARE
WEB SERVICES SECURITY
XML ACCELERATION
WWW.FORUMSYSTEMS.COM
FORUMSYSTEMS
THE LEADER IN WEB SERVICES & SOA SECURITY

SOA Web Services JOURNAL

February 2006 Volume 6 Issue 2

Inside WSJ

XML JOURNAL

SOA Web Services Journal

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

The Backdoor

WRITTEN BY
SEAN RHODY

People who know me would generally agree I'm a straight-forward guy – I pretty much just like to move in the direction I've said I was going, rather than try to move from side to side and finesse something. So when it comes to technology, I tend to like to go with technology because it makes sense, and I usually assume that most IT organizations work that way as well.

But when you look at a technology like Business Process Management (BPM), you can see that the straightforward approach may not be the best, fastest, or even most successful route towards deployment.

BPM is a tougher sell on a straight technology basis, because it relies either on an SOA or an EAI environment that enables a service approach, and because the capabilities it provides have to date been implemented, albeit poorly, in actual code.

BPM as a technology extracts the business rules of an organization using advanced modeling techniques and software to define the business rules, the "what happens when" outside of lower level code. Besides allowing for rapid change in response to changing business conditions, BPM also allows the business community to take a much greater role in the definition of behavior within their software environments.

Clearly, this type of capability can be an asset to an organization that is confronted with frequent changes, dynamic market conditions, or even the consequences of a merger between organizations with disparate computing systems. Yet, because of the nature of the way IT projects are usually funded, this capability is frequently a difficult sell.

Most IT organizations have to fund their projects as discrete systems, therefore you can fund a CRM system, or an Order Management system, or a General Accounting system, even a User Portal. Each of these systems provides an "end user" benefit, one that can be easily quantified and budgeted for. BPM, by contrast, potentially cuts across all of these systems, while providing little visible or tangible benefit.

That's at least partially because funding a development effort and cost usually neglects the operational and maintenance cost of a system. These costs can often be multiples of the original implementation cost over the lifetime of a system. As an example, think of some of the COBOL programs that many large organizations have been nursing along for decades. Compared to the cost of creation, the maintenance costs are many times higher.

This is where the BPM solutions pay – they help reduce operational and maintenance cost. Anything that is programmed has to be tested to death, deployed, and managed. The model-driven architecture (MDA) approach seldom actually works all the way down to the code level and back again, so even if there is some modeling or design, it's typically only documentation when the coding gets done, allowing errors and omissions to creep into the process and creating troubleshooting nightmares.

In contrast, BPM presents the rules in a modeling environment that is completely round trip, and can be tested and debugged more effectively, especially in the difficult cases where a business transaction requires crossing system boundaries. We've all experienced the "he said, she said" finger pointing that goes on when a process that spans two or more systems experiences difficulty. BPM reduces cost by taking the management, the modeling, and the implementation out of multiple silo-based systems and centralizing the capabilities needed to effectively implement business processes rapidly.

It should not be surprising that the calculations necessary to quantify this benefit are convoluted and involved. They require analysis of maintenance and operations, as well as a good understanding of the actual software development life cycle in use in a particular organization – something that is seldom present. Thus while the technology clearly provides benefits, quantifying its value and justifying its cost remain elusive. In the end, the straightforward approach to the problem, which is simply stating the need for the capability, must give way to a more devious approach that builds the capability into the price of one or more system upgrades or packaged implementations.

About the Author

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

sean@sys-con.com

SOA Web Services Journal

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

ADVERTISING
SENIOR VP, SALES & MARKETING
Carmen Gonzalez carmen@sys-con.com
VP, SALES & MARKETING
Miles Silverman miles@sys-con.com
ADVERTISING DIRECTOR
Robyn Forma robyn@sys-con.com
NATIONAL SALES & MARKETING MANAGER
Dennis Leavey dennis@sys-con.com
ADVERTISING MANAGER
Megan Mussa megan@sys-con.com
ASSOCIATE SALES MANAGERS
Kerry Mealia kerry@sys-con.com

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

CUSTOMER RELATIONS/JDJ STORE
CIRCULATION SERVICE COORDINATORS
Edna Earle Russell edna@sys-con.com

sys-con.com
CONSULTANT, INFORMATION SYSTEMS
Robert Diamond robert@sys-con.com
WEB DESIGNERS
Stephen Kilmurray stephen@sys-con.com

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

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

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

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

SYS-CON MEDIA

Architecting for SOA

WRITTEN BY
AJIT SAGAR

The term "architecture group" is a heavily loaded one. I've run into different scenarios at the various clients that have engaged us for consulting on their architecture strategy. In some cases, we have been asked to help seed and grow such a group. In other cases, we've been asked to put together plans to define the organization of an architecture group. And sometimes, we just supplement the existing architecture group.

The SOA and Web services arena is fairly new, and one that warrants the formation of a well defined horizontal group that includes several functions under it. Architecture governance is an area that has always been the charter of architecture bodies in large organizations, but with the advent of SOA, architecture governance has become a much more formalized function. This is because the basic entity of an SOA – the *service* – has formally defined contracts and service-level agreements (SLA); hence, the formation of the architecture that provides services requires strict practices to ensure that the SLAs are met.

I went through an interesting discussion with a colleague on the nature of an architecture group. One view of the architecture group is that the charter of the group is only to define the architecture and to produce the artifacts that support the architecture definition and usage in the form of best practices, architecture and design patterns, and guidelines. This may also include direction on the approved list of vendor products (and versions) that should be used for application development throughout the organization. Another view of an architecture group is that the charter of the architecture groups includes the development of a reference architecture, common components, and other reusable services that supplement the features provided by the third-party products that make up the reference architecture platform. An architecture group also acts as a consulting body that assists projects through mentoring, review, and sometimes also in the development of the application.

I am a firm believer in the second model. To me, documenting architecture is just a function of the group. Other functions include the development and maintenance of actual components and services that are shared across applications. This allows the group to function more efficiently, provide greater value, and not be perceived as an "elitist" group that is aloof from the application portfolios.

As companies adopt SOA, to succeed in the long run it is imperative for them to create an appropriate architecture group that focuses on the best practices for adopting the various facts of service enablement. Some of the key issues that need to be tackled by the SOA architecture group include adoption of new technologies and products, definition of basic components and services, definition and promotion of SOA standards, setting up of an efficient process for application development, review, governance, and so on. One of the main responsibilities is also to align with the business to define the IT strategy for SOA and to provide an implementation roadmap, including the migration of existing applications towards a service-oriented paradigm. To achieve all of these objectives, the architecture group has to be steeped into the practical aspects through creation of reusable components and collaborative interaction with the application portfolios to march toward a common goal.

About the Author

Ajit Sagar is a principal architect with Infosys Technologies, Ltd., a global consulting and IT services company. Ajit has been working with Java since 1997, and has more than 15 years experience in the IT industry. During this tenure, he has been a programmer, lead architect, director of engineering, and product manager for companies from 15 to 25,000 people in size. Ajit has served as *JDJ*'s J2EE editor, was the founding editor of *XML Journal*, and has been a frequent speaker at SYS-CON's Web Services Edge series of conferences. He has published more than 100 articles.

ajitsagar@sys-con.com

Mindreef® SOAPscope®

Take Control

of your

Web Services

Developers • Testers • Operations • Support

Web services and SOA have changed the rules with the introduction of a new XML abstraction layer.

Though most development organizations have tools and processes for managing code objects, they have little or no help for testing, diagnosing, or supporting the XML portion of their Web services.

Mindreef SOAPscope completes your Web services development toolkit by combining XML-aware tools for developers, testers, support, and operations. This flexible combination gives you the right tool for any diagnostic task, whether working alone or collaborating with other team members.

Start taking control of your Web services by downloading a FREE trial version of Mindreef SOAPscope today!

Mindreef SOAPscope: Named best Web services development tool in the 2005 InfoWorld Technology of the Year awards

Test *No coding required!*
Test service contracts and underlying service code with an easy-to-use, forms-based interface

Diagnose *No more wading through angle brackets!*
Find the cause of a problem with powerful message collection, analysis, and diagnostics tools

Collaborate *No more emailing XML snippets!*
Share test or problem data with others, regardless of their roles or system requirements

Support *No more finger pointing!*
Web services require support at the API level – a real burden for most organizations. Mindreef SOAPscope makes it easy by allowing customers and support staff to share Mindreef package files that contain complete problem data

Download a free version of Mindreef SOAPscope at www.Mindreef.com/tryout

Mindreef®
www.Mindreef.com

Federated Service Management

A federated approach to managing highly distributed Web services

Service-oriented architecture (SOA) has emerged as a key strategy for IT and line-of-business executives to jointly enhance business performance and agility in today's intense corporate climate. Using the SOA methodology, business applications are built as an assembly of loosely coupled pieces of business functionality, commonly referred to as services. These services are published, consumed, and combined with other applications over a shared services network, which is often highly distributed within and across enterprise boundaries.

WRITTEN BY
ARUN CANDADAI & ROBERT McKENNEY

Service-oriented development of distributed applications is increasingly common as businesses seek to build and reuse services and service-based processes in new ways to improve performance and gain competitive advantage. Gartner Group has coined the term service-oriented business application (SOBA) to refer to a service-based approach to building distributed applications.

There are two basic entities in the development of SOBAs – service provider and service consumer. The service provider offers services that the consumer uses and composes into a distributed application that solves a particular business problem. Once the distributed application is established, it can in turn be offered as a service for another service consumer to integrate.

When managing services, most businesses expect management functions to be performed at the service endpoints by a provider that controls the platforms on which the services are hosted. Because a SOBA can be composed of highly distributed and heterogeneous services from multiple providers, it is likely that many of the endpoints will be outside the control of the consumer. Yet, service consumers must still apply governance principles to their distributed applications or risk offering an unreliable and underperforming service to their business users.

To provide proper governance, the service consumer needs a way to apply its management policies and logic to the services it consumes, regardless of the endpoints that provide them. Stakeholders across the enterprise need a unified, yet simple, operational approach to service management that provides capabilities such as:

- Support for large numbers of heterogeneous service interactions among services with diverse characteristics and management policies
- Separation of business logic from management logic concerns
- Separation of governance responsibilities between developers and operational administrators
- Provide a view of service behavior in an end-to-end business context
- Ability to monitor and control service behavior at run time

The only way to holistically manage autonomous, disparate, and highly distributed services is to employ a federated approach to service management. Within a federated system, the behavioral aspects of services must be virtualized in a unified manner from the standpoint of the applications consuming the services. This approach provides a service

consumer with a standardized way of enforcing policies, best practices, and other management logic at the composite business service level, which is fully compatible with existing management logic at the service endpoint.

When properly implemented, federated service management (FSM) provides a unified method for governance among applications, services, and devices across a heterogeneous network, which will enable developers to easily create solutions for complex distributed environments. The end-to-end view provided by FSM allows service consumers and providers alike to understand in real time the performance being delivered, and to take appropriate actions to rectify issues related to service quality and SLA compliance.

The Problem

As businesses seek to expose their business processes as services and integrate them into SOBAs, it is not surprising that service management is not the first thing that comes to mind. The shift to a service-based concept of business processes and applications is fairly significant in and of itself; hence, most businesses are wrapped up in determining the granularity of their services and how to secure them. When they do consider service management issues, it is almost always from the service-provider viewpoint. This limits them to managing only the services that they own and control, which often leads to an incomplete management solution.

Viewing governance and management as a service-provider issue has led businesses and third-party vendors to develop platform and container-dependent solutions, which may not be adequate for managing increasingly distributed services across multiple provider domains and hosted on different operating platforms. This management approach typically uses static, platform-specific code deployment to manage services, often mandating that businesses install multiple distributed components (usually from multiple vendors) that result in heavy and expensive systems, which are difficult to maintain. It also does very little to address the fact that management logic for the services usually remains fragmented and service-specific at best.

Businesses seeking to improve their service management capabilities and create a unified management solution must recognize and overcome three governance challenges:

- First, rapidly expanding service networks will be too complex to manage on a point-to-point or ad hoc basis
- Second, the distributed nature of service development and usage within an enterprise makes it difficult to control quality and apply standards
- Finally, perpetuation of code-driven service management will render the architecture too rigid to deal with changing business requirements

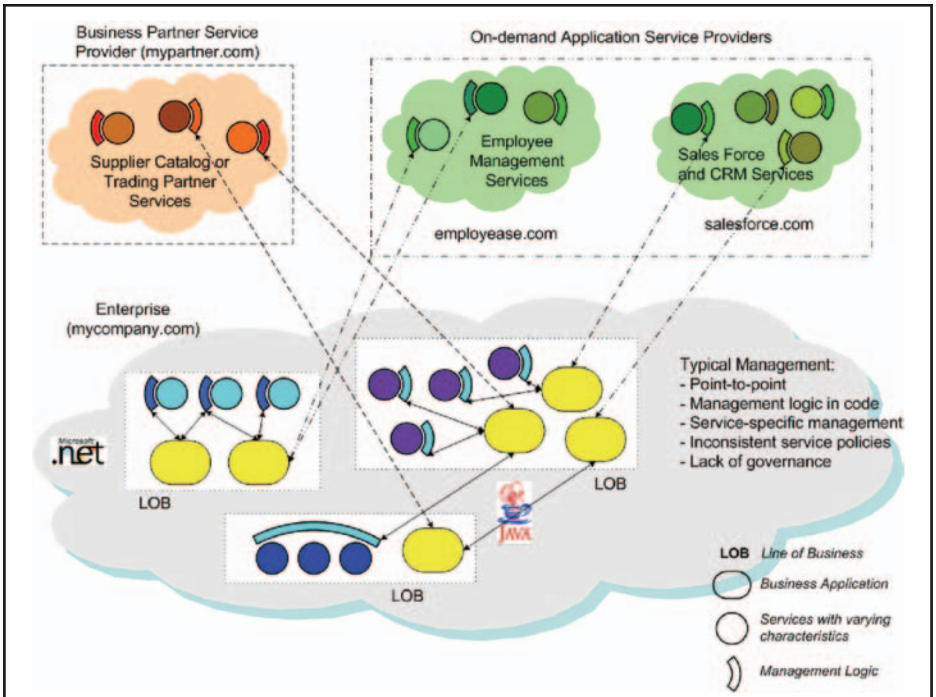


FIGURE 1 **Problem – current approach to service management**

Challenge 1: Service Network Complexity

To develop robust SOBAs, businesses must grow their service networks, expanding them across divisional and enterprise boundaries. Most businesses rely on a point-to-point management approach that usually involves implementing custom management logic (for security, message logging/auditing, performance and SLA monitoring, etc.) applied differently for each individual service. This approach may seem satisfactory when the number of services is small, but as the number of service consumer-provider interactions grows exponentially, it becomes a serious impediment to scaling up.

Challenge 2: Static, Code-Driven Management Architecture

Business agility can roughly be measured by the cost associated with changing its business processes. When SOBAs are managed by embedding custom management logic within the service code, it forces management changes to be static, thereby driving up the times and costs required to implement changes. Further, such changes make SOBAs brittle and less agile due to cascading effects and implementation-time delays across the service network.

Challenge 3: Fragmented Management Logic

As services are integrated over multiple administrative domains, it becomes very difficult to define, apply, and enforce configurations and policies in a consistent manner. Each provider develops and maintains its services according to its own goals and standards, perhaps using different approaches to solve the same issues. This fragmented approach makes it harder to enforce consistency and integrity of service configurations because administrators are inadvertently shielded from getting a complete view of the management logic embedded in the service.

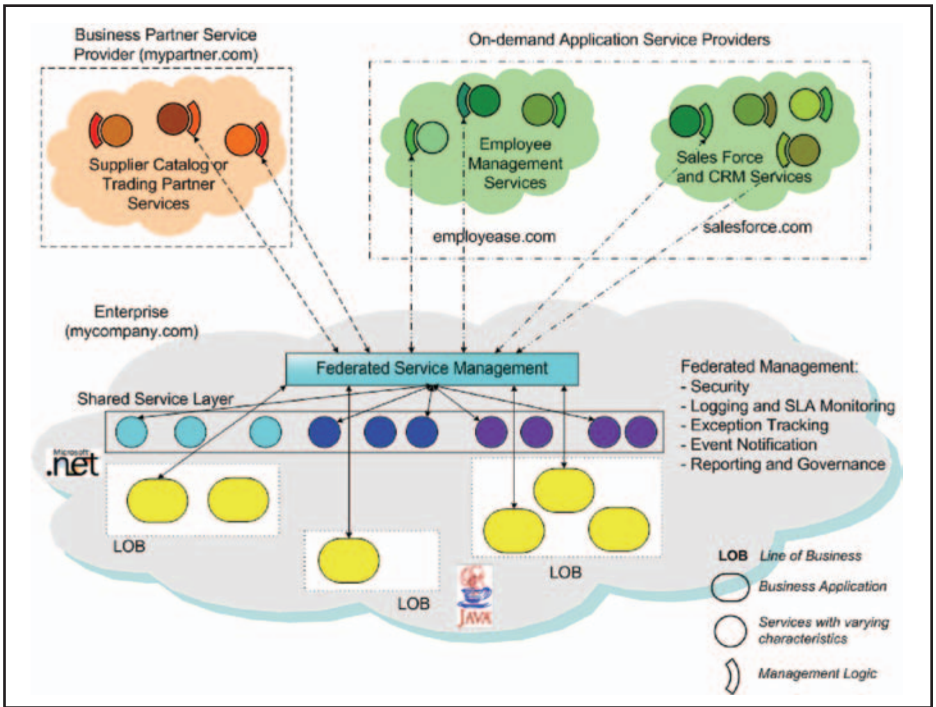


FIGURE 2 **Solution – federated approach to service management**

Consequences of Inadequate Service Management

If the aforementioned service management–related issues are not addressed, they can significantly undermine any value a business may be seeking from developing SOBAs. More important, they can negatively impact the overall business in the following ways:

- Service/operational administrators confuse effort with results because managing at the service endpoints doesn't provide sufficient information to make management decisions for the composite business applications as a whole. For example, when services are managed at the endpoint, SLA compliance is tracked at the endpoint. This provides a skewed view of service performance because it does not fully account for network or other latencies. In this case, the SLA may be fine from the provider's viewpoint and yet fail to meet the consumer's expectations.
- Brand image is tarnished by unpredictable service quality resulting from inconsistent development standards, mismatches in performance expectations, and inability to adapt management logic at run time.
- Business is slower to respond to competitive threats and changes in the marketplace because the higher cost of changing management policies makes it willing to accept greater risk to justify changes.
- Lack of a unified service management approach leads to redundancies and unnecessary development efforts.

The Solution

What is ultimately needed is a way to enhance SOBAs with unified, end-to-end service management and run-time governance. To meet this need, a solution must enable service consumers to federate the management of services they consume. Federating service management means recognizing and understanding the endpoint-management constraints specific to the services, virtualizing the functional and nonfunctional aspects of the services, and applying overarching policies and standards to govern how the services interact with each other and the application. This unified approach empowers service consumers with control over their business performance and enables them to monitor SOBAs in an end-to-end fashion.

Business Value Drivers

A well-designed FSM solution must focus on addressing the key business value drivers for the enterprise: visibility, flexibility, and accountability. SOBAs that are built from highly distributed services demand a management solution for services that cuts across divisional and enterprise boundaries in a seamless manner:

- Visibility represents a business's ability to have real-time answers to the what, when, where, why, and how of various service activities and events that occur. However visibility is about more than a simple registry of WSDL files and some service message logging. Visibility exposes all of the service metadata – the configurations and policies of the services. Real visibility takes this metadata beyond the developer level and design-time environments into the run-time environments of administrators and other operations-level users who need to see and influence all aspects of service behavior.
- Flexibility represents a business's ability to adapt quickly to changing requirements without being constrained by costly, time-consuming code updates or proprietary standards. Achieving real flexibility means lowering the cost of change in how services are configured to an absolute minimum. The most costly changes are ones that require the service to be redesigned. Much like visibility, real flexibility requires freeing management changes from the design-time environment and enabling them in a run-time environment. This gives administrators and business managers the opportunity to act on the information gained from improved visibility.
- Accountability represents a business's ability to monitor and control the relevant services – within the overall business context of the composite SOBA – in order to enforce performance standards and to deal with exceptions. When services interact across multiple domains, the diverse combination of "local" standards and development practices makes it important to accurately identify the source of performance problems or exceptions. Real accountability means being able to focus management activities on the individual services within the integrated SOBA and enforce unified standards in an end-to-end context, regardless of how the services are managed at the endpoint.

Solution Approach

Before crafting and using a federated management solution, a business must establish a

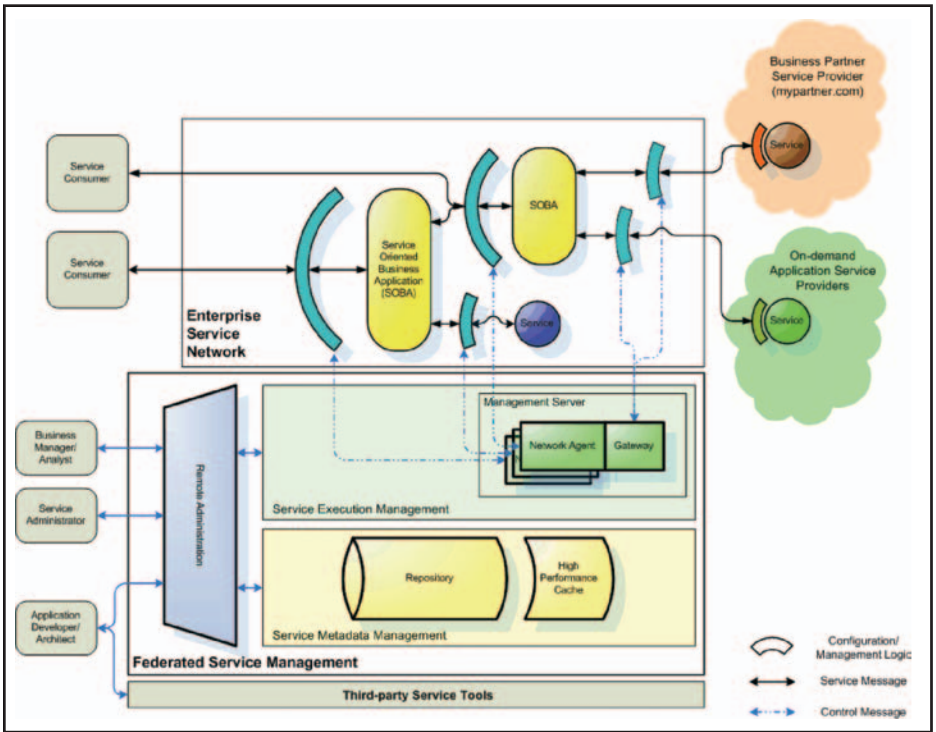


FIGURE 3 **Technology architecture for FSM**

solid foundation of development and management practices for their services. To maximize visibility, flexibility, and accountability, it must avoid practices that embed management logic within service functions and instead externalize management to make it more accessible at the operational level. The proposed FSM solution is based on the following core tenets:

- Separate and decouple management logic from business logic and externalize management code from within the service. This cleanly separates development and operational concerns and allows management logic to evolve independently from the business logic.
- Relieve service developers from responsibility for management logic to free up time required for building better business processes. Developers are hence shielded from routine service management tasks that are best externalized as declarative tasks performed by operational administrators.
- Develop a configuration-based approach to service management that can adapt to changing number, complexity, and behavior of services. Use declarative logic to create configurations for services and their associated policies and rules.
- Make service management a pure operations-level task and empower administrators and business analysts to configure, monitor, and control the business services.
- Implement a common service management layer to streamline the development process and accelerate migration to an enterprise-wide SOA.

Key Solution Capabilities

The list below identifies a partial list of capabilities for the proposed solution:

- Discover and store service definitions for services across divisional and enterprise boundaries
- Define and store configurations and policies related to services, performance metrics, and SLAs
- Cache service configuration and policy metadata to minimize processing times
- Automated search, discovery, and update of services, including the ability to batch load services
- Dynamic run-time execution of management policies
- Search, filter, and view details on any service, activity, message, organization, exception, or alert
- Monitor, track, and audit key service activities and performance metrics
- Identify, intercept, and handle exceptions
- Define and generate alerts based on service policies and SLAs
- Define and generate reports on service operations and performance
- Provide a holistic, 360-degree view of services

The Technology

Key Technology Characteristics

When settling on implementation details, solution designers need to consider the key technology characteristics that will enable them to use FSM to achieve their goals. First and foremost, it should be standards based. Building on 100 percent Web service and open source standards ensures maximum compatibility and avoids the potential for lock-in to proprietary software. Along with standards-based development, the solution should strive for platform-independence and support multiple deployment models to maximize flexibility.

In addition, a FSM solution should be lightweight with a small footprint, and updates, where feasible, should be performed without server restarts or downtime.

Architecture Framework

The proposed technology architecture is based on a "management via mediation" approach in which a management intermediary sits between client applications and endpoint services in order to manage the service interactions. Because the service endpoints could be anywhere, the intermediary usually is located closer to the client applications and business processes. This architecture enables: 1) managing the service metadata via a repository, 2) managing service execution using an intermediary to apply management logic at run time, and 3) remote administration via a console to provide enterprise-wide command and control of services and related activities.

Discovery and Configuration

To provide federated management and governance for SOBAs, the execution layer needs to know more than where the service is and how to invoke it. It needs to identify and store critical information necessary for

> "A well-designed FSM solution must focus on addressing the key business value drivers for the enterprise: visibility, flexibility, and accountability"

defining and configuring services – protocols, policies, SLAs, etc. The set of nonfunctional characteristics defined in the service metadata can be referred to as a service configuration. Configurations can be derived from an extended service definition, when available. Storing service-related metadata as configurations in a repository provides the foundation for monitoring and controlling those services. A repository should ideally include the capability to identify and register services in an automated fashion. As part of its discovery function, a federated repository should be able to search public and private registries to identify new services as well as updates or new versions of existing services.

The repository can be implemented as a high performance data store. In addition, the repository should leverage intelligent caching techniques to deal with compute and network latency issues, and improve management logic processing times.

Management Logic Mediation

To control services in a federated manner, a business must have the flexibility to dynamically define and execute management logic at run time. This can be accomplished by using an intermediary to mediate the management logic into the service process. An advantage to using intermediaries to manage services is that it shields service consumers from any changes in how services are deployed and managed at the endpoints. Essentially, the intermediary serves as a management execution layer that unifies management logic for all of the services used.

The intermediary can operate in two modes:

- ***Management Gateway:*** A gateway provides a single, well-known access point and serves as a proxy between service consumers and service providers. Connecting to a gateway is as simple as changing the access point from the endpoint service binding address to the gateway's address. This mode works well as the intermediary between SOBAs (and other consumers internal to the enterprise) and external service providers.
- ***Management Server:*** A management server acts as a central server to multiple agents distributed across the service network. This mode works well when performance considerations owing to client-intermediary communications latency outweigh the need for zero-installation management software on the various nodes of the service network.

The actual intermediary mode of choice depends on the architectural requirements of the business. In either mode, a management intermediary establishes control and performs its mediation functions by forming a virtual pipeline of compartmented management logic dedicated to various management functions (such as message logging, security, caching, and performance monitoring). The pipeline is designed as a set of request handlers that sequentially handle the request and response message flows. At the center of the pipeline is the service access logic. Each handler dynamically executes management logic from the appropriate configurations and policies stored in the repository.

The management intermediary can be implemented as a Web service and scaled horizontally across multiple servers, offering a high degree of scalability and performance for the overall management layer.

Administration

An administration console is a centralized interface to provide a common point of control for configuring and monitoring all services. The console enables administrators and business managers to define, configure, and monitor business services and other related resources. It provides administrators, developers, and business managers with a means to review services and modify service behavior dynamically and in real time. It also provides powerful dashboard views for monitoring the service network.

A console can be delivered as a simple, Web-based application that can be accessed remotely without any special software installation requirements.

The Benefits

The proposed FSM solution provides business benefits at many different levels. By externalizing the management logic out of the business logic and virtualizing the creation and configuration of business services, an FSM solution reduces IT costs and complexity, increases flexibility and control, and enhances accountability of all services under management. The following are the benefits of using an FSM solution organized by beneficiary groups.

Developers

- Improve service quality and develop SOBAs faster by focusing on business process logic rather than on management logic
- Save times and costs associated with managing services by automating routine steps
- Increase reuse opportunities and reduce service development costs by storing service artifacts in a dynamic repository

Operations Administrators

- Define, control, and fine-tune various services and their associated configurations in real time
- Streamline administration requirements by eliminating the need to deploy fragmented pieces of management code
- Monitor service activities in real time using the activity dashboard

- Review alerts and exceptions in real time, and act on them immediately and in an automated fashion

Line of Business and IT Managers
- Improve service quality and accelerate roll out based on unified design standards and policies across the enterprise
- Enhance customer and end-user experience by using service performance metrics and taking appropriate actions in a timely manner
- Improve regulatory compliance by providing an accountable system for tracking and monitoring services and activities within the service network in a federated manner.

Conclusion

SOA is a key strategy for improving the agility of IT investments in today's dynamic business environment. Service-oriented business applications (SOBAs) of the future will be composed using highly distributed services spanning a wide array of providers and possessing diverse characteristics. The typical approach to service management assumes full control over the services being used, is point-to-point in nature and relies on service-specific management logic, and is static due to code-driven logic. This approach is ill equipped to deal with the growing complexity of highly distributed service networks and the ever-changing landscape of business requirements.

A solution to these challenges is a federated approach for managing services through a unified management layer. Federated service management (FSM) involves virtualizing the behavior of the services and applying overarching policies to govern service interactions in a dynamic manner. The proposed FSM solution unifies highly distributed autonomous services and provides a mechanism to dynamically configure, monitor, and control them efficiently. With a FSM solution in their SOA arsenal, enterprises will be well equipped to handle the complex and dynamic nature of the business challenges of the future.

About the Author

Arun Candadai is cofounder and CTO of GridScope (www.gridscope.com), and is responsible for the company's technology strategy and product development. Prior to founding GridScope, he was lead architect at BEA Systems where he pioneered the use of Web services and SOA for its enterprise infrastructure. Prior to BEA, he held senior engineering management roles at companies including Asera, Covad, and Lockheed Martin. He holds three software patents and has helped develop key Web services standards.

arun.candadai@gridscope.com

Robert McKenney is cofounder and VP of Product Management of GridScope (www.gridscope.com), and is responsible for the company's marketing strategy and product positioning. Prior to GridScope, he directed the product management at Internet software startups such as Integres and PostX. Prior to that he held senior product and program management roles at AT&T Wireless and the US Department of Defense.

robert.mckenney@gridscope.com

Web Services Security – Don't Forget Security on the Way Out

Security requirements for a Web service that are often overlooked

Typically when we think about security for a Web service, our focus is on how to protect it from unauthorized and malicious users. Thus, we tend to concentrate on such things as authentication of the requestor, checking to see that the requestor is authorized to access the service, validation of the request message, and so forth – all things that happen on the way in or during a request for the service. However, there is an equally important set of security functions that need to occur on the way out or after the service has finished processing the request.

WRITTEN BY
TIEU LUU

Because these security requirements are not as apparent when thinking about Web services security, they tend to get built in as an afterthought without as much design and consideration as is necessary. This article describes some of the important security requirements that need to be considered beyond the typical steps that occur during a service request. In this article we'll refer to these steps as "security on the way out."

Typical View of Web Services Security – Only Half of the Picture

Figure 1 shows a simplified view of what we tend to concentrate on when thinking about Web services security. A requestor issues a request for a Web service in the form of a SOAP message. As part of this message, there are claims about the identity of the requestor and any associated attributes. When the Web service receives this request, it first verifies the identity of the requestor. Once the identity is verified, it compares the attributes associated with the identity against those referenced in its security policy to determine if the requestor is authorized. Next, it validates the request message, verifying any signatures, validating the XML, and so forth. When all of this is successful, the request is passed to the service implementation for processing. Although accomplishing all of this is by no means trivial, this is only half of the picture.

The Complete Picture

Figure 2 represents a more complete picture of Web services security, showing all of the steps that occur on the way in as described above, plus all of the steps that may need to occur on the way out. The following are some of the steps:

- Filtering the content of the response message
- Signing the response message (or selected elements of it)
- Encrypting the response message (or selected elements of it)
- Inserting any additional security tokens

The next several sections describe each of these steps in more detail, including the motivation for why they may be needed.

Content Filtering

In many cases, even though the requestor may be authorized to access a service, it may not be authorized to access all of the data that is produced or retrieved by that service. Thus, the contents of the response will need to be filtered to return only the data that the requestor is authorized to access. For example, within the Department of Defense (DOD), a requestor may invoke an enterprise search service to find information related to biological weapons. That search service may find a set of documents related to biological weapons, some of which are classified as Secret and some classified as Top Secret. If the requestor only has a clearance level of Secret, then the search service will need to filter out the documents classified as Top Secret before returning the response.

Ideally, this type of functionality should be handled in the security infrastructure, separate from the implementation of the service. One approach may be to use a message handler that intercepts outbound messages from the Web service and uses XSLT to filter out content that the requestor is not authorized to see. One drawback to this approach is that it may not be the most efficient approach. In some cases, it may make more sense to filter out the content before everything is converted to XML. In other cases, this may not even be feasible. Considering the DOD example from above, if the schema definition for the response data does not include an element or attribute that specifies the classification level of the data, that information would be lost once the response is converted into XML. Thus, the message handler would not have enough information to properly filter the data. In such cases, it may be necessary to have a post-processing security layer within the service implementation to perform the filtering before the response is converted into XML.

Message Signing

There are two purposes for digitally signing the response message: for nonrepudiation and protecting the integrity of the message. These are typically driven by business or legal agreements that require the response messages to be digitally signed to protect both the service provider and consumer.

For example, the accounts payable department of a large organization may provide a Web service for its partners to submit invoices for payment. The implementation logic of this service will need to verify and approve the invoices that it receives. Upon verification and approval, this service will need to digitally sign a response message that is sent back to the partner indicating that the invoice has been received, verified, and approved. If there is ever any type of dispute over late or missing payments, the partner will have a digitally signed response proving that the organization did receive and approve the invoice it sent.

In some cases, the total amount due that is specified on an invoice may not be the total amount that is approved for payment, perhaps because not all goods were delivered or some were defective. Whatever the case may be, the response from this invoicing service may contain an element that specifies the actual amount that was approved for payment. The service provider may want to sign that element of the response message to protect its integrity – so that it can't be modified without detection. This prevents an unscrupulous partner from disputing that a greater amount was approved for payment.

Digitally signing the response message should be handled in the security infrastructure. The implementation should use the W3C specification for XML Signature [XMLSIG]. This is a formal recommendation from W3C that describes the syntax and processing rules for digitally signing selected elements in an XML document. The Web Services Security (WS-Security) standard from OASIS describes how XML Signature can be used to secure SOAP messages and how the signature should be attached to the message headers [WSSEC]. Following this standard allows loosely coupled service providers and consumers to independently sign and verify the SOAP messages that they exchange. Most Web service security products available today provide an outbound message handler that can be plugged into the service and digitally sign the XML response message. Some vendors offer dedicated XML hardware appliances that can "sit in front of" the Web service and intercept inbound and outbound messages and digitally sign and verify the XML messages.

Message Encryption

The reason why message encryption may be needed on the response is pretty obvious – the response may contain sensitive information that only the intended recipient should see. With traditional applications, confidentiality is typically provided through transport and network-level mechanisms such as SSL and TLS. In a SOA Web services environment, such mechanisms may not always be adequate or feasible. Although not explicitly shown in the Figures 1

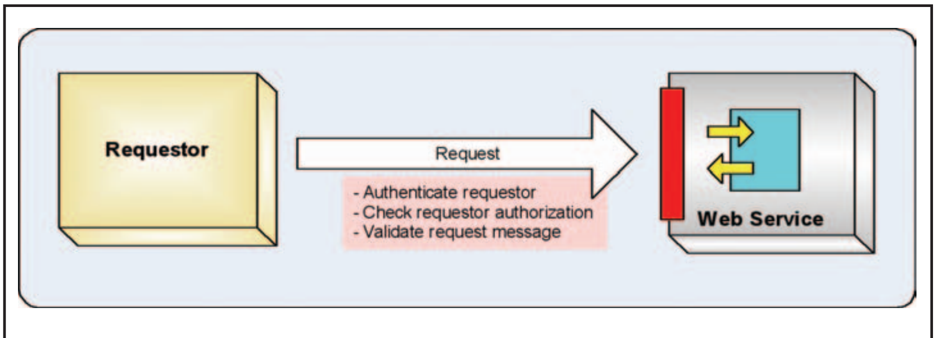


FIGURE 1 **Common, but incomplete view of Web services security**

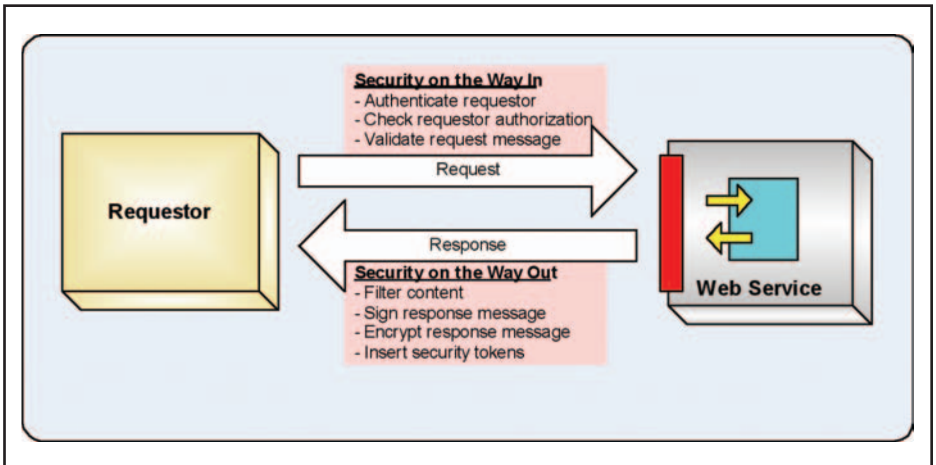


FIGURE 2 **The complete picture of Web services security**

> "Keeping this in mind when designing the security for your SOA will help to ensure that you're not just supporting half of the requirements"

and 2, the response message may go through several intermediaries before it is returned to the requestor or the next target service in a process flow. Some of these intermediaries may be trusted and some may not be. Mechanisms such as SSL and TLS are point to point, so the intermediaries will be able to see the response messages unencrypted. For untrusted intermediaries, it is obvious why this is a problem. For trusted intermediaries, there may be elements of the message that they should be able to access and still other elements that should be kept confidential from them. Thus, message-level encryption is needed for selectively encrypting certain elements of the response message and to prevent exposure of the response message to untrusted intermediaries.

Message encryption should be performed after the response has been converted into an XML message. Similar to message signing, this makes it ideal to be performed in the outbound message handler of the security infrastructure. WS-Security describes how XML Encryption should be used to encrypt any section of the body or header of a SOAP message. XML Encryption is another W3C recommendation that specifies a process for encrypting XML data and representing the results in XML [XMLENC]. Using these mechanisms essentially allows encrypting of the response message at the application level, thus avoiding the aforementioned problems. Just as with message signing, most Web service security vendors provide message handlers or XML appliances that can perform the message-level encryption and decryption so that developers will not need to build it into the service's implementation.

Insert Security Tokens

With the composite nature of SOA-based systems, the flow usually does not stop at a single Web service. That service may need to invoke another service or its response may be returned to an orchestration engine that uses the contents as input to the next service in the process flow. In either case, the service may need to forward along the security tokens from the original requestor or it may need to insert some new tokens with information about the service provider. A security token is just a collection of claims or statements (usually about the identity, privileges, group memberships, etc.) about a subject, which in this case can be the Web service itself or the original requestor of the service. Adding security tokens to the outbound message (either a response or a new request for another service) from a Web service allows the next target service in the flow to authenticate who is making the request and to make an authorization decision.

Adding security tokens to an outbound message can be supported by having an outbound message handler that intercepts the message on the way out and inserts the tokens into the message headers. Unfortunately, the heterogeneous environment in which a Web service typically operates often adds extra complexity. Security tokens can be expressed in multiple formats such as X.509 certificates, Kerberos tickets, or SAML assertions. In a heterogeneous environment there may be multiple interacting domains with different security infrastructures, and they may not all support the same format for security tokens. Thus, the security token inserted by the outbound message handler may not always be interoperable with the security infrastructure of the next target service. Web Services Trust (WS-Trust) is a set of specifications, proposed by vendors such as IBM, BEA, and Microsoft, which describes how security tokens can be issued and exchanged through a trusted authority to enable interoperability across security domains [WSTRUST]. WS-Security defines how security tokens should be represented and attached to the headers of a SOAP message. Using a Web service security product that supports both WS-Security and WS-Trust may help to alleviate some of the complexity involved with supporting this requirement.

Summary

In this article we discussed security requirements on the way out from a Web service. Such requirements typically don't get as much consideration when we think about Web services security. We discussed content filtering, signing the response message, encrypting the response message, and inserting security tokens. These requirements are almost identical to the steps that a client performs when sending a request to a service. The same technologies and standards are used to support both. Fundamentally, this is because the interactions between Web services and their requestors resemble more of a peer-to-peer model instead of the traditional client-to-server model. From a security perspective, this means that there is almost always a parallel set of security processing steps that needs to occur in the other half of the interacting peers. Keeping this in mind when designing the security for your SOA will help to ensure that you're not just supporting half of the requirements.

References

- *[WSSEC] Web Services Security: SOAP Message Security 1.0 (WS-Security 2004):* http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf
- *[WSTRUST] Web Services Trust Language (WS-Trust):* www-128.ibm.com/developerworks/library/specification/ws-trust/
- *[XMLENC] XML Encryption Syntax and Processing:* www.w3.org/TR/xmlenc-core/
- *[XMLSIG] XML-Signature Syntax and Processing:* www.w3.org/TR/xmldsig-core/

About the Author

Tieu Luu is an associate with Booz Allen Hamilton where he designs enterprise data and integration architectures for large federal agencies. Prior to Booz Allen Hamilton, Tieu held lead engineering positions at companies such as Grand Central Communications, Mercator Software, and Aether Systems, where he worked on the development of integration and mobile computing platforms.

luu_tieu@bah.com

EARLY BIRD SAVINGS: Register by February 27th and SAVE!

OMG ANNOUNCES:

SOA, MDA and Web Services Workshop:

Integrating the Enterprise and Beyond

March 27–30, 2006 • Fairfax, VA

Building on the success of last year's event, Object Management Group™ (OMG™) is pleased to host this workshop featuring presentations from architects, developers, vendors, and end-users of SOA/WS systems, many designed and built using the Model Driven Architecture® (MDA®).

Hear industry experts detail best practices at every stage of the application lifecycle — from modeling for requirements gathering and design, through WS and SOA for implementation and deployment, to the latest in security and semantic interoperability. Case studies by early adopters and presentations by tool developers will show how SOA, WS, and MDA combine to build an interoperability environment that extends business automation beyond the firewall to encompass the enterprise's customers and suppliers.

The program also includes educational tutorials on relevant technologies as well as informative panel discussions and two evening receptions. In addition, the final two days feature dynamic technology exhibits and demonstrations.

Highlights of the program include:

KEYNOTE PRESENTATIONS:

- Thomas Koulopoulos, CEO, Delphi Group
- Luba Cherbakov, IBM Distinguished Engineer

TUTORIALS:

- Introduction to Web Services (WS) and Service Oriented Architecture (SOA)
- Modeling Business Process for SOA Applications
- Introduction to Unified Modeling Language™ (UML®) and Model Driven Architecture (MDA)
- MDA, WS and SOA Architecture in the Enterprise

SESSIONS:

- Extending SOA Throughout and Beyond the Enterprise with WS & MDA
- Best Practice in Design and Implementation of WS and SOA
- Tool Chains & Middleware for MDA, WS and SOA Apps
- Security, Scalability, Performance and other Challenges
- Standards in WS, SOA and MDA
- Semantic Aspects and Ontologies
- Case Studies

PANELS:

- Integrators and Tool Implementers
- Case Studies: Lessons Learned

HOSTED BY:

OMG® OBJECT MANAGEMENT GROUP

CO-SPONSORED BY:

M1 Global

FOR MORE INFORMATION OR TO REGISTER:

CALL: +1-781-444-0404

VISIT: www.omg.org/soa-mda-ws

EMAIL: loughry@omg.org

Joining Enterprises with Web 2.0

Big push in 2006

The notion of building bridges to service providers and managing the interaction will become more commonplace in 2006 as we learn to accept that many services we leverage within an enterprise are services we may not host. The technology exists today. We need to define and refine our approaches now, including architectures, enabling technology, and use of standards. Most enterprises are way behind.

WRITTEN BY
DAVID LINTHICUM

We are moving toward a day when most of our enterprise applications may be delivered as services, and thus provide a more economical way to approach information technology management with businesses going forward. This is also the great equalizer since businesses, both large and small, will have access to the same number and quality of services, much as they do with Web sites today. Shared services will create many opportunities, including better agility and the ability to operate a business with fewer IT resources. In essence, we're moving to Web 2.0 where service delivery over the Internet will be added to information delivery as the key strategic value of the Web to businesses, as well as extending the Web as a true platform.

All you have to do is to look around you. With the advent of SaaS (Software as a Service), guys likes Salesforce.com and NetSuite are cleaning up with soaring subscriptions. Moreover, the Web is getting the right interface with rich client technologies, such as AJAX, emerging to provide a much better dynamic user experience. Let's face it; the Web has grown from a simple information delivery platform to a grouping of many valuable exposed services with rich dynamic user interfaces. It's really the global SOA, and those who learn to leverage it now will be well ahead of those who ignore the trend.

> "Shared services will create many opportunities, including better agility and the ability to operate a business with fewer IT resources"

However, in order to make this a reality, we must learn to how to bridge the gaps between our enterprise systems and SOAs and Web service providers that exist across the Internet. Special consideration must be given to connectivity, interoperability, security, and shared processes. Problems are easily solvable with the right technology and approaches, but I would say that most out there who are looking at this new opportunity don't have a clue as to how to make the new and old work and play well together.

The journey has just begun down this road, as we're looking to join our existing enterprise system with Web 2.0, and figure out how all of this works and plays well with the notion of an SOA moving toward a global SOA. Clearly, there are a few technical issues that you must address, such as:

- Semantic and metadata management, or, the management of the different information representations among the external services and internal systems
- Transformation and routing, or, accounting for those data differences during run time
- Governance across all systems, meaning not giving up the notion of security when extending your SOA to the global SOA
- Discovery and service management, meaning how to find and leverage services inside or outside of your enterprise, and how to keep track of those services through their maturation
- Information consumption, processing, and delivery, or, how to effectively move information to and from all interested systems
- Connectivity and adapter management, or, how to externalize and internalize information and services from very old and proprietary systems
- Process orchestration and service, and process abstraction, or, the ability to abstract the services and information flows into bound processes, thus creating a solution

At the end of the day, external Web services should function as any other enterprise application, both housing and sharing critical business information as well as services. In other words, there should be no clear line between the existing enterprise applications and the remote

> "Many small businesses might find that the majority of enterprise processing occurs by leveraging outside services – services they don't own and haven't created"

services. Think about it. You have access to thousands of services with a single, on-demand application provider, as well as information, schemas, etc., and the same patterns found in other on-demand application providers as well. Moreover, you subscribe to these services rather than purchase them, and they exist inside of your enterprise as if they are native.

It doesn't take a rocket scientist to figure out that the creation of an SOA on top of these applications, including process/orchestration layers, directory services layers, identity management, monitoring, semantic management, etc., would add a tremendous amount of value, considering the use of those applications and abstraction into real business solutions. Indeed, I find that many SOAs for many businesses actually exist outside of their firewalls, making their on-demand applications work well together. This trend is only accelerating as Web 2.0 becomes more valuable for enterprises.

What's cool about this is that businesses will have to change to remain competitive. As others learn to embrace Web 2.0 within their enterprise, as with the Web of 10 years ago, others in their community will have to do so just to keep up. There are many examples of this today, albeit it's still early in the cycle. Indeed, many small businesses might find that the majority of enterprise processing occurs by leveraging outside services – services they don't own and haven't created. Is that scary, or exciting? I think it's exciting!

About the Author

David S. Linthicum (www.davidlinthicum.com) is the author of three books on application integration and SOA, a frequent speaker at industry conferences, and the host of the "Service-Oriented Architecture Expert Podcast" (www.soaexpertpodcast.com).

■ ■ ■ linthicum@att.net

ENTERPRISE DATA INTEGRATION

A critical piece of a service-oriented architecture

A large company found itself handicapped by an ornery snarl of siloed applications that compromised its agility, performance, and profitability. Its IT department was constantly behind schedule and over budget in hand coding point-to-point connectivity among supply chain, financials, CRM, and other packaged and custom-built legacy applications.

WRITTEN BY

IVAN CHONG

ASHUTOSH KULKARNI

The solution: integrating critical business processes and applications by adopting a service-oriented architecture (SOA). Internal IT personnel and consultants engineered a loosely coupled infrastructure, with reusable services based on XML and standard Web services protocols such as SOAP and WSDL. Once the system went live, the CFO ran a routine query through his dashboard. The answer came back:

You forgot the data.

It's a playful fiction, of course, but it illustrates the perils of an SOA that focuses only on the business process interactions and application interfaces, and neglects the devilish details of data-level incompatibility among the disparate IT systems participating in those processes, including varying formats, semantics, and hierarchies.

Our hypothetical company based its SOA on a Web services–based enterprise application integration (EAI) engine. The technology worked flawlessly in enabling high-level application integration and orchestrating business processes – but it was not designed to deal with the complexities of heterogeneous, inconsistent, dirty data that lies fragmented across the enterprise.

The result: costly and time-consuming hand coding to resolve these data inconsistencies in the SOA implementation, thus violating the very promise of reusability and interoperability that is driving the movement towards SOA. The missing ingredient in this company's SOA was a data services layer built upon an enterprise data integration platform.

The SOA Opportunity

The buzz around SOA has been fast and furious. It's no wonder – organizations recognize an opportunity to slash the cost of application and middleware development and accelerate time to market by "loosely coupling" siloed applications using open standards such as Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), and Web Services Description Language (WSDL).

The widespread adoption of these standards by IT organizations and vendors alike paves the way to expose applications as component-based services for delivery over the Web. By abstracting the underlying business logic, SOA enables services to be wrapped, reused, and orchestrated to give both IT and business far greater responsiveness, flexibility, and speed of execution.

Many early SOA-based implementations have been built on EAI, and J2EE- and .NET-based middleware, including message brokers, application servers, and enterprise service buses. Increasingly however, data integration has become a primary objective. Some 76 percent of AMR Research respondents using or planning to use an SOA named process or data integration as the leading initiative, according to the August 2005 AMR Research report, "Service-Oriented Architecture: Survey Findings on Deployment and Plans for the Future." The findings reflect a growing awareness that a data integration platform can – and should – enrich an SOA with sophisticated data services beyond the scope of application integration–centric technologies.

In other words, to realize the full potential of SOA, including loose coupling and reusability, it's critical that the client application be able to access business-relevant data wherever it resides, in whatever form it is required, and in a consistent and accurate manner.

> "By abstracting the underlying business logic, SOA enables services to be wrapped, reused, and orchestrated to give both IT and business far greater responsiveness, flexibility, and speed of execution"

Ready for Prime Time: Service-Oriented Data Integration

Data integration technologies are ready to help SOA become a transformative force for IT. Over the past several years, data integration technology has been enhanced with built-in support for XML transformations, Web services protocols, JDBC connectivity, and Java Message Service (JMS) connectivity. Advanced data integration platforms also feature metadata capabilities driving the core of their development and run-time infrastructure. This metadata provides an abstraction of the business logic from the technical implementation, and enables them to deliver advanced data integration functionality over a data services layer to the myriad components in the SOA.

For too many years, data integration initiatives, undertaken without the foundation of a data services layer, have resulted in a further proliferation of the siloed systems that they were meant to integrate. For instance, a retailer might have deployed an extraction, transformation, and loading (ETL) tool to synchronize point-of-sale data from retail outlets into an SAP financials application. A second instance of the tool might serve to move SAP financials information into a DB2 data warehouse for analysis. A third instance might work on the front end of the value chain to feed

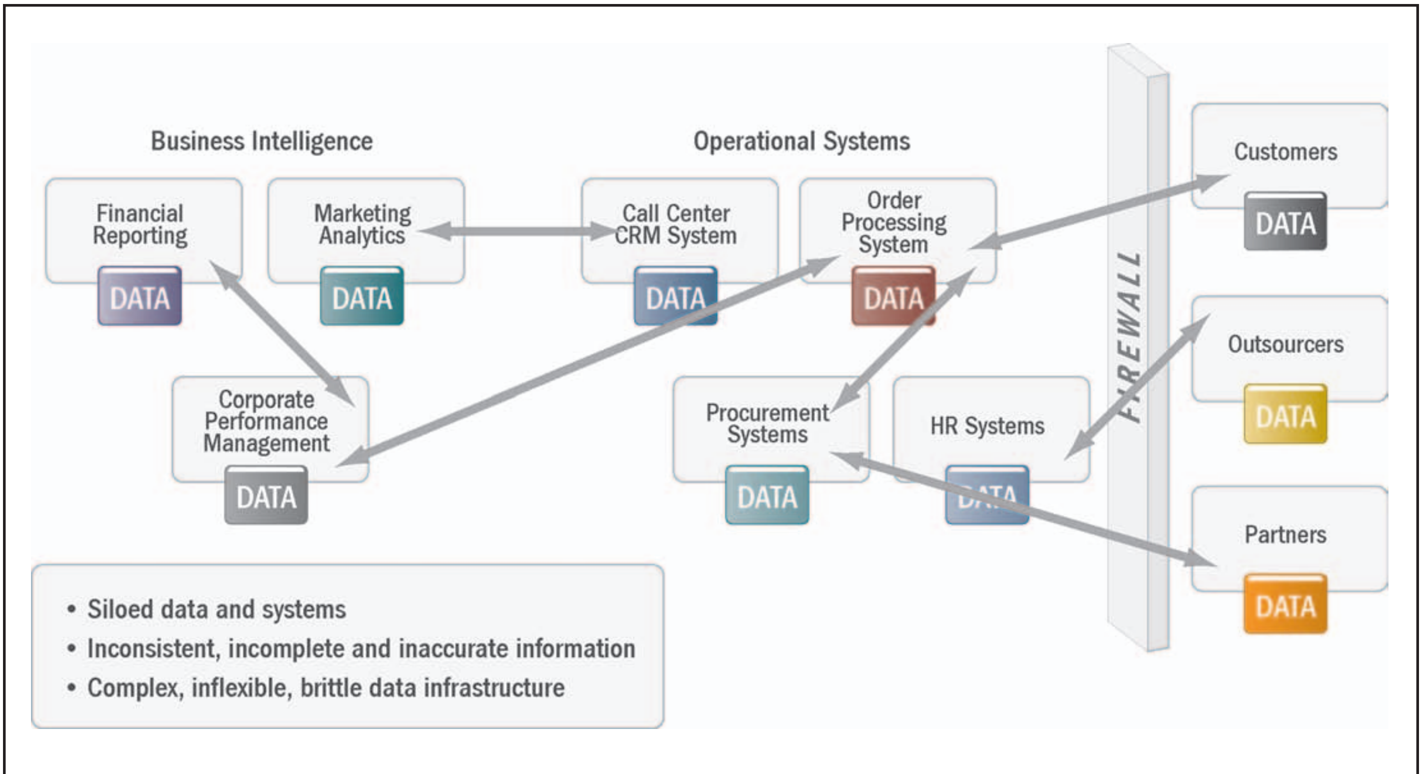


FIGURE 1 **Data is fragmented across the enterprise, resulting in a complex, brittle data infrastructure**

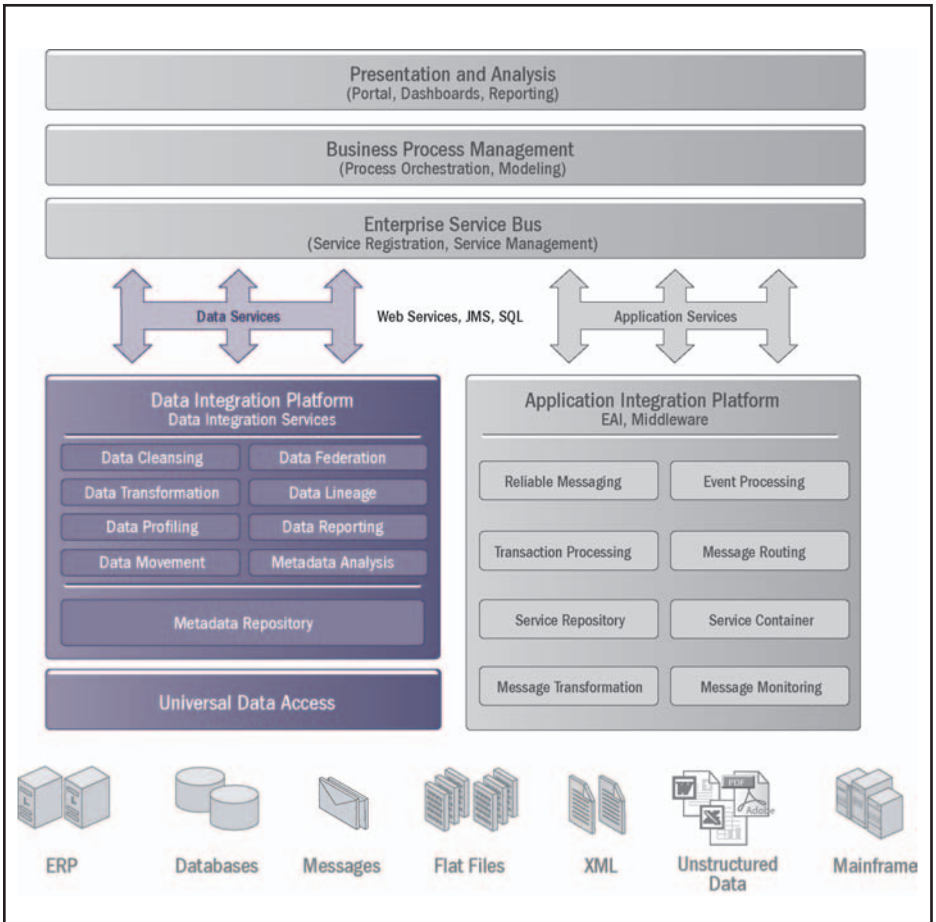


FIGURE 2 **With a service-oriented enterprise data integration architecture, all back-end IT assets are accessed through data and applications services through standard Web services protocols**

product procurement data to an operational data store.

Therefore while the retailer will have achieved data integration among targeted applications, it's still several steps removed from realizing a fluid, end-to-end data ecosystem. SOA removes these barriers of siloed development.

In a modular SOA, a data integration platform serves as another component-based service. Its functionality can be packaged and reused across multiple projects to reduce development and deployment costs. It can help an organization leverage data assets that are currently locked in mainframe, packaged, and homegrown systems through open standards. It can eliminate the need to hand-code data integration connectivity, and enables businesses to realize rapid time to value.

That's what SOA offers data integration technology. Now let's look at the flip side – what data integration does for SOA.

Data Components and Services in an SOA

The most advanced SOA deployments will take advantage of both EAI and data integration technologies. SOA provides an ideal framework for these two technologies to complement one another, with EAI managing transactions and processes among applications, and the data integration platform performing the atomic-level data processing that is generally beyond the scope of EAI systems.

In fact, a common use case is where a company deploys an EAI bus and a data integration platform in an SOA to support master data management initiatives, such as customer data integration. The EAI bus drives business processes and checks customer records in the master data repository. The data integration platform creates the master data repository and populates back-end ERP systems with updated customer information transformed to the appropriate format and semantic definition.

In strategizing options and objectives for an SOA, organizations should assess and understand the functional distinctions between the two technology sets. Let's take a look at three functional components that are the exclusive province of data integration technology – universal data access, a metadata repository and services, and a data integration engine.

Universal Data Access: Scope of Data

Data integration extends the reach of the SOA and its constituent applications into virtually any data source. Prebuilt connectivity and visual mapping environments provide IT architects and developers with a mechanism to tap into information from a variety of sources, including packaged and homegrown applications such as SAP, mainframe and midrange systems such as IMS and VSAM, relational databases such as Oracle and Sybase, and unstructured and semistructured data.

Organizations can use data integration to reach into multiple systems to fetch data, cleanse and transform it into the appropriate formats and semantic definitions, and propagate it across multiple distributed systems. Its service may be invoked by, for instance, an online customer order application to trigger event-driven, read/write data updates across financials, manufacturing, and distribution in near real time.

Metadata Repository and Services: Meaning of Data

A metadata repository provides the SOA with an underlying foundation to understand the lineage of data, the ripple effects of changes, and data-related deficiencies in the architecture. The repository provides a data interaction framework to store and manage data models, transformations, workflows, and dependencies – metadata describes the data logic and its meaning. Through metadata services, data integration technology provides a means to reconcile data semantics across disparate systems; improve reporting, auditing, and data governance; and enable reuse to streamline development and accelerate deployment.

Metadata is also key in equipping organizations with an auditable record of data lineage

> "The technology worked flawlessly in enabling high-level application integration and orchestrating business processes – but it was not designed to deal with the complexities of heterogeneous, inconsistent, dirty data that lies fragmented across the enterprise"

covering all data resources, thus providing an important tool for meeting the compliance requirements of Sarbanes-Oxley and other regulations.

Data Integration Engine: Value of Data

At the core of data integration is an engine that provides organizations with a host of options for moving, integrating, and delivering data among various consumers in an SOA. Its flexibility is important for letting IT professionals architect a system optimized for "right time" data delivery, including high-volume batch data movement, near real-time capture and movement, and changed data capture – only data updated since the last service invocation.

Data integration also offers functionality to help "future-proof" an SOA against rising data volumes, and to meet the requirements for reduced data latency as well as the demands for toughened security and privacy. For example, data integration supports partitioning to optimize parallel processing on multi-CPU hardware, deployment on multi-node server grids for distributed workflow execution and fault tolerance, failover, and fortified security through authentication, authorization, and encryption.

At Your Service: Data Dividends in an SOA

Those are the core data integration components in an SOA. Now let's examine the services and benefits that they deliver for an SOA – data profiling, cleansing, transformation, movement, and auditing:

- ***Data Profiling:*** Data profiling is the process of assessing and understanding the content, quality, and structure of enterprise data. It is an essential step in reconciling semantic differences in common business vocabularies such as customer, address, and product that vary among applications, and which, if unaddressed, result in contradictory information across the enterprise.
- ***Data Cleansing:*** Once data is profiled, a data integration platform can execute data cleansing functions to ensure the validity and consistency of information. It standardizes name, address, and other values, and resolves missing data fields, parses data elements, and corrects poorly formatted or conflicting data.
- ***Data Transformation:*** Data transformation services enable data to be transformed from one form to another to allow reconciliation between data elements residing in different information sources. The transformation services leverage prebuilt and customized mappings that take into account complex data hierarchies and relationships.
- ***Data Movement:*** Data integration offers flexible mechanisms for "right-time" data delivery in an SOA, including high-volume bulk data movement, near real-time capabilities, data federation, and changed data capture that handles only information that has been updated to accelerate load times and minimize operational impact.
- ***Data Auditing:*** Data integration provides in-depth lineage of data – when it was changed, how, by whom, and across which applications – to enable the auditing, reporting, and analysis that is essential to meeting the demands of legislated regulations and internal/external auditors.

The data services provided by the data integration platform can be accessed by other components in the SOA via Web services protocols such as SOAP, messaging systems such as MQSeries or JMS, and programmatic approaches such as JDBC and ODBC.

Where Do We Go from Here?

SOA may still be in its early phases, but the time is right to take a hard look at data-related business objectives and IT resources in your service-oriented architecture blueprint. One key to success is an iterative approach that focuses first on targeted projects with quantifiable business value that are relatively easy to implement. SOA, after all, is a matter of architecture, and no organization is going to rearchitect its systems overnight.

By implementing a data integration platform at the ground level, you can ready your IT systems to fully leverage that most valuable of business assets – data – without reengineering, hand coding, and having to worry about data quality problems down the line. Plus, you'll never worry about receiving another response that says, "You forgot the data."

About the Author

Ivan Chong, vice president of product strategy and marketing at Informatica, has more than 15 years of experience in data integration and database product development. During his eight years at Informatica he has held a variety of senior management positions, including product management roles, in which he transformed the company's product development process.

■ ■ ■ ichong@informatica.com

Ashutosh Kulkarni is the principal manager for Informatica's enterprise data integration strategy, focusing on service-oriented architectures and Integration Competency Centers. Prior to joining Informatica, Kulkarni spent eight years at Sun, where he led Sun's strategic marketing initiative around SOA.

■ ■ ■ akulkarni@informatica.com

Java™

Don't Miss the 2006 JavaOne℠ Conference

More than 300 technical sessions and Birds-of-a-Feather sessions will be offered at the 11th Annual JavaOne℠ Conference. See and hear from the Industry leaders and technology experts over four content-rich days. Included are:*

Introduction to AJAX

Ben Galbraith, *Consultant* | Dion Almaer, *Adigio, Inc*

This session provides an introduction to AJAX and an orientation to the state of the AJAXian universe. It demonstrates the basic AJAXian techniques through live coding and demonstrates and deconstructs more-advanced examples of AJAX.

Effective Java™ Reloaded

Joshua Bloch, *Google, Inc.*

It has been five years since *Effective Java*™ was released. The Java platform has evolved, and we've learned more about how to use it to best effect. Therefore, a second edition of *Effective Java*™ is being released to coincide with the 2006 JavaOne conference. This presentation covers new material that has been added to the second edition, material that should be useful to every working Java technology programmer.

Spring Framework Update

Rod Johnson, *Interface21*

Rod Johnson, the father of Spring, brings attendees up to date on some of the many significant new features in the Spring 1.3 and 1.4 releases. He discusses what's new and cool in the Spring world and examines the implications of these new features for best practice in developing applications with the Spring Framework. Johnson shows code examples throughout the presentation, leaving attendees ready to try these features out for themselves.

To see more information on the conference offerings visit java.sun.com/javaone/sf

*Content subject to change.

PLATINUM COSPONSORS

IBM

ORACLE

GOLD COSPONSORS

JUSTSYSTEM

SAP

SILVER COSPONSORS

QUEST SOFTWARE

TERRACOTTA

Register by April 14, 2006 and SAVE $100.

java.sun.com/javaone/sf

Sun microsystems

JavaOne℠ Conference | May 16-19, 2006

JavaOne℠ Pavilion: May 16–18, 2006, Moscone Center, San Francisco, CA

JavaOne

Sun's 2006 Worldwide Java Developer Conference™

Trusting Reusable Business Components in a Service-Oriented Architecture

A visible quality process is key

One of the business benefits organizations strive to achieve by implementing a service-oriented architecture (SOA) or in utilizing Web services is the opportunity to reuse business components. Asset reuse is one of the core drivers of the SOA or Web service ROI calculation. Although leveraging the service concept provides an avenue for application consolidation and reuse, these same efficiencies also introduce a distinct level of business risk.

WRITTEN BY
WAYNE ARIOLA

For example, in an architecture where multiple distributed applications provide a business with a specific operation, should a single application fail, the impact of the application downtime is isolated only to a limited set of users. Conversely, the practices of application consolidation and service reuse concentrate operations into fewer applications or perhaps a single point of failure. As seen in the Figure 1, the impact or risk associated with a service's failure increases as more entities or business processes are dependent on that service.

Therefore, if a business entity consolidates its applications and exposes a specific operation as a service and the service fails, the potential enterprise-wide exposure to the service downtime can cause significantly more risk.

Although this article is focused on managing the risks associated with application consolidation and service reuse, there are many other challenges associated with reuse that will not be addressed. Topics such as service definition, service granularity, and service access also challenge the ROI associated with reuse.

In spite of the technical risks involved, reusable components are core to the SOA vision. If the business cannot trust these components, these components cannot be reused. Therefore, in order to fully reap the benefits of SOA and Web services, it is critical that companies find an efficient and robust solution that mitigates risk, increases trust, and ultimately promotes reuse. The solution is a visible, objective, and quantifiable quality process that provides users with the confidence to reuse these services.

Learning from the Automotive Industry

Other industries have learned how to create visible quality processes in order to instill trust in reused products. The automotive industry is a prime example of this paradigm. Various automakers have spent a significant amount of money branding certification programs with the purpose of instilling trust that their used cars are secure, reliable, and valuable. In fact, there really is no such thing as a used car (or a reused car) in the auto industry. Instead, the auto industry has spent a significant amount of money branding these "reused" assets as "certified pre-owned" cars. A visible and comprehensible quality program has evolved the "used car" into a functional and reliable reusable asset by the consumers who purchase them. The trust and confidence exhibited by used car buyers is a direct result of the visible certification (quality) programs of various auto companies.

Consider, for example, various companies in the automotive industry and their certification programs for pre-owned cars as illustrated in Table 1.

As seen in the table, various automakers utilize visible, objective, and quantifiable metrics to demonstrate that their pre-owned cars (reused assets) are indeed secure, reliable, and valuable. If any of these quality metrics were absent, confidence and trust in the used car would also be absent. In the table for example, the point inspection score provides a necessary quality check for the car, the aspect of special financing provides a monetary reward for purchasing the car, and the warranty provides additional benefits for the car should the quality degrade over time. Each of these visible, objective, and quantifiable metrics increases consumer trust and ultimately promotes reuse.

The key to such certification programs is the visibility with which they are presented, lead by the "certification" seal. Therefore, promoting and instilling trust in reused business components in an SOA model must be approached in the same manner as the certification of pre-owned cars in the auto industry: a visible quality process must be in place in order to trust the reusable assets.

"Asset reuse is one of the core drivers of the SOA or Web service ROI calculation"

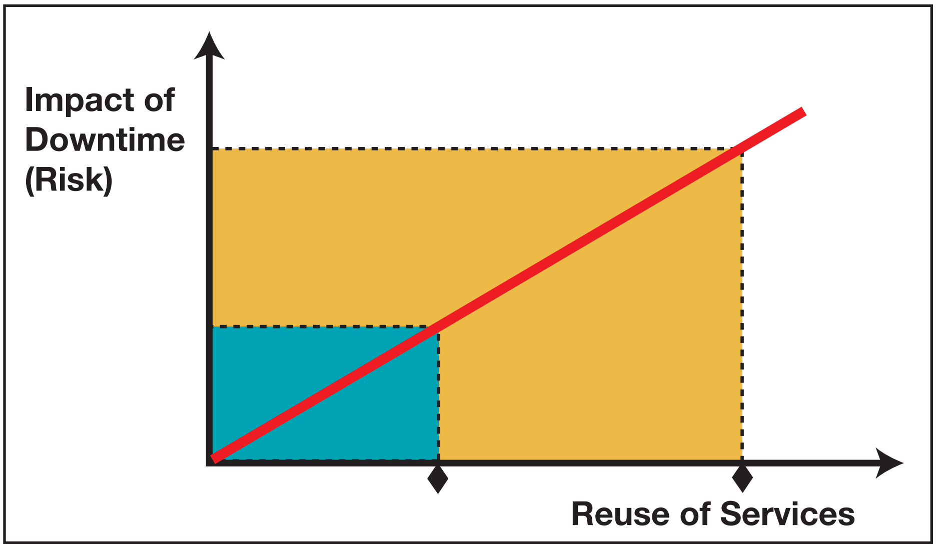


FIGURE 1 **Correlation of service-failure risk and resue of business processes**

Manufacturer	Point Inspection	Special Financing	Certified Warranty Details
Chrysler	125	Yes	8 year/80,000 mile Powertrain Limited Warranty, measured from original vehicle in-ervice daat
Ford GMC	115 110+	Yes Yes	6 year/76,000 miles from the In-Service date 3 months/3,000 miles from the Purchase date
Lexus	161	Yes	3 years from the Purchase dat/100,000 miles from the In-Service date
Mercedes-Benz	130+	Yes	12 months from Purchase date, 100,000 miles from the In-Service date
Toyota	160	Yes	7 year/100,000 miles Limited Power Train Warranty from date when first sold as new

TABLE 1 **Visible quality metrics can promote trust and confidence in reusable assets**

Creating a Visible Quality Process for SOA

Similar to the certification programs of used cars, inspection points of business components must be established, executed, and measured before designing or exploring an SOA project. Such a process must be clearly defined, objective, reliable, and repeatable in order to promote trust in the business components that are to be reused either internally or externally.

When exposing business information and operations either within the enterprise internally, or by sharing data with partners externally, the business must ensure that each part of its system is reliable, and that all of these parts interact flawlessly and securely. Visible quality metrics are necessary to promote internal adoption and reuse, as well as to promote business continuity by providing enough metrics to assure and promote trust with external partners.

Internally, the business must enable itself using these business components. The development organization and business groups must be aware that these business assets are available and valuable. Ultimately, the organization must have trust that the business components they choose to leverage are secure, reliable, and compliant.

Externally, the business must enable its partners to leverage these business assets. With external reuse, the issue of risk compounds because of the mission-critical aspect of these business transactions. A down time of an hour cannot only cost substantial losses in revenue but, more important, can foster the perceived lack of quality and reliability in the company in general. Therefore, trust is especially important when reusing external business components.

At the very minimum, companies should utilize interoperability, unit, scenario, regression, performance, and security penetration testing to provide the necessary inspection points for the business components. As with the certification processes that the automotive industry utilizes, a visible and capable quality process is critical for the success of SOA as well. The lack of visibility and an objectively measurable process can lead to the organization and its partners simply not reusing the business components. This lack of trust, which is reasonable when dealing with mission-critical business processes, will deteriorate the efficiencies that the SOA model promises.

Conclusion

Companies have invested enormous sums of money in building and stabilizing the legacy systems that support vital business components. SOAs offer these companies an efficient and robust way to decrease integration expense, increase business agility, and consolidate applications. SOA offers ubiquitous access to services and the capability to share data with business partners, customers, and information systems with unpa ralleled efficiencies. However, reusing business components via SOA inherently increases risk. Not only is the concept of reuse in jeopardy if internal entities do not trust the publisher of the service, but also external entities can reject leveraging the service if they feel that their business process could be at risk by using the service.

The migration to a service-oriented architecture will not be successful without an implementation plan for a robust and objective quality process. Just as in the evolution from "used cars" to "pre-owned cars," a visible quality process will promote trust, promote adoption, increase application quality, and ultimately reduce business risk.

About the Author

Wayne Ariola, vice president of Corporate Development at Parasoft, oversees the company's Corporate Development team as well as the Enterprise Test team.

wayne.ariola@parasoft.com

Successfully Planning for SOA

Part 2 – Building your SOA roadmap

■ In this second article about service-oriented architecture (SOA), I offer a concrete plan, along with tips and insights, to help you build an effective SOA roadmap, and to help ensure the success of your SOA initiative.

WRITTEN BY
STEPHEN BENNETT

Any great journey starts with a goal or destination, and your organization's decision to implement SOA is no different. However not unlike the pioneers who set off west in their wagons, you may start with only a vague idea of what awaits you, or how you might get to your destination. To be successful, you must assess your strengths and weaknesses, establish clear direction, choose a route, and then consistently reassess that route as you follow it. You must, to put it simply, create your own unique map for your journey.

What Is an SOA Roadmap and Why Do You Need One?

SOA is an IT strategy that organizes the discrete functions contained in enterprise applications into interoperable, standards-based services that can be combined and reused quickly to meet business needs. The benefits of SOA will only be realized if the balance between long-term goals and the shorter-term needs of the business are preserved. This balance can be maintained by instituting a set of organizational, financial, operational, design, and delivery practices from the outset of your SOA initiative. However it is important that these culture-changing disciplines are deployed in an incremental and iterative fashion, rather then a "big bang" approach, which allows for an organizational learning curve. In essence, an SOA roadmap is an iterative and incremental way to capture (and recapture) your organization's unique plan as you progress.

Your SOA Roadmap should contain three critical characteristics:

- ***Maturity:*** Treat your SOA Roadmap as a "living document" that continually captures experiences and lessons learned. As your SOA roadmap matures, your SOA initiative reaches higher levels of sophistication, in a controlled manner. The creation of an SOA roadmap begins with an assessment of your organization's current capabilities and disciplines that are applicable to SOA. This process can be initiated by using BEA's Online Self-Assessment Tool (www.bea.com/framework.jsp?CNT=index.htm&FP=/content/solutions/soa/).

"The benefits of SOA will only be realized if the balance between long-term goals and the shorter-term needs of the business are preserved"

FIGURE 1 **The BEA domain model**

- ***Scope:*** A complete SOA roadmap should encompass all six domains (see Figure 1). These domains, while distinct, are interrelated and interdependent. Executing on each domain is fundamental to the success of an enterprise-wide SOA initiative. The SOA roadmap should clearly delineate the boundaries of your SOA initiative and establish a transparent and flexible timeline for achieving SOA goals. These goals should be broken down into manageable phases, which can then be realized in an iterative and incremental manner.
- ***Quality:*** By applying a "Learn and Adapt" process at each milestone, and by being both iterative and incremental, your roadmap will remain relevant throughout the SOA initiative. To ensure your SOA roadmap's quality, communicate and validate it with all stakeholders, soliciting feedback and buy-in from all quarters.

How to Build an SOA Roadmap

There are four phases to developing your SOA roadmap: SOA planning, SOA maturity assessment, SOA future vision, and SOA roadmap definition.

SOA Planning

During this phase, your SOA initiative is organized and defined. Stakeholders are brought into the process through communications and briefings, and mutually agreed upon priorities and parameters are set. Because this phase involves employees across your organization, clear and ample communication is critical. During this phase you will:

- Define the scope of SOA
- Establish boundaries and alignments with other IT initiatives
- Appropriately showcase the business justification for SOA
- Show alignment of existing and future business initiatives

SOA Maturity Assessment

During the SOA maturity assessment phase, you will establish a metric for where you are today. Here you will define what services and capabilities you currently have that can serve as a starting point for SOA, as well as identify projects that might serve as foundation projects. Through a series of interviews and questionnaires, your teams should examine the various domains – analyzing, base-lining, and validating the "as-is" current situation for each. Use of BEA's domain model allows you to structure your examination of the following:

- ***Business Strategy and Process:*** A top-down view of business strategies and processes
- ***Architecture:*** A review of current architectures, policies, and standards
- ***Cost and Benefits:*** Overview of existing cost structures and benefits cases
- ***Building Blocks:*** An analysis of existing services, processes, tools, and technologies
- ***Projects and Applications:*** Review of existing systems and in-flight and planned projects
- ***Organization and Governance:*** Analysis of existing governance structures and policies

SOA Future Vision

In this phase, teams use workshops to determine and define the future desired "should-be" state and ensure cross-organizational buy-in:

- ***Business Strategy and Process:*** Correlation

> "Your SOA roapmap should be treated as a 'living document' that continually captures experiences and lessons learned"

BY NOW THERE ISN'T A SOFTWARE DEVELOPER ON EARTH WHO ISN'T AWARE OF THE COLLECTION OF PROGRAMMING TECHNOLOGIES KNOWN AS AJAX!

REAL-WORLD **AJAX** ONE DAY SEMINAR

www.ajaxseminar.com

March 13, 2006	April 24, 2006
Marriott	DoubleTree Hotel
Marriott Marquis Times Square	
New York, NY	San Jose, CA

For more information
Call 201-802-3022 or
email events@sys-con.com

How, in concrete terms, can you take advantage in your own projects of this newly popular way of delivering online content to users without reloading an entire page?

How soon can you be monetizing AJAX?

This "Real-World AJAX" one-day seminar aims to answer these exact questions...

Led by "The Father of AJAX" himself, the charismatic Jesse James Garrett of Adaptive Path, "Real-World AJAX" has one overriding organizing principle: its aim is to make sure that delegates fortunate enough to secure themselves a place – before all seats are sold out – will leave the seminar with a sufficient grasp of Asynchronous JavaScript and XML to enable them to build their first AJAX application on their own when they get back to their offices.

Jeremy Geelan
Conference Chair, Real-World AJAX
jeremy@sys-con.com

HURRY! LIMITED SEATING THIS SEMINAR WILL SELL-OUT
CALL 201-802-3022 TO REGISTER!

Jesse James Garrett
Father of AJAX

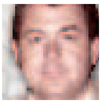
Scott Dietzen
Creator of WebLogic, Ph.D., President and CTO, Zimbra

Bill Scott
AJAX Evangelist of Yahoo!

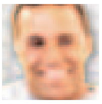
David Heinemeier Hansson
Creator of Ruby on Rails

Satish Dharmaraj
Father of JSP, Co-Founder & CEO, Zimbra

Rob Gonda
Best-Selling AJAX Author, CTO, iChameleon Group



Dion Hinchcliffe
Co-founder & CTO, Sphere of Influence Inc.



Ross Dargahi
Well-known AJAX Evangelist & Co-founder and VP of Engineering, Zimbra

Early Bird (Before January 31, 2006)	$995
Discounted Price (Before February 28, 2006)	$1,195
Seminar Price (After February 28, 2006, and if any seat is available)	$1,295

MEDIA SPONSOR
ajax developer's journal

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

PRODUCED BY SYS-CON EVENTS

of SOA future vision with business strategies and processes

- ***Architecture:*** Architecture guiding principles, requirements, policies, standards, and reference architecture
- ***Cost and Benefits:*** Metrics and measurement requirements
- ***Building Blocks:*** Shared services infrastructure requirements, standardize tools
- ***Projects and Applications:*** SOA mapping to projects and applications
- ***Organization and Governance:*** Governance and compliance structures and policies

> "The SOA roadmap should clearly delineate the boundaries of your SOA initiative and establish a transparent and flexible timeline for achieving SOA goals"

SOA Roadmap Definition

This phase is where the SOA roadmap is initially defined. A complete gap analysis should be performed for your corporation's SOA goals and appropriate timelines, based on the information gathered in the previous three phases. Near-term events will be more detailed, while later events will be more fluid – so that they might incorporate lessons learned as you move forward.

- ***Business Strategy and Process:*** Opportunity alignment by business value
- ***Architecture:*** Near-, medium-, and long-term reference architecture roadmap
- ***Cost and Benefits:*** Roadmap of future metrics, cost structures, and benefits cases
- ***Building Blocks:*** Prioritization of shared services strategy and standardized processes
- ***Projects and Applications:*** Project and application impact
- ***Organization and Governance:*** Proposed governance structures and policies

Your SOA roadmap should be treated as a "living document" that continually captures experiences and lessons learned. As your roadmap matures, your SOA initiative will reach higher levels of sophistication in a controlled manner (see Figure 2).

FIGURE 2 **The SOA "learn and adapt" roadmap**

Conclusion

The goal of this article is to provide you with a framework for creating your own SOA roadmap, and an explanation of why that roadmap is so important for your SOA initiative. Your roadmap is your guide for what to develop, when to develop, and when to deploy what you've developed, and should be your single most powerful tool for a smooth deployment of SOA. For more information on BEA's SOA solutions, please visit www.bea.com/soa.

References

- *BEA Domain Model Whitepaper (PDF):* http://contact2.bea.com/bea/www/soarc/login.jsp?PGM=1&PC=10SOSO4
- *Successfully Planning for SOA:* http://contact2.bea.com/bea/www/soarc/login.jsp?PGM=1&PC=10SOSO4

About the Author

Stephen Bennett is a consulting technical manager at BEA Systems. He has over 20 years of industry experience covering IT strategy, enterprise architecture, and agile software engineering. Stephen is currently focusing on SOA thought-leadership and in developing BEA's SOA service offerings, specifically around SOA governance.

sbennett@bea.com

ENGAGE AND EXPLORE...

The Technologies, Solutions and Applications that are Driving Today's Initiatives and Strategies...

JUNE 5-6, 2006 Roosevelt Hotel / New York, NY

RESISTER TODAY!

SOA 10th International WebServices Edge conference+expo

The Sixth Annual SOA Web Services Edge 2006 East - International Web Services Conference & Expo, to be held June 2006, announces that its Call for Papers is now open. Topics include all aspects of Web services and Service-Oriented Architecture

Suggested topics...

- > *Transitioning Successfully to SOA*
- > *Federated Web services*
- > *ebXML*
- > *Orchestration*
- > *Discovery*
- > *The Business Case for SOA*
- > *Interop & Standards*
- > *Web Services Management*
- > *Messaging Buses and SOA*
- > *Enterprise Service Buses*
- > *SOBAs (Service-Oriented Business Apps)*
- > *Delivering ROI with SOA*
- > *Java Web Services*
- > *XML Web Services*
- > *Security*
- > *Professional Open Source*
- > *Systems Integration*
- > *Sarbanes-Oxley*
- > *Grid Computing*
- > *Business Process Management*
- > *Web Services Choreography*

RESISTER TODAY!

2006 ENTERPRISE> OPENSOURCE CONFERENCE+EXPO

The first annual Enterprise Open Source Conference & Expo announces that its Call for Papers is now open. Topics include all aspects of Open Source technology. The Enterprise Open Source Conference & Expo is a software development and management conference addressing the emerging technologies, tools and strategies surrounding the development of open source software. We invite you to submit a proposal to present in the following topics. Case studies, tools, best practices, development, security, deployment, performance, challenges, application management, strategies and integration.

Suggested topics...

- > *Open Source Licenses*
- > *Open Source & E-Mail*
- > *Databases*
- > *ROI Case Studies*
- > *Open Source ERP & CRM*
- > *Open-Source SIP*
- > *Testing*
- > *LAMP Technologies*
- > *Open Source on the Desktop*
- > *Open Source & Sarbanes-Oxley*
- > *IP Management*

Submit Your Topic Today! events.sys-con.com

Sponsored by

WebServices Journal, XML Journal, .NET Developer's Journal, eclipse developer's journal, WebSphere Journal, Information Storage+Security Journal, wldj, JDJ, LinuxWorld Magazine, MX developer's journal, asp.netPRO, SD Times, CoDe, Software Test & Performance

*Call for Papers email: events@sys-con.com

Attention Exhibitors:

An Exhibit-Forum will display leading Web services and OpenSource products, services, and solutions

For Exhibit and Sponsorship Information ▸ Call 201 802-3023

Produced by SYS-CON EVENTS

Inducing Behavioral Adaptations Within Web Services

Gaining control over dynamic changes in Web services

Web services have evolved from plain application-integration enablers to value-added stand-alone functionality providers such as getting a quote to a comprehensive business function like processing orders. Enterprises are exploiting this new revenue model by offering such business functions as services on usage basis. Information sharing instances such as Amazon's e-commerce services have helped in the creation of new powerful solutions using business functions as services.

WRITTEN BY


NAVEEN KULKARNI


GEO PHILIPS KURAVAKAL


LIPIKA SAHOO


SUNNY SAXENA


MANIVANNAN GOPALAN

Why Do We Need Adaptation in Web Services?

In the vision of e-business, it becomes important to address the dynamic nature of business environments. As third parties access more and more business functions, the richness of Web services offered should be adequately addressed. With Web services having to participate and cooperate in a networked system, they may need to frequently join or leave a process, or at times may need to change their preferences by tailoring or adding functionalities. Such unanticipated changes as demanded by businesses and the associated complexity necessitate an adaptable nature to web services where specific features can be introduced or deactivated in a planned manner.

The adaptation of Web services with regards to constant change in the business has to deal with the behavior of the service. Some of the existing approaches of adaptation primarily concentrate on transformation of the content and much less effort has been made towards behavioral adaptation. Much as object-oriented programming introduced encapsulation and inheritance, the behavioral adaptation should also introduce features in a modular way. However in contrast to the object-oriented way, certain significant issues need to be addressed:

- *Context-aware provisioning,* which is necessary because Web services will be involved in widely distributed and heterogeneous solutions
- *Diversified personalization,* for improving quality of service and delivering user-centric functionality
- *Nondisruptive feature insertion/release,* to facilitate the high availability requirement

A scenario of a loan agent providing a best possible loan offer is constructed to explain the aforementioned issues. This example will be used later on in the article to illustrate certain adaptation scenarios. A process as shown in Figure 1 is initiated to select the best loan offered by any participating loan service to meet specific requirements as stated by the consumer. In the real world, these loan services can be considered to be the endpoints of various financial institutions. The process would also look up the consumer's credit score from a trusted credit rating service.

Aren't Business Rules Sufficient?

Even though business rules also define or constrain the behavior of services, they are primarily intended to enforce certain explicit motivations based on contextual data. On the other hand, Web services in e-business scenarios would have to deal with proactive and reactive adaptations. Such adaptations have to be non-enforcing and yet facilitate the requestor's ability to choose the behavior. To understand this better consider a simple personal loan offering service. The bank may decide to run a promotion for a specific season by virtue of which a business rule could be added to the service so that loan requests with credit scores in the range of 60 to 75 on a 100-point scale would reduce the interest rate by 0.25 percent. However converting the offered personal loan to a line of credit having variable interest rates would involve changes in the service logic itself. This wouldn't be a feasible option because it has to be made available only for a specific time period. Moreover, because requestors have

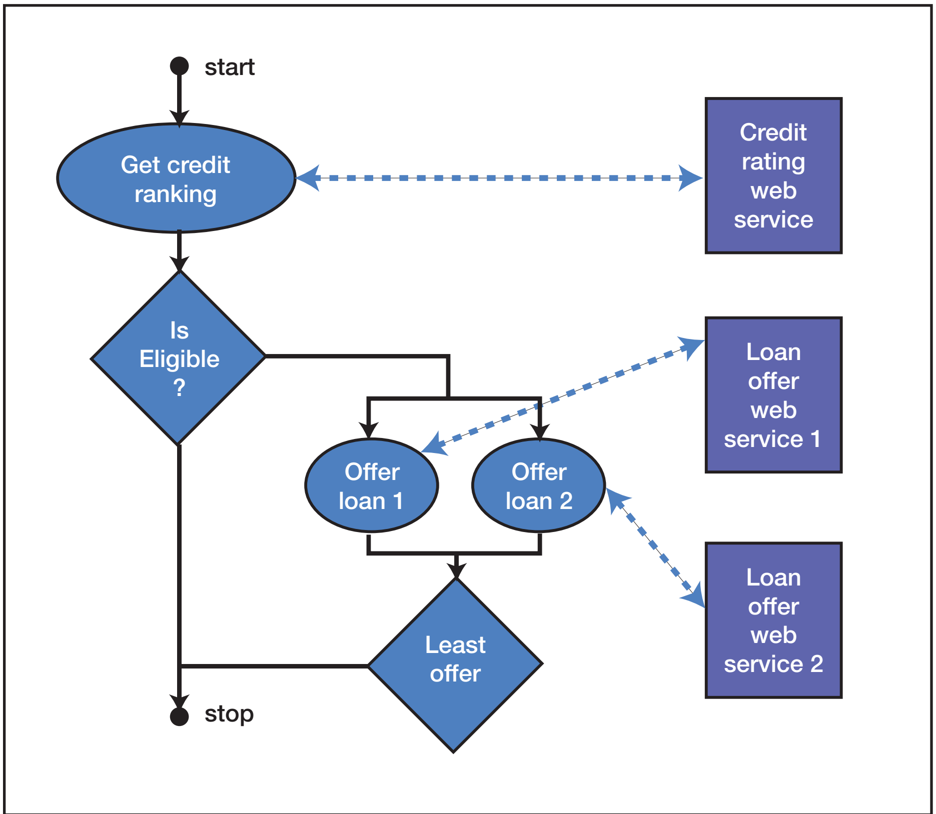


FIGURE 1 **Loan offer process by an agent**

different preferences some might not opt for this conversion while others might. Using business rules, such adaptations would get even more complex if the preferences are negotiated before the request is made to the service.

Adaptations May Not Be Just for Functional Features

The behavior of Web services as in the case of any computational system would be made up of certain functional and nonfunctional features. The sole purpose of the nonfunctional features would be to enable services to function for various execution conditions through changes in the behavior of the service without altering the core service logic. A distinguishing factor of such adaptations is that often the service itself might be unaware of the existence of nonfunctional features. To put it another way: the nonfunctional features are not configured at design time but are categorized under operational constraints.

The operational constraints would essentially relate to the communication characteristics between the interacting services. There are various WS-* standards that are trying to address the nonfunctional requirements of Web services. There are also some competing standards for similar functionality. For example, to tackle reliable, guaranteed message delivery, a group consisting of Sun, Fujistu, Oracle, etc. came out with WS-Reliability and a second group consisting of IBM, BEA Systems, Microsoft, and TIBCO came up with the WS-ReliableMessaging specification. Though both provide mechanisms to ensure basic delivery assurance, the difference between them is in how they incorporate other standards such as WS-Security, WS-Addressing, and WS-Policy. On the contrary, there are some specific widely accepted and employed standards for a given problem area that have overlapping features. For instance, in the case of security, there are standards such as Security Assertion Markup Language (SAML) as well as WS-Security that are both widely used to provide security cover to Web services. Because this is the case with the ever-growing stack of WS-* standards, participants within a composition have many options to choose from. Even with such disarray within the stack, some of the reasons that participants differ in their choices could be the pace of adoption of the standard, or the preference for a specific platform/tool choice made during the service development, etc.

Essential Considerations for Adaptation

The behavioral adaptation is to be considered in the context of interaction between the service provider and the requestor. To appropriately customize Web services, the contextual information has to be processed so that the intended interaction is achieved. In order to do so the following considerations are essential for leading to behavioral adaptations within Web services.

Variation Points or Hotspots

The unanticipated introduction of new

> “Even though business rules also define or constrain the behavior of services, they are primarily intended to enforce certain explicit motivations based on contextual data”

features has to coexist with the base functionality offered by the Web service. It is also possible that the adaptations would have to intervene between the basic flows. Web services planning for such adaptations will have to identify the variation points or hotspots where new features can be induced. The identification of such hotspots is very crucial and decides the degree of flexibility that a Web service can offer. Another important aspect of hotspots is their role in the weaving of the adaptations. This facilitates the ability of the Web service to truly cater to and take part in a wide variety of solutions.

Proactive and Reactive Nature

Proactivity in the adaptation could be accomplished by deciding on the service behavior even before the change or before the invocation of the service. On the other hand, reactivity is the consequence of the change after the service invocation. For example, the credit reporting service could exhibit a proactive behavior in providing a simple report that contains the credit score and summary or a complete report that details public records, past dues, payment, collections, open accounts, inquires, etc. Reactive behavior may be accomplished by providing the credit score on different scales like 400-900 or 100 points.

Late Binding

The undetermined quality of adaptations necessitates late binding. Without having late binding, the service needs to be as abstract as possible and has to extend itself to add the features indented for adaptation. This may not be feasible and may lead to chaos in managing requirements, which leads to the cluttering of features.

Negotiations

Negotiations provide the property of compatibility and replaceability to the interactions between the requestor and the provider. The constantly changing preferences could be accommodated through negotiations so that both requestor and provider can agree on the best possible terms for information exchange. For example, the way in which security credentials are passed can be negotiated. To ensure his confidentiality, the service requestor can opt for a lengthier encryption key and similarly the service provider can specify various options for the encryption key length from which the requestor can choose.

Noninvasive Adaptations in Web Services

As mentioned earlier, one of the important issues to be solved during adaptation is the insertion/removal of features nondisruptively or noninvasively. In other words, the base service logic should not be aware of any changes. This is also a key issue for maintaining high availability, which is an important quality of a service attribute. Earlier works in this direction have advocated the use of an aspect-oriented paradigm that was essentially used to support many of the cross-cutting concerns. Furthermore, with our experience in working with aspect orientation, we also conciliated with

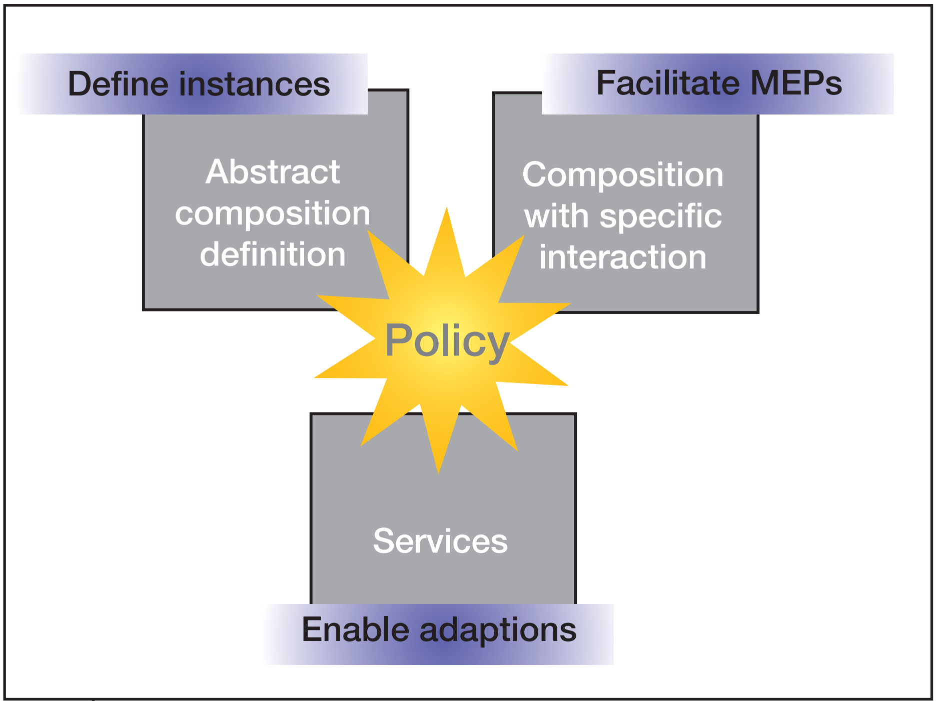


FIGURE 2 **Role of policy**

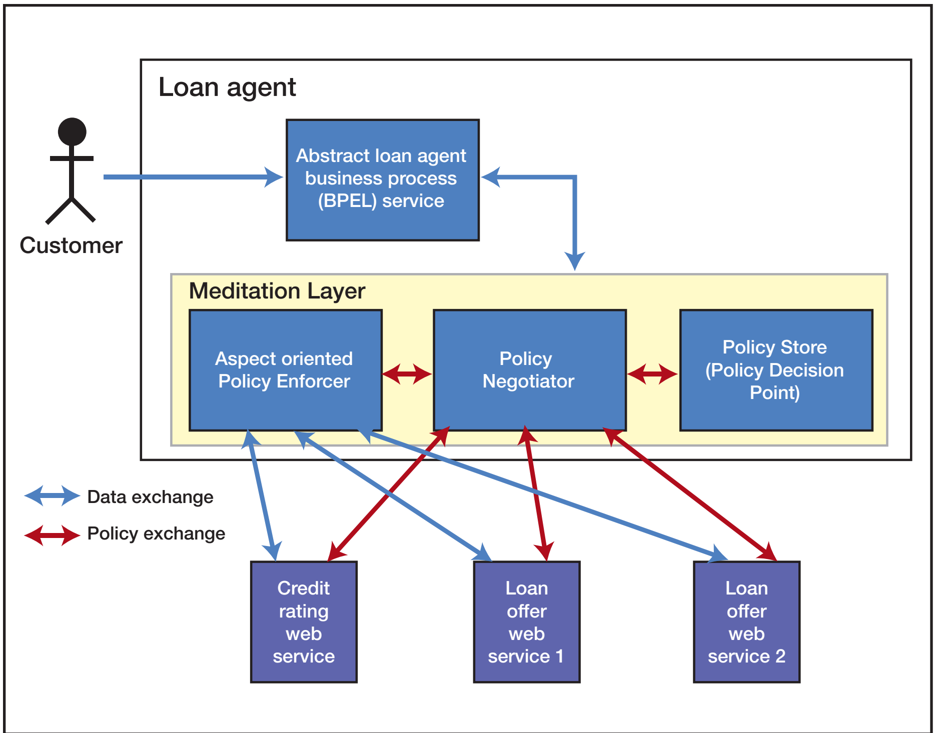


FIGURE 3 **Aspect-oriented mediation layer**

the application of aspect-oriented software development (AOSD) to handle the dynamic changes for Web services.

AOSD addresses the first two of the concerns discussed in the "Essential Considerations for Adaptation" section. It introduces units of modularity called "aspects" that contain code fragments called "advice" and hotspots of information called "pointcuts." Various advices can be introduced based on the kind of adaptation needed. Specifically with the "join-points" (a collection of join-points forms pointcuts) advices can be plugged in at the well-defined hotspots. Due to having various advices to be weaved, most of the aspect-oriented programming (AOP) models such as AspectJ have introduced relative precedence based on their ordering within the aspect body. The property-based pointcut designators facilitate the proactive insertions of advices and control flow–based pointcuts aid in the reactive insertions of advices.

The Role of Policies

We envisioned adaptation in services as the accommodation of new features on an as-needed basis. AOSD addressed the basic need of feature introduction, but to accommodate preferences of both the service requestor and the provider (e.g., late binding and negotiation) we use policy-driven techniques. Policy in Web services refers to a set of rules, facts, capabilities, and preferences that could regulate the behavior of the service. Policy assertions are used to convey the conditions between the interacting services. The service providers would advertise their policies for which the requestors might choose to opt. This would result in a specific behavior that reflects the accepted condition. The policy assertions could hence be used for providing the necessary preferences in adapted behavior of Web services over and above the functionality they embody.

As shown in the Figure 2, policy plays a central role in providing self-adaptation and autonomy to Web services, realizing specific instances of abstract compositions and accommodating preferences between interacting services. In order to facilitate coordination among adaptations we used weaving of aspects. In addition to this we apply policies to reconcile and decide which aspects needed to be woven on the fly. A hierarchy of policies can also be used to provide a rich set of behaviors.

How Do We Achieve Noninvasive Adaptations Using Aspects?

The core idea is that of binding aspects and policies to enrich Web services. In the aforementioned scenario, adaptations were induced in the various participating services by considering dynamic situations such as seasonal (Christmas and New Year) offers like converting EMI-based loans to personal line of credit, variation in scoring in credit reports, and security variations such as choosing the algorithms for encryption of relevant information in messages. The following are details of the participating services:

- ***Loan Agent Service:*** A composition service invoked by the customers that orchestrates the various other services listed below. BPEL4WS, an industry-accepted standard was used to compose the orchestration. Oracle's BPEL process engine was used to host this service.
- ***Credit Rating Service:*** A simple Web service that gives the credit score, given the unique identity of the customer. This service is only accessible to the loan agents who are registered prior to the requisition. The authorized loan agents are identified through a simple username/password–based authentication. This service provides various means of specifying the credentials. Also the credit scores are provided on varying scales such as 400-900 or 100-point scale.
- ***Loan Offering Services:*** Two simple Web services that offer loans based on the individual's credit score, the amount requested, and the repayment period. These services provide various offers such as discounts on interest rates and conversion of type of loan at regular intervals based on the market conditions.

Because AOSD provided a paradigm to address adaptability, policy-driven aspect selection was adopted to enrich the aforementioned Web services and their variations. A mediation layer was plugged into the framework in between the BPEL process and the external Web services as shown in Figure 3. This layer is responsible for the enforcement of the required policies and the enabling of the relevant aspects. This layer weaves necessary aspects to achieve adaptability within the loan agent service. The aspects would be the enforcement plugs for the policies chosen (policy enforcement point – PEP). A policy store acting as decision point (policy decision point – PDP) stores and fetches various policies based on the contextual information provided to the mediation layer. The policies are expressed using the simple and flexible grammar provided by Sun's Extensible Access Control Markup Language (XACML). XACML provides a model where policies are expressed using XML notations and also has the provision to evaluate policies to label as accept, deny, not applicable, or indeterminate state. To negotiate the policies between the loan agent service and external services, a policy negotiator is used. Once the policies are accepted the negotiator enables the aspects that are woven into the PEP.

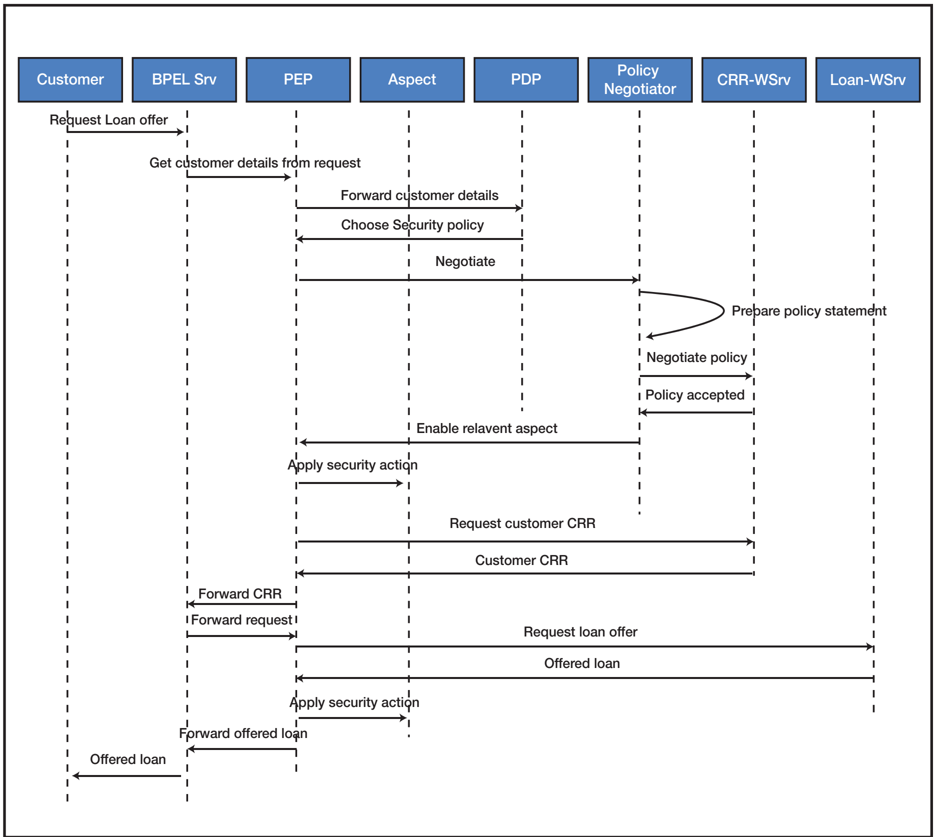


FIGURE 4 **Policy negotiation by the mediation layer**

> "The behavioral adaptation is to be considered in the context of interaction between the service provider and the requestor"

To understand the weaving of aspects that bring about necessary adaptations, let's consider a section of the interaction between the loan agent service and the credit rating service (CRS) as shown in Figure 4. When the request is made the orchestration facilitates the extraction of the customer's unique identification from the customer request and constructs the message requesting the

> A hierarchy of policies can also be used to provide a rich set of behaviors

credit score to be sent to the CRS. Rather than being sent directly to the CRS, this request gets intercepted by the mediation layer. The PEP in the mediation layer receives the message and based on the message definition the endpoint is selected from the configuration. The PDP then decides on all of the policy actions that are to be associated with the CRS such as the usage of the PBEWithMD5AndDES algorithm for sending credentials. These actions are then forwarded to the policy negotiator that creates policy statements as shown in Figure 5. The policy statements are then exchanged with the CRS, which either accepts or denies them. The policy actions accepted have associated aspect advices that are enabled. So when the PEP forwards the request to the CRS, it passes through the aspect that adds the credentials using WS-Security standards through the use of the accepted algorithm.

Figure 6 gives a snapshot of the PEP and an associated simple aspect that uses the hotspots for adaptations. Because the PEP is a service itself, it has a process operation that receives the messages. It also provides another operation called induceVariations that is used to mark hotspots within the PEP. This operation takes in the message and returns the modified message. As shown, the hotspots are marked before sending the request message and then after receiving the response message. The hotspots and the aspect's advices are triggered, thereby implementing necessary changes as required either during the assignment or before or after invocation. In the interaction between the loan agent and the CRS when the induceVariations operation is called before calling the CRS, the pointcut would add the credentials to the request message. Then returning from the induceVariations operation the modified request message is assigned to the context of the Web service call, thus facilitating transparent (with regard to the PEP or loan agent service) adaptation.

Adaptations in Orchestrated and Choreographed Services

Composition of various independent activities within the view of business functionality is achieved through choreography or orchestration of Web services. The difference between choreography and orchestration is that in case of the latter, one of the services has control over defining the role within the composition. It is thus evident that orchestration languages such as BPEL4WS by nature provide some flexibility. For example in the scenario discussed above, the flexibility in the loan agent process is seen through dynamic partner binding, i.e., addition or deletion of loan offer service. However in case of practical scenarios, orchestrated services too have to support unplanned, on-the-fly changes. The changes however may not necessarily mean changes to the static view of the process (as with different partners), but rather over every instance of the process. Efforts in this regard such as AO4BPEL are adding extensions to the existing BPEL4WS specification, thus allowing specific adaptation for orchestrated services in a modular fashion.

In case of choreography, not one service truly owns the conversation. Because they are collaborative in nature, each party involved describes its part in the interaction and is completely dependent on the observable behavior of every service. The adaptation hence in this case is very essential for managing the logical dependencies between unknown interactions. Specifically adaptation that involves aspect binding plays a prominent role in planning such interactions. Moreover, using the policies will also allow dynamic interaction patterns that would closely follow the changes in business.

Conclusion

Flexibility is an important aspect of e-business that may be achieved by using Web services. The dynamic environment of e-business necessitates constant changes. Web service providers have to evolve so that unplanned changes can be adapted to the contextualized information. Initiatives such as AdaptiveBPEL and AO4BPEL, which have explored ways of transparently allowing customizations. The use of aspect-oriented approaches and policies seems promising for on-the-fly adaptations. A middleware made up of a mediation layer based on policy-driven selection of aspects would especially result in realizing faster time to market for truly adaptive Web services.

The source code for the article accompanies the online version of the article, which can be found in the SOA WSJ archives at http://webservices.sys-con.com/read/issue/archives/, (Vol.6, iss.2).

References

- *Amazon Web services:* www.amazon.com/webservices.
- Gisolfi, D. "An Introduction to Dynamic e-Business": www-128.ibm.com/developerworks/library/ws-arc1/
- *Web Services Architecture, W3C working group note, February, 11 2004:* www.w3.org/TR/ws-arch/
- Erradi, A., Maheshwari, P., Padmanabhuni, S. "Towards a Policy-Driven Framework for Adaptive Web Services Composition," NWeSP 2005, Seoul, Korea.
- Cibrán, M.A., Verheecke, B. "Dynamic Business Rules for Web Service Composition," Second Dynamic Aspects Workshop 2005.
- Ortiz, G., Hernández, J., Clemente, P. J. "Reusable Web Service Choreography and Orchestration Process," Workshop on Engineering Service Composition 2005.
- Charfi, A., Mezini, M. "Using Aspects for Security Engineering of Web Service Compositions," International Conference on Web Services 2005.
- *AspectJ project:* http://eclipse.org/aspectj

About the Authors

Naveen Kulkarni specializes in Web services and service-oriented architecture. He currently works with the Web Services Centre of Excellence in SETLabs, the technology research division at Infosys Technologies, India. He has published papers in international conferences such as the IEEE International Conference of Web Services and has spoken at various industry forums exclusively on Web services. His interests include enterprise adoption of services, legacy modernization, and capacity planning, as well as QoS in Web services ecology.

■ ■ ■ naveen_kulkarni@infosys.com

Geo Philips Kuravakal currently works with the Web Services Centre of Excellence in SETLabs, the technology research division at Infosys Technologies, India. He is currently involved in the development of an enterprise Web service integration framework. His primary area of interest is the Semantic Web and its applicability to Web services, along with newer programming techniques such as AJAX.

■ ■ ■ geo_kuravakal@infosys.com

Lipika Sahoo currently works with the Web Services Centre of Excellence in SETLabs, the technology research division at Infosys Technologies, India. Her core area of work involves dynamic adaptability and management of Web services. She is currently involved in various research activities within the group relating to MDA and AOSD.

■ ■ ■ lipika_sahoo@infosys.com

Sunny Saxena currently works with the Web Services Centre of Excellence in SETLabs, the technology research division at Infosys Technologies, India. His interests range from Web service security platforms to aspect-oriented development models.

■ ■ ■ sunny_saxena@infosys.com

Manivannan Gopalan specializes in legacy systems, legacy migration to SOA, and Web services. He currently works with the Web Services Centre of Excellence in SETLabs, the technology research division at Infosys Technologies, India. He has published papers in international conferences such as the IEEE International Conference of Web Services.

■ ■ ■ manivannan_gopalan@infosys.com

Listing 1: Sample policy statement

```
<?xml version="1.0" encoding="UTF-8"?>
<Request ...>
    <Subject>
        <Attribute AttributeId="urn:oasis:names:tc:
xacml:1.0:subject:subject-id"
                    DataType="http://www.w3.org/2001/
XMLSchema#string">
            <AttributeValue>LoanAgent</AttributeValue>
        </Attribute>
    </Subject>
    <Resource>
        <Attribute DataType="http://www.w3.org/2001/
XMLSchema#string"
          AttributeId="urn:oasis:names:tc:xacml:1.0:resource:
resource-id" >
            <AttributeValue>http://localhost:8080/axis/
HLoan</AttributeValue>
        </Attribute>
    </Resource>
    <Action>
        <Attribute AttributeId="urn:oasis:names:tc:
xacml:1.0:action:action-id"
                    DataType="http://www.w3.org/2001/
XMLSchema#string">
            <AttributeValue>ChristmasOffer</
AttributeValue>
        </Attribute>
    </Action>
</Request>
```

Listing 2: Code snippet showing hotspots in PEP

```
/* SimplePEP.java */
...
public void process(SOAPEnvelope req, SOAPEnvelope resp)
throws AxisFault{
  try {
    // Get the endpoint for the message based on encoded
rules
    ...
    // Get the actions associated with the endpoint and
then
    // forward the call to the policy negotiator.  The
negotiator lookups
    // the policy store of the necessary policy state-
ments, if the policy
    // statements are accepted then necessary aspects are
enabled
    Collection actions = SimplePDP.
getActions(destination);
    SimpleNegotiator negotiator = new SimpleNegotiator();
    negotiator.negotiate(destination, actions);

    /** variation hotspot -- begin **/
    Message variedReqMsg = induceVariations(currCtx.getRe-
questMessage());
    /** variation hotspot -- end **/

    // create the call to the external web services
    ...
    call.setRequestMessage(variedReqMsg);
    ...

   /** variation hotspot -- begin **/
   Message variedResMsg = induceVariations(currCtx.getRe-
sponseMessage());
   currCtx.setResponseMessage(variedResMsg);
   /** variation hotspot -- end **/

  }catch(Exception ex){
     // handle exception
  }
}

public Message induceVariations(Message env){
  Message msgForVariation;
  msgForVariation = msg;
  return msgForVariation;
}

/* -- Aspect definition -- */
aspect FunctionalAdaptations{

  pointcut variation () : execution(* induceVaria-
tions(..));

  before () : variation() {
    // the necessary code to do the changes before chang-
es are assigned
    // to the new SOAP Message object
  }

  after () : variation() {
    // the necessary code to do the changes after changes
are assigned
    // to the new SOAP Message object
  }
}
```

Asynchronous Web Services Using WS-Addressing

Developing asynchronous Web applications using callback services

Today's IT organizations have tens of applications and services that perform some well-defined tasks such as inventory, billing, expense reporting, and order entry. With the evolution of Internet and e-business, enterprises have started to think about how different applications in a disconnected mode can work independently but at the same time be a part of an information workflow process.

WRITTEN BY
RAGHU KODALI

This new demand brought us to the concept of building new applications and integrating them with existing ones. Integrating new and old applications becomes a very complex task, because of both the number of applications that enterprises have as well as the complex business workflow of the enterprise.

Service-oriented architecture (SOA) provides an architectural pattern that can help to address the integration problem in the enterprise. Web services constitute one of the most viable solutions, and could be used as an implementation vehicle for SOA to integrate old, existing, and new applications using synchronous and asynchronous communication and a loosely coupled transaction model.

Asynchronous Applications

Enterprise applications or business processes that have information or process workflow typically are long running and asynchronous in nature. For example, a portal or Web application can provide a user interface to submit a loan application(s) to a loan flow business process, where the business process is orchestrating a discrete set of Web services such as credit service and third-party loan brokerage services. This loan flow business process could be a long running one in order to cater to the workflow (automated or human) involved, collect the loan offers, and finally come up with a best loan offer.

Web Services can be used not only for integration among applications, data, and services, but they can also support an asynchronous communication model. Applications or services can communicate by exchanging messages in an independent fashion without having to wait or block resources on the service consumer side.

A service consumer (which can be a any kind of Web application, desktop application, or process) invokes a Web service or business process by sending a message/request, which is typically an XML document. The service consumer does not have to wait for the message/response or block the resources with continuous wait time or by polling mechanism, which can become very expensive in terms of processing overhead. For example, in case of the loan flow scenario described above, the client application, which in this case is a Web application, can keep polling the business process to find out the whether the loan has been processed or not by using polling mechanism. Figure 1 shows a client application submitting the loan application that keeps polling until it receives a loan offer.

Asynchronous Web Services

Instead of polling the enterprise application, business process, or Web service, we could adopt an asynchronous communication model between the client and the application, service, or business process. Asynchrony can be achieved by applying different architectural models to Web services. The key requirement for choosing asynchronous communication here is the service consumer or the client cannot wait for the response to come back. This could be due to many reasons that include holding up the system resources on the client side, the network connectivity, or that it just takes an unpredictable amount of time for the

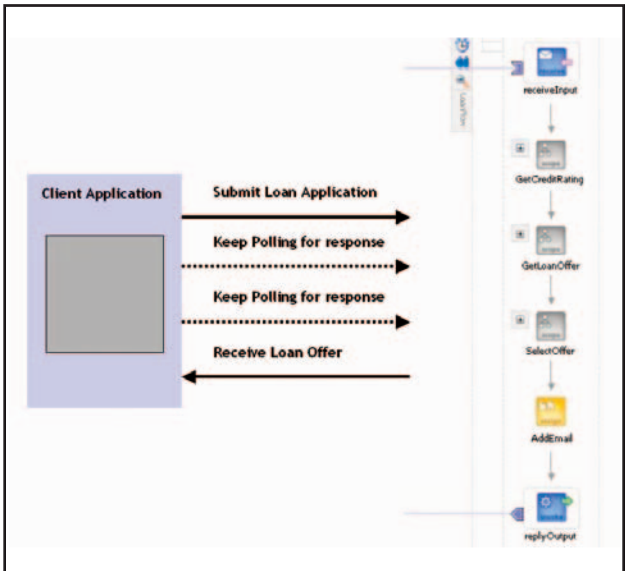


FIGURE 1 **Asynchrony with polling**

response to come back. Typical asynchronous Web services architectural models are:

- A service consumer can send a SOAP message without having to block any resources to receive a message back.
- A service consumer and service provider can go through a series of conversations by passing around a conversation ID. This type of usecase is known as *conversational* Web services.
- A service consumer invokes a service, and the provider will send a callback once the process is completed. In order to receive the callback response, the service consumer will have a Web service endpoint that the provider uses to send the message back. Figure 2 shows a service consumer that has an endpoint and submits a loan application and the service provider, which is a business process and sends a callback to the consumer callback endpoint once the process is complete.

In order for the service provider to send the message/response back, the consumer will have to provide callback information or the details about where the notification or reply has to be sent. The Web Services Addressing (WS-Addressing) specification, which has been developed under World Wide Web Consortium (W3C), provides a standard mechanism to provide the callback information in the SOAP header element.

WS-Addressing

WS-Addressing provides way to specify message addressing information independent of transport layer. WS-Addressing provides a way to specify delivery, reply-to, and fault-handler addressing information in a SOAP envelope. WS-Addressing can be used in conjunction with other specifications such as WS-Security to authenticate and WS-Policy to define policies for the service.

WS-Addressing has two key constructs or artifacts:

- Endpoint reference
- Message information headers

Endpoint references in the WS-Addressing specification contain the information needed to identify/reference a Web service endpoint. Endpoint references can also be used to provide addresses for individual messages sent to and from Web services.

The WS-Addressing specification defines a set of message information headers that allow uniform addressing of messages independent of underlying transport. These message information headers convey end-to-end message characteristics, including addressing for source and destination endpoints as well as message identity. The WS-Addressing specification comes with WS-Addressing schema, which has complex types defined for endpoint references and messaging properties.

The endpoint reference contains the properties shown in Table 1. Only the address property (URI) is mandatory. The rest are used to provide additional information, behavior, and requirements of the endpoint. Table 2 shows the message information header properties.

Listing 1 shows a SOAP message that is

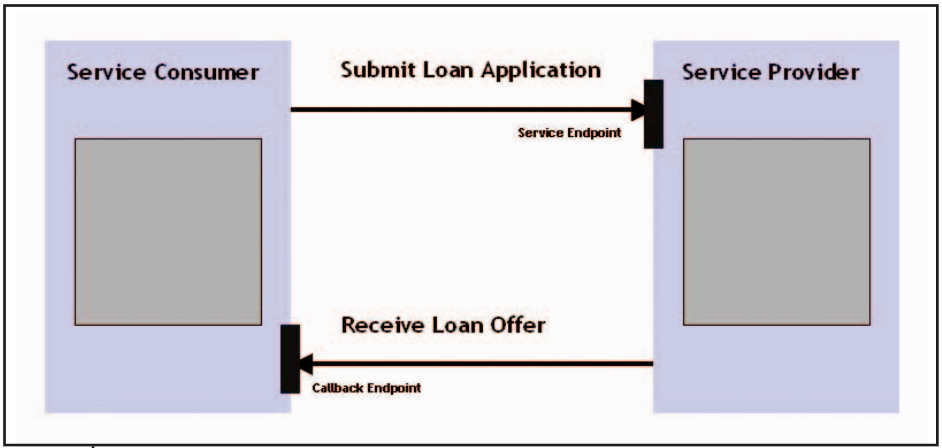


FIGURE 2 **Asynchrony with callback architecture**

Property	Description
Address	URI that points to a network address.
Reference Properties	Collection of properties that can be attached from an application specific namespace and that are used to identify a service. An endpoint reference might have a property that tells the service that it has to handle a customer whose status is Gold.
Reference Parameters	Collection of properties that can be attached from an application specific namespace and used to identify a resource.
Selected Port Type	Qualified name (QName) of portType of the endpoint. Similar to the portType in WSDL.
Service Port	Qualified name (QName) that identifies the WSDL service element.
Policy	Can contain WS-Policy elements that describe the behavior and requirements of the endpoint.

TABLE 1 **Properties of the endpoint reference**

Property	Description
Destination	Address of the intended receiver.
Source Endpoint	Where the message originated or the source of the message.
Reply Endpoint	Endpoint reference that identifies the receiver of the replies. If reply is expected, the message should contain this property. If this property is present in the message, then the message ID property becomes mandatory.
Fault Endpoint	Endpoint reference that identifies the receiver for all fault messages. If this property is present in the message, then the message ID property becomes mandatory.
Action	URI that identifies the semantics of the message.
Message ID	Unique identifier for the message.
Relationship	Complex type of element that indicates how one message is related to another one.

TABLE 2 **Properties of the message information header**

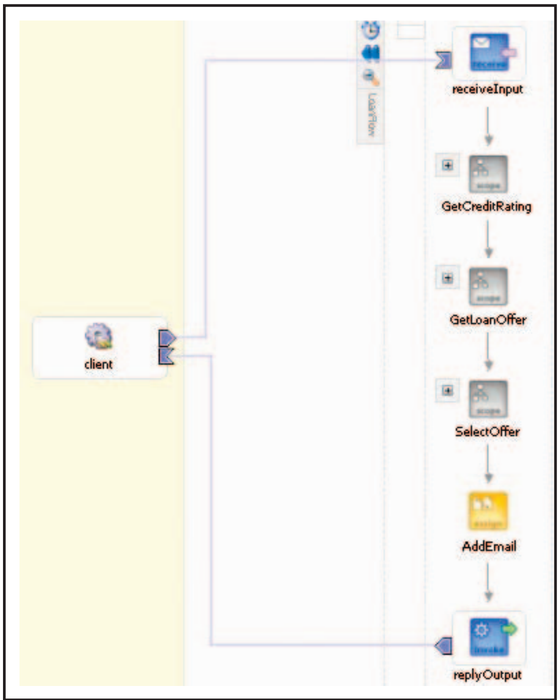


FIGURE 3 **Loan flow business process**

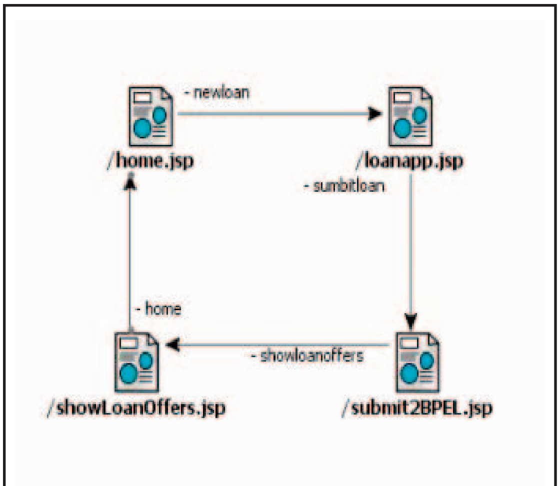


FIGURE 4 **Web application page flow**

used to invoke the loan flow business process, and contains the WS-Addressing information in the header element. WS-Addressing, in simple terms, is very similar to how the "to" and "from" addresses are written in postal envelopes.

Implementing WS-Addressing Services

As discussed in earlier sections of this article, client applications or service consumers expecting a response back in asynchronous fashion need to provide an endpoint, which is capable of receiving message/response back from the other end or from the service provider. Web applications and desktop applications aren't really geared toward or designed to work as callback clients. In order to provide or mimic the asynchronous conversations, we can have an intermediate service layer or a Web service, which could act as a callback client to the service provider or business process invoked by the client applications. Once the callback Web service receives the response it can process the callback message and take further action that will complete the loop for the user who has invoked a service or business process from the client application.

In the next part of this article we will illustrate the aforementioned use case with a business process developed using Business Process Execution Language (BPEL), a callback Web service developed using JAX-RPC and a Web application developed with JSF that acts as a service consumer and invokes the BPEL business process. Oracle Fusion Middleware components (Oracle BPEL Process Manager, OC4J, Oracle JDeveloper) were used to develop the above use case.

To start, we have a loan flow business process developed using BPEL (see Figure 3). This loan flow process checks the credit worthiness of the loan applicant by invoking a credit rating service provided by a credit rating company. Once the applicant is determined to have a good credit history, the loan flow process interacts with services provided by mortgage brokers and gets multiple loan offers. Finally the loan flow process has orchestration logic that determines the best loan offer from the offers received.

Once the back-end business process is ready, we need a Web application that is built with JavaServer Faces (JSF), which the customer uses to submit loan applications. Figure 4 illustrates the JSF navigation model for the Web application. Once the customer submits a loan application, the JSF application invokes the loan flow BPEL process (which is a Web service) using the stubs generated by the utilities provided by Oracle Application Server.

Once the JSF application invokes the loan flow business process, the business process will determine the best loan offer. We would like to notify the customer who has submitted the application via e-mail or SMS message on the status of the loan application and the loan offer that comes out of the loan flow process.

To achieve this we will create an intermediate Web service, which we will call the loan flow callback service. This loan flow callback service will have an endpoint that the BPEL loan flow business process can use to send the callback or loan offer that will be further processed by callback service.

Once the loan flow callback service is in place, we need to process the incoming loan offers. To keep it simple we will just send an e-mail notification to the customer who has applied for the loan, and he will be able to pick the message from the JSF application and act upon it either to take the offer or reject it. Listing 2 shows the `onResult()` method, which is the operation that receives the callback from the loan flow business process and sends out

> "Applications or services can communicate by exchanging messages in an independent fashion without having to wait or block resources on the service consumer side"

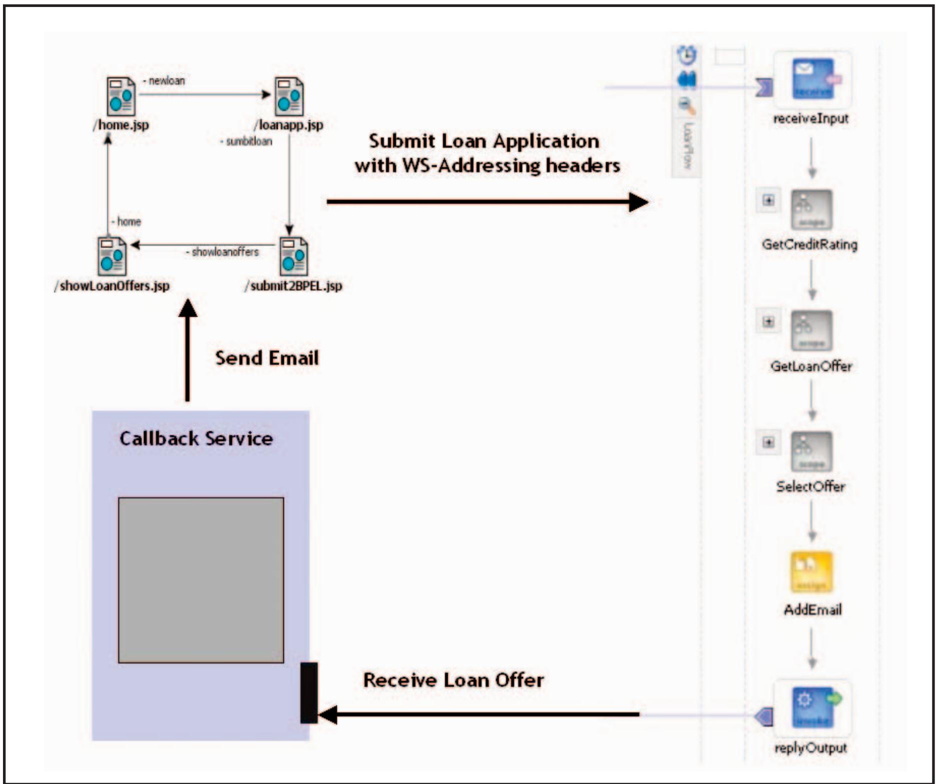


FIGURE 5 **Asynchronous loan flow with WS-Addressing**

an e-mail with the loan offer.

So far so good: one additional thing that needs to be taken care of during the invocation of loan flow business process is to make sure that the SOAP message header element is populated with WS-Addressing information. The source code in listing 3 constructs a SOAP message, which has the loan application and populates the header of this SOAP message with WS-Addressing information that will tell the BPEL business process to send a callback to loan flow callback Web service after the loan application has been processed and the best offer has been achieved. Figure 5 illustrates the completed use case with flow described above.

Summary

To summarize, Web services constitute one of the most viable options to implement standards-based integration in enterprises. The WS-Addressing specification provides a standard mechanism to send message information independent of the transport layer.

WS-Addressing provides number of benefits that include the transport layer not being restricted to HTTP, and working in conjunction with other WS-* specifications it can be used for different patterns such as request/response, one-way, or conversational Web services.

Acknowledgments

The author wishes to acknowledge Olivier Le Diouris (olivier.lediouris@oracle.com), solution architect in Oracle Fusion Middleware Group, for his assistance in completing the aforementioned use case and source code.

About the Author

Raghu R. Kodali is consulting product manager and SOA evangelist for Oracle Application Server. He leads next-generation SOA initiatives and J2EE feature sets for Oracle Application Server, with particular expertise in EJB, J2EE deployment, Web services, and BPEL. He holds a Masters degree in Computer Science and is a frequent speaker at technology conferences. Raghu is also a technical committee member for the OASIS SOA Blueprints specification, and a board member of Web Services SIG in OAUG. He maintains an active blog at Loosely Coupled Corner (www.jroller.com/page/raghukodali).

■ ■ ■ raghu.kodali@oracle.com

Listing 1

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/
envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns0="http://schemas.xmlsoap.org/ws/2003/03/address-
ing" xmlns:ns1="http://www.autoloan.com/ns/autoloan">
   <env:Header>
      <ns0:ReplyTo>
         <ns0:Address>http://localhost:8888/
LoanFlowCallbackService/LoanFlowCallbackService</ns0:
Address>
         <ns0:PortType>null:LoanFlowCallback</ns0:
PortType>
         <ns0:ServiceName>null:LoanFlowCallback</ns0:
ServiceName>
      </ns0:ReplyTo>
      <ns0:MessageID>urn:my-unique-id</ns0:MessageID>
   </env:Header>
   <env:Body>
      <ns1:loanApplication>
         <ns1:SSN>123-45-6789</ns1:SSN>
         <ns1:email>myself@home.org</ns1:email>
         <ns1:customerName>Maurice Poulet</ns1:customer-
Name>
         <ns1:loanAmount>1234.56</ns1:loanAmount>
         <ns1:carModel>BigCar</ns1:carModel>
         <ns1:carYear>1999</ns1:carYear>
         <ns1:creditRating>50</ns1:creditRating>
      </ns1:loanApplication>
   </env:Body>
</env:Envelope>
```

Listing 2

```
 public void onResult(String providerName,
                        boolean selected,
                        boolean approved,
                        double APR,
                        String email,
                        Relationship relatesTo)
    {
      System.out.println("Result for " + email);
      try
```

```
        {
          String content = "Your loan has been " + (approve
d?"approved":"rejected") + " by " + providerName;
          EMailClient.send(email, "Your Loan", content);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
     }
```

Listing 3

```
public  void invokeLoanFlow() {

//…..
          try {
              loanclient.LoanFlowPortClient myPort = new
loanclient.LoanFlowPortClient();
              System.out.println("calling " + myPort.
getEndpoint());
              // Add your own code here

               String input = "Oliv";
               AttributedURI uri = new AttributedURI();
               uri.set_value(new URI("http://local-
host:8888/LoanFlowCallbackService/LoanFlowCallbackService
"));

               AttributedQName portQname = new
AttributedQName();
               portQname.set_value(new QName( "http://call-
back.service/", "LoanFlowCallback"));
               ServiceNameType serviceName = new
ServiceNameType();
               serviceName.set_value(new QName("http://
callback.service/", "LoanFlowCallback"));

               EndpointReferenceType replyTo = new
EndpointReferenceType();

               replyTo.setAddress(uri);
               replyTo.setPortType(portQname);
               replyTo.setServiceName(serviceName);
               AttributedURI messageID = new
AttributedURI();
               URI mURI = new URI("urn:my-unique-id");
               messageID.set_value(mURI);

               double loanAmount = 1234.56;
               int creditRating = 50;
               myPort.initiate("123-45-6789", "myself@home.
org", "Maurice Poulet", loanAmount, "BigCar", "1999", cre-
ditRating, replyTo, messageID);
               System.out.println("Invoked BPEL Process");



          } catch (Exception ex) {
              ex.printStackTrace();
          }
//…………
}
```

XML JOURNAL SUPPLEMENT

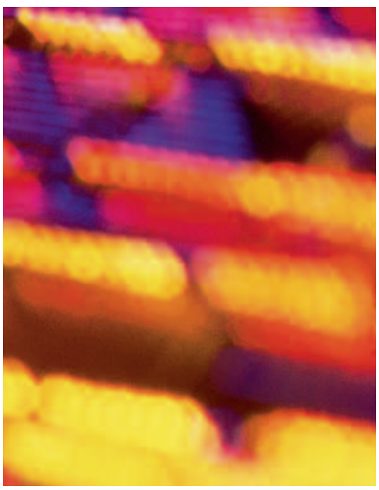
HOME
Enterprise Solutions
Content Management
Data Management
XML Labs

This Month

Getting to Know Xindice

Selim Mimaroglu

Apache Software Foundation's Xindice is an open source native XML database. Apache provides great software to developers such as the Apache Web Server, Tomcat Application Server, Cocoon Web Development Framework, Struts Framework, Ant, and many more under an open source license. Apache Web Server is an industrial-strength product that is used by many high-traffic Web sites. Apache Tomcat is a servlet container that implements both Java Servlets and Java Server Pages. The fact that Xindice is a member of the Apache Software Foundation gives it substantial credibility, because Apache produces well-known, well-respected software. We will explore the Xindice native XML database beginning with its installation and advancing step by step. As the Xindice home page suggests, you are encouraged to pronounce it as "zeen-dee-chay" with your best faux Italian accent. Xindice supports XPath query language, XML:DB, and XUpdate standards in Java, in addition to XML-RPC API for non-Java programmers.



XML-Based Interop, Close up

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

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

Content Management: Organization, dissemination, and presentation of information

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

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

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

Getting to Know Xindice

An open source native XML database

WRITTEN BY
Selim Mimaroglu

Apache Software Foundation's Xindice is an open source native XML database. Apache provides great software to developers such as the Apache Web Server, Tomcat Application Server, Cocoon Web Development Framework, Struts Framework, Ant, and many more under an open source license. Apache Web Server is an industrial-strength product that is used by many high-traffic Web sites. Apache Tomcat is a servlet container that implements both Java Servlets and Java Server Pages. The fact that Xindice is a member of the Apache Software Foundation gives it substantial credibility, because Apache produces well-known, well-respected software. We will explore the Xindice native XML database beginning with its installation and advancing step by step. As the Xindice home page suggests, you are encouraged to pronounce it as "zeen-dee-chay" with your best faux Italian accent. Xindice supports XPath query language, XML:DB, and XUpdate standards in Java, in addition to XML-RPC API for non-Java programmers.

Installation

You can download both the source files and the binaries from the home page of Xindice at http://xml.apache.org/xindice/. As I write, the latest stable release is 1.0, and the current milestone is 1.1b4. I downloaded the binary files in both webapp and .jar formats. You don't need to download the source files if you aren't interested in reading or modifying them.

Version 1.1 of Xindice doesn't run as a stand-alone server, but instead runs under a servlet 2.2-compliant application server. The documentation states that it's tested both with Tomcat and Jetty. Since Tomcat is very popular and I am familiar with it, I decided to run Xindice with Tomcat. You can download Tomcat from http://tomcat.apache.org. I downloaded the Windows executable, binary core distribution version 5.0.28.

Because I already have a Java SDK (Software Development Kit) installed on my machine, I didn't need to download it again, but if you don't have a Java SDK, you have to download version 1.4.x of J2SE (Java 2 Standard Edition) from http://java.sun.com.

I installed Xindice and Tomcat and tested all of the examples in this article on my laptop, a 1.4GHz Pentium M processor, running on Windows XP Professional with 512MB main memory, without any problems. These are the installations steps I followed:

1. Install Java 1.4.x SDK.
2. Set JAVA_HOME environment variable to the installation directory (on my system the value of JAVA_HOME is D:\j2sdk1.4.2_03).
3. Install Tomcat. (This step is straightforward because Tomcat comes with an installer. By default Tomcat will install under C:\Program Files, with a space between the words Program and Files: unfortunately there will be some problems when running Xindice, so I changed the installation location to D:\Tomcat_5.0. Any location will do as long as it doesn't have any space in it.)
4. Unzip the Xindice webapp distribution, and rename the .war file (xindice-1.1b4.war) to xindice.war.
5. Drop xindice.war (the extension war stands for web archive) under Tomcat's webapps directory (this is D:\Tomcat_5.0\webapps on my system).
6. Start Tomcat using the "Configure Tomcat" application (it's located under Start->All Programs->Apache Tomcat 5.0-

"You can have several XML files under a collection, which is a neat way of organizing the documents"

>Configure Tomcat).

7. In a few minutes Tomcat will unzip the .war file. When it does, you should see a new Xindice directory under D:\Tomcat_5.0\webapps.
8. Now it's time to create a new environment variable called XINDICE_HOME. If you followed the locations I suggested, XINDICE_HOME should have D:\Tomcat_5.0\webapps\xindice\WEB-INF as its value. If you didn't, modify the value accordingly.
9. As the last step, put D:\Tomcat_5.0\webapps\xindice\WEB-INF under your Path since it contains the Xindice's command-line tool.

Starting the Server

Starting or stopping the Xindice native XML database means starting or stopping the Tomcat Server. Tomcat by default runs on port 8080, and you can change the port during or after the installation. To check if your Tomcat is running properly, after starting it visit http://localhost:8080/ (or http://127.0.0.1:8080/): you should see the welcome page.

Command-Line Tool

We can perform the following tasks using the command-line tool:

- Add/remove a collection
- List collections
- Add/remove a document
- Retrieve documents
- Import/export a directory tree
- Add/remove a collection index
- Execute an XPath query

Listing Collections

Let's list the collections installed by Xindice after the installation. Remember that the server should be running in order to execute this command successfully:

```
> xindice list_collections -c xmldb:
xindice://
  localhost:8080/db

trying to register database
        system
        meta
Total collections: 2
```

Each command has an abbreviated version. For example, we could have used lc instead of list_collections. For clarity, in this article I will use the full name of the commands. After familiarizing yourself with Xindice, you probably won't want to write the commands in full. You can find the abbreviated versions and the descriptions of all the commands at the Xindice Commandline Tool Guide.

In this example, the full context is xmldb:xindice://localhost:8080/db, which is specified by the –c option. When using Tomcat we have to specify the full context: Tomcat runs on port 8080, but Xindice is preconfigured for Jetty, which runs on port 8888. If you don't want to specify the full context, you have two options: change the Tomcat's port to 8888 or install Jetty instead of Tomcat. If the server is running on a different machine and not on the localhost, you should replace the localhost portion of the context with the correct URL.

After executing this command the system responded by telling us that there are two collections: systems and meta. Like the other popular open source native XML databases such as eXist and Berkeley DB XML, Xindice too uses the idea of collection. A collection is similar to a folder in terms of functionality. You can have several XML files under a collection, which is a neat way of organizing the documents.

Adding a Collection

Let's create a new collection called dictionary:

```
> xindice add_collection -c xmldb:xin-
dice://
  localhost:8080/db -n dictionary

trying to register database
Created : xmldb:xindice://local-
host:8080/db/dictionary
```

As in the previous example, –c specifies the context that is the root location (all of the data files and collections are stored under this location). By executing this command successfully we created a subcollection called dictionary. The context of this new location is xmldb:xindice://localhost:8080/db/dictionary.

Listing the collections reveals that now we have three collections, of which two of them were previously created.

```
> xindice list_collections -c xmldb:
xindice://
  localhost:8080/db

trying to register database
```

```
        dictionary
        system
        meta
Total collections: 3
```

Adding a Document

I created a small sample XML file for using in the following exercises. The 150KB file is available at www.cs.umb.edu/~smimarog/xmlsample/.

```
> xindice add_document -c xmldb:xindice://localhost:8080/
db/
  dictionary -f C:\xindice_dictionary.xml -n dict150k

trying to register database
Added document xmldb:xindice://localhost:8080/db/diction-
ary/
  dict150k
```

Notice the context in this example is xmldb:xindice://localhost:8080/db/dictionary. The –f option specifies the location of the XML file to be added. The –n flag is optional. If used, whatever follows it will be the key of the document. If you don't provide a key, Xindice will create it for you automatically. Unfortunately, Xindice doesn't support XML files bigger than 5MB. Documents larger than 5MB can be divided and then stored in the database. Dividing the files into smaller pieces is not very convenient, and sometimes it's technically impossible. I hope Xindice developers will fix this soon in a future release.

Indexing a Collection

Xindice provides fine granularity indexing; that is, we can select the elements and the attributes for indexing. It's also possible to index all of the elements and attributes. If you know the workload, you know which elements and attributes are invoked, and you can create indexes on these elements only. For unpredictable, ad hoc environments, I think it's best to index all of the elements and attributes. We can index all elements by using the * pattern as the example below demonstrates.

```
> xindice add_indexer -c xmldb:xindice://localhost:8080/
  db/dictionary -n elementindex
-p '*'

trying to register database
CREATED : elementindex
```

The –n option specifies the name of the index and –p option specifies the pattern. In this example * (star) stands for every element.

Similarly we can index all of the attributes by using the pattern *@*:

```
> xindice add_indexer -c xmldb:xindice://localhost:8080/
  db/dictionary -n attributeindex -p '*@*'
trying to register database
CREATED : attributeindex
```

The patterns * and *@* make sense if you think in terms of XPath. In XPath, * stands for any element, and *@* stands for any element's attribute.

Indexing is very important if you want to have good performance. Xindice uses Apache Xalan as its XPath query processor. If you are familiar with Xalan you may remember that it doesn't have an indexing mechanism, but Xindice provides indexing and query-optimization facilities to speed up query performance on top of Xalan.

Executing XPath Queries

The XML Schema for our sample XML file, XML Schema of TCSD, is located at www.cs.umb.edu/~smimarog/xmlsample. You may want to check the schema to understand the structure of the sample document we are about to use. *dictionary* is the root element, *e* stands for

> "Indexing is very important if you want to have good performance"

an entry, and *hw* stands for headword (see the Listings section for XPath Query1, 2, and 3 and Listings 1-3).

Notice that some attributes have been added to the results and were not part of the original document. The src:col, src:key attributes and src namespace definition are appended to the results. src:col is the collection, and src:key is the key of the XML file. Although in some cases this information may be useful, I am not a fan of any modification or addition done by the database server in order to make the results more convenient.

Programming with Xindice

Xindice provides the XML:DB API for Java programmers. XML:DB is a very popular API, which is supported by many native XML databases such as eXist and Tamino. Xindice also provides XML-RPC API for non-Java programmers. It's hard to find a programming language that doesn't support XML-RPC: PHP, Python, Tcl, C, C++, Perl, Lisp, Delphi, Dylan, Java, JavaScript, and many other languages support XML-RPC. Xindice supports updates through the XUpdate standard. If you are interested in programming with Xindice, I recommend reading Xindice Developer Guide, which has many good examples.

Acknowledgements

Thanks to Glenn Hoffman for reading and commenting on this article.

Author Bio

Selim Mimaroglu is a PhD candidate in computer science at the University of Massachusetts in Boston. He holds an MS in computer science from that school and has a BS in electrical engineering.

smimarog@cs.umb.edu

XPath Query1: This query retrieves all of the headwords located under the dictionary collection.

> xindice xpath -c xmldb:xindice://localhost:8080/db/dictionary –q “/dictionary/e/hwg/hw”

Listing 1: Results of XPath Query1

```
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>planks</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>gaul</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>minute</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>spot</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>slippery</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>adorns</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>homeward</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>serpents</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>gaunt</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>exclamations</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>disposer</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>walter</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>smilets</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>priamus</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>opinion</hw>
```

XPath Query2: This query retrieves all of the entries that have the headword “minute”

> xindice xpath -c xmldb:xindice://localhost:8080/db/dictionary -q “/dictionary/e[hwg/hw/text()=’minute’]”

Listing 2: Results of XPath Query2

```
<e id=”E2” src:col=”/db/dictionary” src:
key=”dict150k” xmlns:src=”http://xml.apache.org/
xindice/Query”><hwg><hw>minute</hw>
<pr>1/k96SMVL^2</pr>

<pos>n.</pos>

</hwg>
<et><cr>E143</cr>
<cr>E180</cr>
<cr>E530</cr>
<cr>E308</cr>
<cr>E215</cr>
…
</e>
```

XPath Query3: Retrieve the headwords of the entries containing the word “hockey”

> xindice xpath -c xmldb:xindice://localhost:8080/db/dictionary -q “/dictionary/e[contains(., ‘hockey’)]/hwg/hw”

Listing 3: Results of XPath Query3

```
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>planks</hw>
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>gaul</hw>
…
<hw src:col=”/db/dictionary” src:key=”dict150k”
xmlns:src=”http://xml.apache.org/xindice/
Query”>opinion</hw>
```