

SOA WORLDTM

MAGAZINE

www.SOA.SYS-CON.com

MARCH 2008 / VOLUME: 8 ISSUE 3

Service Reuse & Entitlement

Resolving access control tasks

10 MICHAEL POULIN

18 Service Oriented Infrastructure (SOI)
YOAV DEMBAK

24 SOAP Over JMS Interoperability
STANIMIR STANEV AND ROB BARTLETT

30 Next-Generation Service Infrastructure & the Semantic Challenge
JEAN-PIERRE LORRÉ





SERVICE LEVEL
AUTOMATION & PREDICTA**b**ILITY



END-TO-END LOGICAL
TO PHYSICAL MAPPING & VISI**b**ILITY

Deliver predictable service levels and scale operations cost-effectively with B-hive Conductor™. Conductor continuously monitors user transactions and leverages the power of virtualized infrastructure to provide an automated, real-time response to changing conditions.

Take a free, simple, fast and non-intrusive test drive. Visit: www.bhive.net/download



SCALA**b**LE COST-EFFECTIVE OPERATIONS

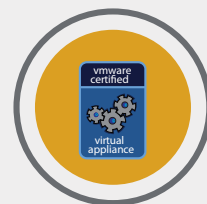
100% Organic
Virtual Appliance



- ◉ 0% Agents
- ◉ 0% Synthetic Transactions
- ◉ 100% Non-intrusive
- ◉ 100% Self-learning

 **b-hive**

Tel: (650) 358-1300



INTERNATIONAL ADVISORY BOARD

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

TECHNICAL ADVISORY BOARD

JP Morgenthal, Andy Roberts, Michael A. Sick, Simeon Simeonov

EDITORIAL

Editor-in-Chief

Sean Rhody sean@sys-con.com

XML Editor

Hitesh Seth

Industry Editor

Norbert Mikula norbert@sys-con.com

Product Review Editor

Brian Barbash bbarbash@sys-con.com

.NET Editor

Dave Rader davidr@fusiontech.com

Security Editor

Michael Mosher wsjsecurity@sys-con.com

Research Editor

Bahadir Karuv, Ph.D. Bahadir@sys-con.com

Technical Editors

Andrew Astor andy@enterprisedb.com
David Chappell chappell@sonicsoftware.com
Anne Thomas Manes anne@manes.net
Mike Sick msick@sys-con.com
Michael Wacey mwacey@csc.com

International Technical Editor

Ajit Sagar ajitsagar@sys-con.com

Executive Editor

Nancy Valentine nancy@sys-con.com

Associate Online Editor

Lindsay Hock lindsay@sys-con.com

PRODUCTION

LEAD DESIGNER

Abraham Addo abraham@sys-con.com

ASSOCIATE ART DIRECTORS

Louis F. Cuffari louis@sys-con.com
Tami Beatty tami@sys-con.com

EDITORIAL OFFICES

SYS-CON MEDIA
577 CHESTNUT RIDGE ROAD, WOODCLIFF LAKE, NJ 07677
TELEPHONE: 201 802-3000 FAX: 201 782-9637
SOA World Magazine Digital Edition (ISSN# 1535-6906)
Is published monthly (12 times a year)
By SYS-CON Publications, Inc.
Periodicals postage pending
Woodcliff Lake, NJ 07677 and additional mailing offices
POSTMASTER: Send address changes to:
SOA World Magazine, SYS-CON Publications, Inc.
577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677

©COPYRIGHT

Copyright © 2008 by SYS-CON Publications, Inc. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy or any information storage and retrieval system without written permission. For promotional reprints, contact reprint coordinator. SYS-CON Publications, Inc., reserves the right to revise, republish, and authorize its readers to use the articles submitted for publication. All brand and product names used on these pages are trade names, service marks, or trademarks of their respective companies. SYS-CON Publications, Inc., is not affiliated with the companies or products covered in Web Services Journal.



Put on a Happy Face(book)

WRITTEN BY SEAN RHODY

I had the opportunity recently to speak at a Microsoft event on Web 2.0. It was an interesting evening, with speakers from several organizations discussing various issues and strategies that could be used to move the bar forward on the Web.

Now if you're a faithful reader, you've seen me rant every so often about the inadequacies of browser technology when it comes to delivering applications over the Internet. The advantages are well known – zero footprint, controlled install base, etc., but the disadvantages have been known to drive me crazy. Anyone who's ever hit the back button after kicking off some process and had it end up completely botched knows exactly what I'm talking about.

So the things that I saw at the conference really made me feel better about the future of human interaction with applications. Not just from a browser perspective either, although technologies such as Silverlight and Flex are some of the first real steps I've seen in creating technology that isn't least common denominator in terms of controls, presentation, and communications. Finally we won't have to see the screen refresh itself each time we select a state from some drop down box. I spent a little time examining the two packages from a usability perspective and found that both of them had great potential for creating real user interfaces in a browser. I still worry about the browser back button though. But a video Microsoft showed at the conference around the future of interface technology had me drooling – what you might see in a sci-fi movie may be coming our way sooner than we think.

But in conjunction with the discussion of technology for Web 2.0, we also started to discuss some of the other aspects of the Web. One in particular is the phenomenon of social networking. I work as a consultant, and social networking or social computing is a topic that many firms are trying to wrestle to the ground and make sense of.

Blogs, Wikis, Facebook, and LinkedIn are just a few of the consumer-based examples of the social community building that occurs via the Web. People come together, across the globe, drawn by some common interest and become part of a virtual community. The effects of this can be enormous – look at how one man working on an operating system changed how the world view's computers, just by making his work open source and freely distributing it. It's not just computers that benefit from this sense of community. Any product or service that can develop a fan base or following provides countless business opportunities. Sometimes it's about using the service to deliver a message, sometime the service is actually the product, but if you can get people together on a particular subject, you have a powerful vehicle.

And this leads me, finally, around to the subject of this month's focus – SOI, Virtualization, and Grid services. An inevitable consequence of successful social networking is expansion. Maybe you need more computing power to render that massive online game for an ever-growing horde of players, or you need more bandwidth for certain parts of your service to provide better streaming and more efficient content delivery.

Virtualization allows us to move capacity in ways that are directly meaningful to the business. Running on top of a Grid infrastructure, the entire corporate ecosystem, or at least major portions of it, can be controlled to provide optimized service and efficiency. Want to scale down power usage in the evenings when most users are offline? Simply have the grid control the number of available processors and shut down unneeded CPUs and disks. Need to have some capability available during backup periods and downtime? Use virtualization to slice yourself a minimal capability for users to be content with instead of forcing them offline for a period of time. The possibilities are staggering. ■

About the Author

Sean Rhody is the editor-in-chief of SOA World Magazine. He is a respected industry expert and a consultant with a leading consulting services company. sean@sys-con.com



DOWNLOAD PRINT VERSION

3 Put on a Happy Face(book) SEAN RHODY

6 Bring the World of IT Applications & Communications Together with SOA JOHN BEDNAREK

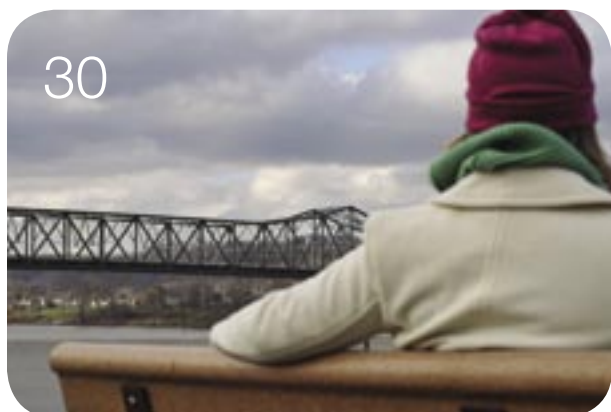
10 Service Reuse and Entitlement MICHAEL POULIN

18 Service Oriented Infrastructure (SOI) YOAV DEMBAK

20 Do You Have a DSG (Dumb SOA Guy) Issue? DAVID S. LINTHICUM

24 SOAP Over JMS Interoperability STANIMIR STANEV AND ROB BARTLETT

30 Next-Generation Service Infrastructure & the Semantic Challenge JEAN-PIERRE LORRÉ





Making decisions with only half the information is ... useless.

SOA is a moot point if you don't include your mainframe systems

There are a lot of vendors claiming they can solve integration issues, but very few have the know-how to actually get to the heart of your business — your legacy system. And if you can't get all the information, your plans for smooth integration and a service-oriented architecture come to a standstill.

Our time-proven technology lets you SOA-enable your host, so you can respond to new business demands economically, by maximizing the investments you've made in your legacy assets over the years. Without taking unnecessary risks.

Find out how AT&T, Freightliner, PPG Industries and other leading companies have quickly and cost effectively incorporated their legacy systems into Service Oriented Architectures. Download the white paper at www.attachmate.com/soa.



Bring the World of IT Applications & Communications Together with SOA

BY JOHN BEDNAREK

Service Oriented Architecture (SOA) is a business-driven concept based on a style of architecture that uses loosely coupled services and components to support the requirements of business processes and users. It is evolutionary in terms of its distributed computing approach (software running on multiple platforms) and modular programming style.

The value of SOA is in its ability to create “building blocks” of functions or services that can be rapidly and cost-effectively connected into the existing business infrastructure.

Previously, software programs were written as monolithic, closed, integrated programs. The addition of each new feature impacted the entire program, requiring a full program test for each program code update. SOA's distributed computing approach and modular programming style avoids this by creating “building blocks” of functions or services that can be connected.

With SOA, a service such as ringing a phone, forwarding a call, or conferencing a call is developed as a building block that can easily be modified or integrated into a new application. A change to or addition of a new feature is a change to the building block of the service, unconcerned about how it might impact other functions in the entire software program. Building blocks can also be combined (“mashed up” to use an SOA term) to create new “composite services” or applications.

The major advantage of SOA is the ease with which one service can “talk” to another (by connecting the building blocks, where each block is a service) — without concern or even knowledge of the underlying interfaces or connections. So a business can rapidly link and sequence services in a process known as “orchestration” to meet new or existing business system requirements.

As businesses become more and more dependent on the Web and on instant access to communications and information, the number of business applications that need to know about the network is increasing. If communication functions can be exposed as Web Services that can be integrated to other applications this can create new and exciting applications. In fact, applications will become rich, highly valuable, and transformational for customers when they become highly collaborative and when they interact with and leverage the power of the underlying network.

The challenge is how to do this simply and rapidly.

Ease of Integration

Web Services offer one method of implementing SOA and provide a standardized way (or technology) of integrating Web-based applications using standards-based interfaces such as XML and SOAP. Using Web Services, service or application components can be published to the rest of the world. Web Services support interoperable machine-to-machine (i.e., PC) interaction or communication over a network like the Internet.

The primary protocol used in communicating Web Services is HTTP/HTTPS. SOAP, that's

CORPORATE

President and CEO

Fuat Kircaali fuat@sys-con.com

Senior VP, Editorial & Events

Jeremy Geelan jeremy@sys-con.com

ADVERTISING

Senior VP, Sales & Marketing

Carmen Gonzalez carmen@sys-con.com

Advertising Sales Director

Megan Mussa megan@sys-con.com

Associate Sales Manager

Corinna Melcon corinna@sys-con.com

Advertising Events Associate

Alison Fitzgibbons alison@sys-con.com

SYS-CON EVENTS

Event Manager

Sharmonique Shade sharmonique@sys-con.com

CUSTOMER RELATIONS

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

Richard Walter richard@sys-con.com

ACCOUNTING

Financial Analyst

Joan LaRose joan@sys-con.com

Accounts Payable

Betty White betty@sys-con.com

SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM

1-201-802-3012 or 1-888-303-5282

For subscriptions and requests for bulk orders, please send your letters to Subscription Department

Cover Price: \$6.99/issue

Domestic: \$99/yr (12 issues)

(U.S. Banks or Money Orders)

For list rental information:

Kevin Collopy: 845 731-2684, kevin.collopy@edithroman.com;

Frank Cipolla: 845 731-3832, frank.cipolla@epostdirect.com

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

the Service Oriented Architecture Protocol, is the message envelope format and can use HTTP/HTTPS, XMPP, or SMTP as its transport protocol. As a service or application component, Web Services are self-contained and self-describing. They can be discovered using UDDI (Universal Description, Discovery, and Integration), which acts like a registry and describes Web Services so developers can find them easily (to incorporate into other applications). And then there's WSDL (the Web Services Description Language), which provides Web Services interface syntax to facilitate connecting these services to one another. XML (the eXtensible Markup Language) provides a language that can be used between different platforms and programming languages and express complex messages and functions. Web Services use XML to code and decode data and SOAP to transport it using open protocols.

The key advantage of Web Services is that they use standard technologies such as XML, HTTP, SOAP, and WSDL to recognize, identify, and communicate with these building blocks of functions or services to develop new services (composite services) simply and easily.

For example, to increase the capacity of the "my conferencing service" feature now requires a simple code change in the conferencing service building block. The software developer can simply test that single building block, verify that it works, and then launch it as an overall feature of the PBX.

Before SOA, that same software programmer would search for the specific line of code (out of million of lines) that impacted that conference service, make the change, and then follow the impact of the change in other services (i.e., how it connected to other services). Finally, those million lines of codes would be tested then run in the lab for a couple of days in hopes nothing would go wrong.

What Are the Business Benefits?

Enterprises that effectively align technology with business goals achieve a competitive advantage. The adoption of Service Oriented Architecture is an effective way to organize the discrete functions contained in enterprise applications into interoperable, standards-based services that can be combined and reused quickly to meet business needs. The IT world is already using SOA and Web Services to facilitate the integration of business processes and business applications.

"Now the world of communications can participate in the integration of business applications and processes," according to Richard Tworek, general manager of SOA and next-generation unified networks at Nortel. "Using SOA frameworks and Web Services standards, vendors like Nortel have developed software that abstracts or exposes communication features from underlying communication

network infrastructures (PBXs and communication servers) and presents them as building blocks that can be integrated with other services (IT or telecom). Thus you can take real communications capabilities such as click-to-connect, IM (instant messaging), and location and presence and integrate them into other business applications and processes creating communication-enabled applications and processes."

A simple example can be communication-enabling your SCM (supply chain management) process. When inventory for a certain item falls below a certain level (say 10% of the monthly average), an IM can be sent to the purchasing officer. Furthermore, you can enact an automatic purchase order to the supplier and IM both supplier and purchasing officer that this has occurred. The benefits of this communications-enabled application or process are speed, accuracy, and agility leading to overall higher productivity and lower costs. This adaptation and interaction between the communications capabilities and the business applications is made easier by the adoption of SOA-based software architectures and Web Services technologies in the communications domain. Communications capabilities can be made available as services that can be combined with IT-based services and reused quickly to meet business needs.

By organizing enterprise IT (with telecom) around services instead of around applications, SOA provides key benefits. It

- Improves business agility, productivity, and speed (for both business and IT)
- Allows IT and telecom to deliver services faster and align closer with business
- Allows the business to respond quicker and deliver optimal user experience
- Masks the underlying technical complexity of the IT, network, and telecom environment

This results in more rapid development and more reliable delivery of new and enhanced business services.

Organizations that have adopted Service Oriented Architecture environments in their IT domains are experiencing dramatic results, including increased revenues, increased customer satisfaction, lower operational costs, and higher returns on their existing technology investments.

In summary, SOA is better because it uses a modular, distributed, building block approach to create, develop, and deliver new products and features faster, more simply, and in a less resource-intensive way. And building blocks can be mixed and matched to create new applications such as communication-enabled applications — providing yet-another benefit besides those already described. ■

About the Author

John Bednarek is responsible for Nortel's global SOA product marketing strategy, driving its Communications-Enablement Strategy across Nortel's enterprise and carrier businesses. He is a seasoned business professional with over 20 years of product marketing, business development/alliances and sales experience in the networking and software industry - delivering, marketing and selling products and services ranging from embedded systems to total system solutions for targeted markets. John earned a Bachelor of Commerce degree, with honors, from Carleton University.

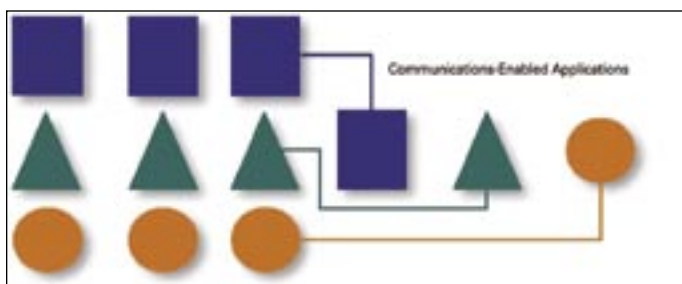


Figure 1: SOA-Enabled Service Components





_INFRASTRUCTURE LOG

_DAY 84: Feeling really disconnected. We're not getting the most out of our existing assets. Service and application integration is a nightmare. Our connections are restrictive. We've got to stop working on these islands.

_Please rescue me from this lack of connectivity.

_DAY 87: I've taken back control with IBM WebSphere solutions. Now we can service-enable and connect our existing assets for mission-critical goals. We can reuse existing applications and save money by eliminating redundant systems. Now we're ready for any SOA integration project.

_Plus, no more jellyfish stings.

WebSphere®

Download the enterprise service bus white paper at:
IBM.COM/TAKEBACKCONTROL/CONNECT

Service Reuse and Entitlement

Resolving access control tasks



BY MICHAEL POULIN

At a glance the reuse of a service and entitlement to those service results have nothing in common. But on the second thought, the more a service gets reused the greater the chance of serving users with different access rights to the service results.



We'll discuss two models of reuse and see which one is more suitable for resolving data visibility tasks.

Service Reuse Models

In the early days of SOA, service reuse was one of the main selling points in pitching the new architecture principle to business clients. It's still a mystery to me why the idea of reuse has worked this time – it's not new to IT or business. Before, object-oriented architecture promised business reuse and component-oriented architecture swore to it. Then IT finally got a great deal of reuse out of standardized API; this helped solve some business problems but failed to provide the promised scale of improvement.

In this article we'll analyze a traditional service reuse – reuse “as is.” But industry knows another kind of reuse – a container-like reuse. Some of you are probably familiar with J2EE (now JEE) containers for Web and business (EJB) applications and components. The same container can be reused for deploying multiple different components. The container enforces a certain development model

by specifying what design elements are possible and impossible to use in the components, such as multithreading.

By analogy to the container, the same service operation can be reused as a channel providing the exchange of different extended messages between service consumer and service provider. Here we'll analyze container-like reuse of the service – reuse via extension.

To complete this observation, let's explain what we mean by a service reuse. If one understands service reuse as a use of the service in multiple projects, it may or may not be service reuse. It depends on how different the projects are.

Assume we are working in financial fund management. The task is to represent historical fund prices to the consumer. To implement this task with services we need, for example, to invoke an authorization service to control access to 1) the related Web page, 2) the business service aggregating historical fund prices, 3) the fund business service, and 4) the data acquisition service to retrieve data from the persistent store. How many reuses do we have in this example? My answer is none. We just used the authorization service several times against different protected entities.

Reuse in the service eco-system means that a service gets used in

different execution contexts. For example, a mutual fund valuation service is built into the multinational financial organization. The service may be reused for funds operating in different countries. Even if countries speak the same language, like the U.S.A. and the U.K., the business execution context is different in these two places because of different laws and industry regulations.

Reuse Companion

A question about service interface granularity usually accompanies decisions of service reuse. The granularity of a service interface is a matter of art rather than a strongly regulated practice. It is a tradeoff in every case and has to take into consideration not only coding requirements but also performances, network capabilities, and service lifecycle management.

Indeed, if the interface provides fine-grained operations, like an API, it's suitable for the reuse "as is." Such reuse may reach high numbers at the beginning of a reuse process. However, try to recall the scale of effort required to move to a next version of the API, which is not totally backward-compatible... It's close to a nightmare – and the wider reuse, the worse it is.

If we use a coarse-grained service interface, it's less sensitive to the changes than the fine-grained one. However, some people believe that a coarse-grained service interface promotes the risk of massive programmatic mistakes. They think that such interfaces disallow concrete compilation control over the information passed through the "widely open" operations. I disagree with such an opinion because:

- Control based on operation granularity is inherited from the OOA, which doesn't necessarily work for function-oriented SOA services
- The granularity of operations in a SOA is usually combined with strong types of the messages – message schemas. Following SOA best practices, for a document-style service, the document schema is one of the strongest data-type controls
- The granularity of the operations and message schemas aren't a responsibility of the developer (who can make programmatic mistakes) but rather the design decision. As such, they go through multiple design reviews and controls, which minimize the chance of mistakes.

In the next sections we'll demonstrate that reuse via extension provides the needed compromise between granularity and reusability.

Service Categories and Reusability

In SOA practice and in some advanced SOA books (e.g., written by Thomas Erl), SOA services are grouped into three major categories: utility, entity, and business service. As the name suggests, utility services are helpers that provide service-oriented environment for other services and applications.

An example of a utility service is a security access control service. It is agnostic to any business processes, entities, tools, instruments, and applications; so, it can be reused in many execution contexts.

Another category is entity services. These services are tricky things – there's a debate in the SOA community over whether an entity service is a SOA anti-pattern. An entity service represents business data objects in the service world. However, each service consumer has its own "view" of the shared business data object, and each architectural layer and execution context has its own

requirements for the business data object. Obviously, we can expect less traditional reuse of the entity services than utility services.

The final category comprises business services. Such services implement business functions, features, units of work, and, occasionally business activities. Business services can work together representing an implementation of a business process. Some business processes may be embedded into the business services. Figure 1 illustrates reuse of the different service categories.

It might surprise some but in the business world we don't see much reuse of business services, i.e. the use of a service in different business contexts or situations. This is because of the business ownership model and the complexity of management across business lines. Currently, a SOA business service is seen as a reflection of the real business world. So business service reuse is less across the enterprise than the early SOA enthusiasts claimed.

Figure 2 depicts the difference in reuse between service categories depending on the reuse model. As described above, traditional reuse degrades due to inflexibility to changes in the business caused by the differences and changes in the execution contexts. On the other hand, coarse-grained service interfaces provide certain flexibility. So, we're looking for a mechanism capable of balancing specific and generic aspects of the service interface. Reuse via extension allows using the strongly typed service operations, typical for fine-grained interfaces simultaneously with flexible, extendable messages, which is closer to the coarse-grained model.

As the graph in Figure 2 shows, reuse via extension doesn't offer many benefits over traditional reuse for utility services. However, the more dynamic a business environment becomes, i.e., the more changes required to keep the business competitive in the market, the more traditional reuse loses to reuse via extension for the entity services and, especially, for the business services. Below we'll discuss one possible way of implementing reuse via extension for Web Services.

Web Services Example of Reuse via Extension

Let's consider a use-case where a SOA service with a Web Service interface values an investment fund in a hypothetical company

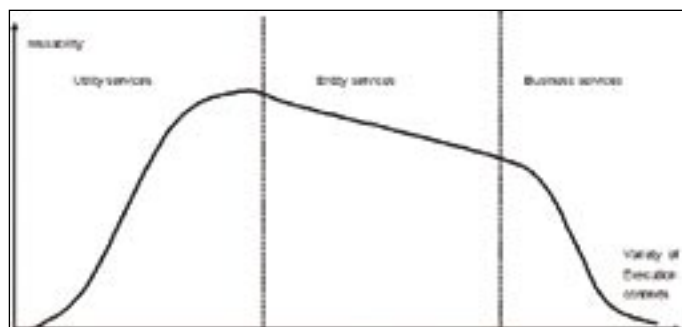


Figure 1: Traditional service reuse by categories

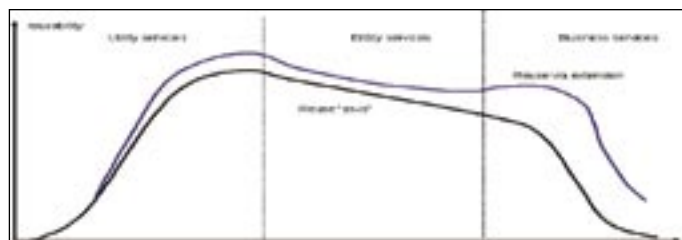


Figure 2: Service reusability by service reuse models

called Enterprise. Sometimes valuation is needed for fund processing but, in the future, it could be the final action in a business process and the valuation would have to be stored in a database.

The business client has indicated that not all requirements for the valuation are formulated yet but the valuation operation has to be available to end users as soon as possible. The development team recognizes that the valuation would be reused in multiple business scenarios over time, and some of them aren't known yet. If the number of actions associated with the valuation grows and each action to be represented as a service function, there is little chance for traditional service reuse. However, if the new service functions are defined as extensions of the same core operation, we can achieve service reusability with reuse via extension.

Service Interface Definition

The development team creates a Web Services interface for the Valuation Service (see Listings 1 and 2) and defines only one operation – `valueFund()`. (Listings 2-13 can be downloaded from the online version of this article at <http://soa.sys-con.com>). It's important to notice that the operation name reflects the valuation action for the funds, not for bonds or anything else. This is how an operation type can be specified for service reuse via extension. The operation is associated with request, response, and failure messages and with corresponding message XML Schemas.

The design goal of the interface is to loosely couple message XML Schemas with the interface operation. As Listing 2 shows, each message has its own schema and namespace defined in different files.

Basic Message Definition

With reuse via extension, a Web Service uses the same port (end point) and port type. The port type always contains the same operation with the same basic in/out/failure messages.

The basic “in” message defines the request for fund valuation. The basic “out” message defines the response, i.e., fund valuation. Since the service anticipates potential failures in service resources and connectivity, the interface defines one failure message – `serviceResourceUnavailable`.

In the Listings 3, 4 and 5 we find XML Schemas for the request message, the response and the failure message. The business action of fund valuation is represented by the element “`valueate`.” It contains three sub-elements that detail the action by specifying a fund name and fund and requester identifiers.

Basic Message Extension

The development team has decided to implement all additional business actions – new functions – in a consistent reuse via extension manner. So, the business action “store” valuation has to be implemented as an extension of the basic message XML Schema. The extension of the schema is imported as a new namespace that defines composite business action – “store.”

Let's discuss the implementation specifics shown in Listings 6 and 7. First, all elements related to the business actions are listed in the `<all>` construct and all of them are optional. This means a service consumer isn't obliged to specify any one of them if it's not needed. The basic response message reserves “`emptyRequest`” element for such case.

Second, all of the schemas have been designed to expose the namespaces in instance documents (as directed by `<elementFormDefault>` with value “qualified”). This points to the fact that all related XML documents deal with different namespaces. The service

consumer can choose whether to use a particular namespace or skip related elements. That is, the service consumer chooses to use the extension of the service or not with no harm to the existing code. Finally, any element related to a new business action can be added to the `<all>` independently of the others. For example, a service consumer can specify only the action “store” without the action “valueate” or both in any order. This is possible because the service maintains its internal state. The service is stateless as far as the consumer's invocations go (no state is saved between consecutive calls to the `valueFund()` operation) but the results of each business action are cached in the service until the response message is formed and returned. The Service Description states that all business actions in one invocation are performed for the same valuation. In the hidden service implementation, execution of any action checks if the valuation was done already and reuses it in the composed action.

As a result of such a design, a service consumer can choose to work with any available extensions when the consumer needs to and isn't forced by the service provider to update to a new version of the service. Thus the organization can schedule and gracefully manage a transition onto new service functionality for all existing consumers in accordance with service governance policies.

Further Message Extension

At the moment the service was released into production, the business client specified a new requirement: when the valuation stores, certain business consumers and other services have to be notified about this event. Also some business consumers interested in the notification may not be necessary among existing consumers of the Valuation Service.

The development team has decided to implement the new requirement by sending notification messages using message-oriented middleware (MOM). As they agreed before, any new solution has to minimally affect existing service consumers.

Following the reuse via extension concept, the request message schema has to include new business notification action represented by the new element “notify” (see Listing 8). In the definition of “notify” (see Listing 9), the “`addresseeID`” stands for an identifier of the addressee to be notified. If it's not specified (`minOccurs=“0”`), the notification is restricted, if its value is NULL, the notification may go to everybody as a broadcast, otherwise the notification has to go to a specified unlimited list of addressees as a multicast.

Obviously, the notification may fail while valuation succeeds. Thus the service interface has to include an additional failure message related to MOM or the existing failure message may be extended as shown in Listings 10 and 11. So `FundValuationException` can contain either a `serviceResourceUnavailable` or `valuationNotificationFailure` alert.

To summarize the case, we can see that the development team has implemented two new business features for the fund valuation service – “store” and “notify” – without changing the service interface. It was done by importing additional namespaces into the request message schema and changing the failure messages schema. Service consumers who need only a few features can ignore unrelated schema extensions since they're optional and don't interfere with the basic schema/namespace.

Entitlement Meets Reusability

Now assume that the Enterprise becomes a global organization and starts business units and clients in different countries.



Sound Architecture Requires Proper Planning

WEB AGE SOLUTIONS - YOUR TRAINING PARTNER FOR SOA



In all phases of SOA migration, Web Age Solutions provides training and customized services from awareness to implementation. We support vendor specific or generic SOA training tailored to your organization's needs.



Custom training for complementary SOA technologies

XML • WEB SERVICES • WSDL • SOAP • WEBSphere • WEBLOGIC • JBOSS • J2EE • SPRING/HIBERNATE/STRUTS • WID/WBI/WMB

Custom training plans for virtually every job role

BUSINESS ANALYST • ADMINISTRATOR • DEVELOPER • ARCHITECT • QA/TESTER • MANAGER • EXECUTIVE • SECURITY

Consulting services for all phases of SOA migration



Add Web Age Solutions to your plan & stay ahead of the competition

www.webagesolutions.com - 877.517.6540 - soa@webagesolutions.com



One possible work scenario comprises a business analyst in the U.S. who uses the Valuation Service to work with fund shares that belong to a citizen of Singapore. According to Singapore law private financial information can't be seen by others outside of Singapore without special agreement. So, to reuse the Valuation Service, Enterprise's compliance requires special data access control that hides some returned data in certain cases.

This is a typical data visibility or data entitlement scenario that requires an access control at the data level per service consumer and service result. Use of the service internationally (versus locally) means a change in the service execution context, i.e., in a business context for this case. Thus we may need to consider a special service response version to be used in the new execution context.

The service provider shouldn't decide when to respond with a special message version because it's outside the scope of the business service tasks. A service consumer also doesn't necessarily know upfront where the fund shareowner is, what the related laws are, and which response version of the data format to request. Instead, an Entitlement Service is supposed to be involved to specify whether returned data should be hidden and how.

To meet the new requirements, the development team has decided to build a response message XML Schema extension (Listings 12 and 13) that shows obfuscated valuation. Initially, this schema extension will be known only to the consumers working in the international area. Later, there would be a plan for a graceful transition of all service consumers onto the new response version.

The Entitlement service distinguishes between consumers working in local and international areas and informs the Valuation Service which response schema to use. The Entitlement service can also perform obfuscation of the returned value.

Thus the ability to offer different data visibility via a message schema extension allows for the dynamic, on-demand data access control in SOA services. The wider a service gets reused, the greater the chance different data visibility will be required.

Conclusion

This article compared traditional "as is" services reuse with an advanced mechanism of service reuse via extension of the message schemas. Both kinds of reuse are observed for utility, entity, and business services. Service interface granularity is also considered as a factor that contributes to service reuse capability.

Reuse via extension has a dual effect. A service vendor/provider can add new functionality by operating with extensions – optional

XML namespaces – and without affecting existing consumers. A service consumer can decide which extension to use with which functionality and when. To upgrade to new functionality, a service consumer has only to change the message parsing procedure for dealing with a new namespace. Because the parsing mechanism of the message in the document-style Web Service can be decoupled from the service's proxy code, the entire communication proxy code can be reused for newly added functionality with no changes.

Manipulating message details via different namespaces makes it possible to control data visibility inside the message. That is, different data can become visible to or hidden from different consumers based on their entitlement to information. Entitlement services can decide when and how to apply access control to the data and use different message extensions for different service consumers. This allows reusing the services in many execution contexts with a variety of customer audiences.

Resources

- Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl#_wsdl.
- A Collectively Developed Set of Schema Design Guidelines. "Multi-Schema Project: Zero, One, or Many Namespaces?" <http://www.xfront.com/ZeroOneOrManyNamespaces.html>.
- Michael Poulin. "Service Versioning for SOA." SOA-Web Services Journal. Vol. 6 Issue 7. <http://webservices.sys-con.com/read/250503.htm>.
- Michael Poulin. "Entitlement to Data." Java Developer's Journal. Vol. 11 Issue 1. <http://webservices.sys-con.com/read/163285.htm>.
- Thomas Erl. "The Principles of Service – Orientation Part 2 of 6: Service Contracts and Loose Coupling." http://searchwebser-vices.techtarget.com/tip/1,289483,sid26_gci1171966,00.html. ■

About the Author

Michael Poulin works as an enterprise-level solution architect for Fidelity International in the UK. He is a Sun Certified Architect for Java Technology, certified TOGAF Practitioner, and Licensed ZapThink SOA Architect. Michael specializes in distributed computing, SOA, and application security.

m3poulin@yahoo.com

Listing 1. Service Definition

```
<?xml version="1.0"?>
<!-- specific service bindings document -->
<definitions name="FundValuation" targetNamespace="http://enterprise.com/services/fundvaluation/service" xmlns:tns="http://enterprise.com/services/fund-valuation/service" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:defs="http://enterprise.com/services/fundvaluation/definitions" xmlns="http://schemas.xmlsoap.org/wsdl/">

  <import namespace="http://enterprise.com/services/fundvaluation/definitions"
    location="http://enterprise.com/services/fundvaluation/FundValuationInterface.wsdl"/>

  <binding name="FundValuationSoapBinding" type="defs:FundValuationPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="ValuateFund">
      <soap:operation soapAction="http://enterprise.com/services/ValuateFund"/>
    </operation>
  </binding>
</definitions>
```

```
<input>
  <soap:body use="literal"/>
</input>
<output>
  <soap:body use="literal"/>
</output>
<fault>
  <soap:fault name="FundValuationException" use="literal"/>
</fault>
</operation>
</binding>

<service name="FundValuationService">
  <documentation>mutual fund valuation</documentation>
  <port name="FundValuationPort" binding="tns:FundValuationBinding">
    <soap:address location="http://enterprise.com/services/fundvaluation"/>
  </port>
</service>
</definitions>
```




DataServices
WORLD

JUNE 24, 2008 NEW YORK CITY
www.dataservicesworld.sys-con.com



13th International

2008
SOA WORLD™
CONFERENCE & EXPO

The Number One SOA Event Comes to New York Colocated with the Virtualization Conference & Expo

June 23-24, 2008 at The Roosevelt Hotel

Service-oriented architectures (SOAs) have evolved over the past few years out of the original vision of loosely coupled web services replacing constrained, stovepiped applications throughout enterprise IT. Every major enterprise technology vendor today has developed its own SOA strategy, supported by innumerable mid-size companies and start-ups offering specific SOA aspects or entire solutions. This explosive growth in SOA technology is in response to a global demand--IDC estimates that spending on SOA services alone will grow from \$8.6 billion to more than \$33 billion by 2010.

SOA World Conference & Expo 2008 East brings together the best minds in the business to New York for a two-day conference that offers comprehensive coverage of SOA and what it means to enterprise IT today. As Zapthink analyst Jason Bloomberg has noted, "SOA involves rethinking how the business leverages IT in many various ways." Attend SOA World Conference & Expo 2008 East and learn from more than 100 speakers about how SOA is transforming business--and the way IT and business managers think

about their businesses, processes, and technology.

The colocation of SOA World Conference & Expo 2008 East and Virtualization Conference & Expo delivers the most relevant content to IT and business. Conference attendees will be able to choose from four great tracks at this year's event. Mix and match all you want, or slot into your favorite topic for the duration! The tracks include:

- Web 2.0/AJAX and SOA
- Interop Standards and Services
- Real-World SOA
- SOA Technology
- Virtualization
- Specially Selected Hot Topics

Who Should Attend?

CEOs and CTOs, senior architects, project managers, Web programmers, Web designers, technology evangelists, user interface architects, consultants, and anyone looking to stay in front of the latest Web technology.

REGISTER TODAY AND SAVE!
www.soaworld2008.com



Developing Web Services is Not the Same as Testing Them or Supporting Them

Get the help you need. The Mindreef SOAPscope Server product family provides individuals and project teams, especially those who are new to Web services, with easy-to-use and cost effective role-based products.

This gives architects, testers, developers and support engineers the advantages you need at the desktop to make sure quality, compliance and performance are happening throughout the SOA and service process.

Then, as your whole team starts to work together effectively throughout a project or across the company, you can easily migrate to SOAPscope Server for cross-team collaboration.

Trust the tools that 40 of the Fortune 100 use - and try them for yourself!

Visit www.mindreef.com/tools4me to try a no-risk evaluation,

Mindreef®
The SOA Quality Company™

Copyright Mindreef, Inc. All Photographs: Vadim Rybakov | Agency: Dreamstime.com | All Rights Reserved

Finally!

SOA Quality, Governance and Testing Tools Designed with YOU in Mind!



SOAPscope Server is the industry's leading team-based solution for testing and verifying the quality of service-oriented architectures. It encapsulates all of the features that Mindreef provides for architects, developers, testers and support engineers into a single, server-based solution that allows entire project teams to collaborate effectively while delivering well-tested, scalable and policy-compliant Web services and composite applications.



SOAPscope Architect is a powerful design-time governance and SOA quality platform for policy rules authoring, design-time support, prototyping, change-time and run-time support. SOAPscope Architect provides the ability to establish SOA design standards by combining codified industry policy sets with customized and codified organizational best practices, easily enforcing compliance throughout design and development.



SOAPscope Tester is a testing and SOA quality platform that brings powerful load testing and test automation capabilities early to the service lifecycle. It provides an integrated set of tools for Web services performance and load testing, design-time support, prototyping, change-time and run-time support. SOAPscope Tester helps QA engineers, testers and consultants to identify quality problems and potential performance bottlenecks early in the lifecycle, saving time and money while improving quality, trust and reuse.



SOAPscope Developer is a cost-effective desktop platform with integrated tools for problem diagnosis and resolution, unit testing, and supporting service consumers. SOAPscope Developer helps developers and support engineers successfully create, test, deliver and support Web services and SOAs and improve productivity by simplifying and automating tedious and time-consuming XML-oriented tasks, and improving service quality.

attend a Webinar, or download our 40-page guide to SOA testing.

www.mindreef.com/tools4me
(603) 465-2004 ext. 581

Service Oriented Infrastructure (SOI)

Why your old data center infrastructure won't scale in the SOA age

BY YOAV DEMBAK



To realize the benefits of Service Oriented Architecture, today's applications need corresponding innovations in data center infrastructure. This article examines the cornerstones of the new Service Oriented Infrastructure and explains how Service Level Automation coupled with server and network virtualization can help align infrastructure and operations with business objectives.

SOA represents the latest stage in the evolution of application architecture. The migration from mainframe to client/server to Web-enabled distributed applications all required substantial changes to underlying data center infrastructure. Without those changes, the new architectures wouldn't have satisfied end users or scaled effectively. In this respect, SOA is no different. In fact, some might say that it stresses the existing application infrastructure further than any previous development.

SOA environments are defined by loosely coupled application modules (services) that can be brought together and shared in different ways to address the dynamic needs of the business. This kind of interdependent environment, in which multiple moving parts combine together in unpredictable patterns to form business services, poses a number of production challenges.

The Butterfly Effect

Perhaps the greatest challenge to SOA is the butterfly effect. Complex dependencies between the different application modules mean that a slight change to a remote service can have a dramatic (sometimes drastic) impact on an end user or an application on the other side. SOA applications also often rely on third-party services, increasing the dependency on elements that are out of the application owner's control. Widely distributed modules, coupled with a lack of centralized ownership, only increase the complexity.

At the same time, code changes are frequent and each one has the potential to impact the underlying infrastructure in ways that are hard to foresee. In a SOA environment, ensuring consistent service delivery levels and meeting end-user expectations requires constant vigilance from a team of highly skilled experts.

The Age of Social Applications

SOA applications are dynamic not only in terms of their internal structure, but also in terms of their interaction with users and exter-

nal applications. Since multiple applications and end users can subscribe to each service dynamically, usage patterns are highly unpredictable. In a SOA environment, it's a challenge to allocate resources or govern usage and service levels effectively.

We're entering an age in which application components exhibit "social" trends, represented by organic and viral growth that tends to be both unpredictable and exponential in nature. Use of successful popular services grows exponentially while the unsuccessful ones show a linear growth pattern followed by a decline.

The Hidden Cost of SOA in a Conventional Data Center

While in their infancy, it's still possible to run SOA-based applications on existing physical infrastructure. However, the cost of scaling and manual support operations rapidly begin to grow in an exponential pattern that makes current operations models unfeasible.

Each time there's an application change – say, the code is updated or a new application subscribes to a service – and there's a demand spike from the user side, the application is vulnerable to availability or performance failures. Preventing this entails constant maintenance of a large buffer – up to 10x is appropriate given exponential growth patterns. The other option is to implement a manual procedure in real-time that modifies the infrastructure in response to every change on the application or demand side. Over time, both of these approaches will become too expensive and too resource-intensive to be practical.

To handle the flexible unpredictable nature of SOA applications, data centers need a corresponding Service Oriented Infrastructure that features adaptive programmable building blocks, application-aware networking, and service level automation.

SOI = (Virtualization)Service Level Automation

SOI should be based on real-time, adaptive, programmable building blocks that change and react to service level requirements as defined by the business. Each application module instance should run in a separate envelope with the ability to expand and shrink on-the-fly. Additionally, multiple, distributed instances of each application module should appear as one single instance to end users and other applications for scalability, high-availability and disaster recovery purposes.

Enter Virtualization

Virtualization is defined by the ability to separate the physical from the logical. For example, virtualization could make one physical server appear as multiple logical machines or make multiple physical machines appear as one logical machine.

SOI requires both kinds of virtualization; the first, to run multiple application modules on the same machine without interfering with each other's logical environments and the second, to assure the scalability and high availability of the service across multiple physical locations.

Server Virtualization

The ability to run multiple logical instances on one physical machine and allocate resources dynamically to each environment can be found in server virtualization technology (such as the ones provided by VMware and XenSource). Flexible sizing, coupled with the ability to run multiple containers in parallel on a single physical server, maximizes the utilization of infrastructure.

Network Virtualization

The second kind of virtualization, which allows for multiple distributed replicas of an application module to appear as one to the external world, can be found in network application switches. To get the most out of SOA, the application should run on an adaptive, programmable, and application-aware network that connects the user to the different services and the application modules to each other. Since the service “containers” change in size and physical location constantly (based on demand, physical constraints, and geographical cost factors), the network should be able to adapt, so that all changes are seamless to the end user as well as to additional application modules up or down the logical stream.

The network must be able to make the necessary decisions to assure that there are no interruptions to business services due to these internal or external changes. Today, this kind of adaptive network is supported by application switches (such as Citrix's NetScaler). These switches provide an API that can change policies on-the-fly as well as view and change parameters in application payloads (beyond the existing protocol fields handled by regular switches).

Service Level Automation

Service level automation is the orchestrating tier that aligns infrastructure and operations with business objectives. This tier goes beyond simple availability to ensure that each business service gets resources according to the service level goal set by the business. Ultimately, the service level goal is equal to the price – in infrastructure and operational terms – that the business is willing to pay to deliver a service.

Service level automation synthesizes the various application and infrastructure components and puts them into the context of user transactions. It can recognize and map the logical structure of a business service and maintain that knowledge as it changes at both the application and infrastructure levels. This makes it possible to manage them – and automate them – according to pre-defined policies.

While the infrastructure building blocks described above – server and network virtualization technologies – can already be found in production environments, service level automation is often missing. This creates a situation in which the infrastructure has to be



Figure 1: B-hive's dynamic business-drive automation for real-time application infrastructure

programmed on-the-fly by engineers who are probably reacting to events at the infrastructure level (server is down, CPU is 80%) or end-user complaints. This approach is reactive and resource-intensive, and can't fully leverage the capabilities of the other two SOI infrastructure tiers.

With service level automation, goals are defined in business terms rather than in infrastructure terms. The existing model, in which service levels are defined by infrastructure uptime or CPU consumption, provides little correlation with the end-user experience.

Updating the Current Operations Model

The task of keeping business services up and running to the satisfaction of the end users is the responsibility of the operations groups. Today's operations teams generally rely on the following factors:

- Human knowledge and analysis of application environments
- Manual infrastructure changes in response to hardware failures or end-user feedback/complaints
- Standalone or “silo” visibility into each OS and hardware instance in the data center.

For the reasons discussed, SOA environments are testing the limits of this approach. With frequent changes, only highly skilled engineers are capable of analyzing the application environment. Adjusting the infrastructure manually in reaction to end-user complaints or failures quickly leads to an inefficient, non-scalable operation cost structure. Finally, relying on OS and hardware-level information often no longer correlates the end-user perspective in composite environments.

CONTINUED ON PAGE 21

Do You Have a DSG (Dumb SOA Guy) Issue?

Understanding the core value of SOA

BY DAVID S. LINTHICUM

I get these about once a week: an e-mail from a Yahoo or Google e-mail account that talks about issues within a large enterprise that are related to building their first instance of SOA.

The fact is that most of these e-mails are not around proper approaches or the right enabling technology; they are around the people issues. Specifically, the emerging existence of Dumb SOA Guy(s), or what I call DSGs.

DSGs are those people (sorry ladies, I'm including you with "guys" as well) who seem to have the political power within the IT organizations, but don't have a clue as to what a SOA is, nor how to go about building one. The core issue is that the "guys" within the organization, those who understand what SOA is and how to build it, typically don't have the political skills to gain control of the projects. And, the guys who have the political skills and are the chosen ones for the new SOA work, don't have a clue as to what SOA is so they deal with it as a technology or system, and not what it is...architecture. I bet some heads are nodding as you're reading this column, right?

Again, the technology around SOA is simple, and I never worry about how we're going to leverage the technology to solve a problem. However, the people issues are more worrisome and more difficult to fix. DSGs are out there and will, I think, continue to grow as increasingly SOA becomes the "important project" with high visibility to corporate leadership. I've noticed that when that occurs, the DSGs move in quickly to control the projects.

Many DSGs attempt to oversimplify the process, rapidly moving through or even foregoing the planning steps. Their main focus is the selection of the technology, or, in some cases, they attempt to force-fit a problem with a predetermined technology solution. This can never be good.

The fact is that SOA is a complex distributed system and thus complex to plan, design, build, and test. The time spent in

planning will pay huge dividends later. There should be a very rigorous process/methodology defined that, at a minimum, provides you with a semantic-level, service-level, and a process-level understanding of the problem domain, not to mention the governance model and security strategies. Trust me, you can get SOA right the first time, but with more planning and sweat than you expect.

At issue is the fact that many people in the planning stages of SOA do not get the proper advice and guidance as to how to proceed, or even what a SOA actually is. Thus, the larger tragedy is that many of these projects will hit the wall, and do so with an impact that will reflect poorly on the notion of SOA, when it's not really a SOA issue at all.

The problem is that many DSGs don't understand the difference between JBOWS (Just a Bunch of Web Services) and a true SOA, and accept JBOWS as "experience." In reality, it's an indication that the DSGs don't understand the core value of SOA, and thus could send you off in all sorts of dangerous and costly directions. So, make sure to hire people who understand that SOA is really about configuration, agility, and changeability, and not just about service enablement. It's very easy to expose services; turning those services into solutions is another level of sophistication.

Not sure how to solve this one, other than to shine a light on the problem. Leadership within the Global 2000 will have to understand it and fix it. ■



About the Author

David S. Linthicum (Dave) is an internationally known application integration and SOA expert. In his career Dave has assisted in the formation of many of the ideas behind modern distributed computing including Enterprise Application Integration, B2B Application Integration, and SOA, approaches and technologies in wide use today. He keynotes at most major SOA and Enterprise Architecture conferences, maintains one of the most read SOA blogs, is the host of the weekly SOA Report Podcast, and is the author of 10 books, three on integration and SOA topics.

linthicum@att.net

Service level automation can control, in real-time, the virtualized application infrastructure – both network and server. It creates a new service-oriented operations lifecycle that can deliver the high-level, business-driven services goals assigned to each service and user group. At the same time, it reduces the amount of time engineers need to spend on configuring and sizing infrastructure 24/7, while addressing the requirements of each application module.

Closing the Communication Gap

A service-oriented operations lifecycle starts at the top-down – with the ability to monitor, analyze, and report service level measurements and goals in business terms and react with real-time infrastructure automation to achieve those goals.

Application owners are concerned about performance indicators such as end-user latency and transaction error rate rather than silo-level performance indicators such as network uptime or service CPU utilization. This often leads to communication problems, with the infrastructure and operations teams on one side, doing their best to monitor and react to low-level infrastructure events, and the application business owner on the other side, reacting to negative sentiments from end users suffering from unpredictable service levels. Service level automation can bridge this gap.

The Service-Oriented Operations Lifecycle

Operations in the SOA age are a constant cycle, optimizing infrastructure to achieve service level goals for different business services that change on-the-fly.

The first step in a service-oriented operations lifecycle is therefore the ability to understand the status of the service from the end user's perspective (transaction latency and health) and then analyze how the different application and infrastructure components impact that end-user service level. To understand the complex dependencies and adapt to the constant changes, this process must be automatic and self-learning – it can't rely on knowledge of the inner structure of the service, which often resides only with specific individuals inside the application groups.

The second stage of the automated operations lifecycle displays the discovered services as a map of business functions as conducted by actual users. At this point, the business owners can define the service-level goals that they find acceptable and are willing to finance in terms of infrastructure.

The third stage in the lifecycle automates virtualized application infrastructure to close the gap, in real-time, between current service delivery levels and the pre-defined, business-driven goals. At this point, the service level automation tier uses the API available on the programmable application infrastructure to do things such as:

- Add 15% of a CPU to a virtual server that's running an oversubscribed application module that's taking three seconds to complete instead of the desired 1.5 seconds.
- Shut down a virtual server running a database that's failing on 20% of its authentication queries, while bringing it up on a different physical machine and redirecting all service calls transparently to the new machine.
- Prioritize high-end trader transactions over third-party API calls across six application module pools distributed across three continents to ensure that the "rogue" API calls don't consume over 5% of the resources required to complete mission-critical and

revenue-generating end-user transactions.

Bottom Line: The Benefits of SOI

A real-time adaptive SOI managed according to business-driven, service-level goals can actually improve user satisfaction while saving the organization money.

Aligning the Business & IT

SOI takes SOA beyond just the ability to release new features faster and adds the ability to ensure that services continue to function – all the time – at the levels required by the business.

- **Controlling Opex** – Without an adaptive SOI, the cost of reactive manual operations doesn't scale. The time will come when the usage model or the complexity level of a service will drive its price up beyond its associated revenue.
- **Reducing Capex** – An adaptive SOI can maximize resource utilization up to 10x while ensuring that service levels are always met. It eliminates the need to overprovision infrastructure buffers while providing the ability to share resources dynamically and safely across multiple physical and geographical locations.

About the Author

Yoav Dembak is CEO and co-founder of B-hive Networks, Inc. He has spent many years working with distributed composite applications in some of the largest data centers in the world. Prior to founding B-hive, Yoav held senior positions in marketing and technical sales at Breach Security, where he led the design and launch of BreachGate, a product family of Web application Firewall solutions for the enterprise market. Prior to that, he held various technical sales and marketing positions with Gilan Technologies, a leading provider of Web content integrity solutions for enterprise Web sites acquired by Breach Security. Yoav began his career at Gilat Satellite Communications. He served three years in the Israel Defense Forces in a special counter-terrorism unit and holds a bachelors degree in computer science and business management from Tel Aviv University. To learn more about B-hive, please visit www.bhive.net.

Advertiser Index

ADVERTISER	URL	PHONE	PG
Altova	www.altova.com		35
Attachmate	www.attachmate.com/soa		5
SOA World Conf& Expo '08	http://soaworld2008.com/		15
JavaOne 2008	http://java.sun.com/javaone		22,23
DataServices World	www.dataservicesworld.sys-con.com		36
B-hive	www.bhive.net/download	650-358-1300	2
IBM	www.ibm.com/takebackcontrol/connect		8,9
Web Age Solutions	www.webagesolutions.com	877-517-6540	13
Mindreef	http://mindreef.com/	603-465-2004 x581	16,17

General Conditions: The Publisher reserves the right to refuse any advertising not meeting the standards that are set to protect the high editorial quality of *Net Developer's Journal*. All advertising is subject to approval by the Publisher. The Publisher assumes no liability for any costs or damages incurred if for any reason the Publisher fails to publish an advertisement. In no event shall the Publisher be liable for any costs or damages in excess of the cost of the advertisement as a result of a mistake in the advertisement or for any other reason. The Advertiser is fully responsible for all financial liability and terms of the contract executed by the agents or agencies who are acting on behalf of the Advertiser. Conditions set in this document (except the rates) are subject to change by the Publisher without notice. No conditions other than those set forth in this "General Conditions Document" shall be binding upon the Publisher. Advertisers (and their agencies) are fully responsible for the content of their advertisements printed in *Net Developer's Journal*. Advertisements are to be printed at the discretion of the Publisher. This discretion includes the positioning of the advertisement, except for "preferred positions" described in the rate table. Cancellations and changes to advertisements must be made in writing before the closing date. "Publisher" in this "General Conditions Document" refers to SYS-CON Publications, Inc. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.



MORE OF WHAT YOU NEED

200+ technical sessions

More than **100**
Birds-of-a-Feather sessions

15 Hands-on Labs

- Web 2.0
- Rich Internet applications
- Compatibility and interoperability
- Open source
- E-commerce collaboration
- Scripting languages
- Java Platform, Standard Edition (Java SE)
- Java Platform, Enterprise Edition (Java EE)
- Java Platform, Micro Edition (Java ME)

Platinum Cosponsors



Cosponsors





JavaOneSM

Sun's 2008 Worldwide Java Developer Conference™

EVERYTHING

ABOUT JAVA™ TECHNOLOGY. AND SO MUCH MORE.

You won't want to miss the JavaOne conference, the premier technology conference for the developer community. This year's Conference presents the latest and most important topics and innovations today to help developers access even more richness and functionality for creating powerful new applications and services.

**Save
\$200**

Register by April 7
at java.sun.com/javaone

Please use priority code: J8PA5SC



JavaOneSM Conference | May 6–9, 2008

JavaOneSM Pavilion: May 6–8, 2008, The Moscone Center, San Francisco, CA





SOAP Over JMS Interoperability

Exposing a Java Web Service via JMS using Apache Axis 1.4 and consuming it from both Java and .NET clients

BY STANIMIR STANEV AND ROB BARTLETT

Web Services are becoming the chosen way of exposing interoperable units of work as services. Today consumers and providers of software services talk different languages, and SOAP makes them understand each other.

SOAP can be transported via almost anything, and we sometimes joke that we can even do SOAP over FedEx if necessary. Some architects lay down their design based on SOAP over HTTP services, but as the project grows, they face challenging scalability problems. One of the most accepted ways of addressing these problems is to use JMS. We usually provide HTTP wrappers – still exposing our services as SOAP over HTTP, but those are lightweight fronts that just send or publish a message to a queue or a topic to trigger heavy business logic.

We build layers and layers, and as they help isolate different components, they impact performance and require additional managing. We can imagine a world of services after years of development, where to reach the very back end of business logic, a message flies through so many hubs that becomes impossible to attain reasonable SLAs anymore.

Why don't we eliminate the extra hubs when we can? How about if we reach the services directly through JMS? True, this may not be very appropriate for services exposed on the Internet, but it's probably more likely and preferable to do it in an organization where architectures like EDA (Event Driven Architecture) are implemented to help the production, detection, consumption of, and reaction to events.

We're going to show how to expose a Java Web Service via JMS using

Apache Axis 1.4 and consume it from both Java and .NET clients by re-using the generated stubs and proxies and just changing the transport.

Java Service Consumer + MOM + Java Service Provider

As shown in the diagram, our service consumer side consists of components that are both written by us (represented by the blue boxes) and those that are provided by Axis (represented by the white boxes). The picture on the service provider side is similar. In the middle, we have MOM (Message-Oriented Middleware), which increases the interoperability, portability, and flexibility of an application by letting it be distributed over multiple heterogeneous platforms. MOM reduces the complexity of developing applications that span multiple operating systems and network protocols by insulating developers from the details of the various operating systems and network interfaces.

Sample Service

Our sample Shipping Service has a GetDistance operation that accepts two ZIP codes and returns the distance between the two locations in miles.

To understand the roles and functionality of each component better, let's go through them.

Service Consumer Side

Consumer

The consumer is code we write. It uses generated-by-WSDL2Java stubs and proxies to instantiate the service request, populate it with request values, and provide it to the Axis client engine for transportation to the service endpoint.

The default transportation mechanism that Axis uses is the commonly accepted HTTP one and obviously it works with service endpoints that are specified by their http locations like HYPERLINK “http://momentumsi.com/service/shipping”.

In our case, we want Axis to use JMS as a transport of our payload. We’re looking for a way to reuse the generated-by-WSDL2Java code and just instruct Axis to deliver the payload via JMS instead of HTTP.

Fortunately, Axis provides a way to register different transports that should be used for the service endpoint URLs of the specified protocol. For example, we can register the provided-by-Axis org.apache.axis.transport.jms.JMSTransport to serve endpoint URLs that are specified by jms:// protocol like this:

```
import org.apache.axis.client.Call;
import org.apache.axis.transport.jms.JMSTransport;
...
Call.setTransportForProtocol("jms", JMSTransport.class);
```

The JMSTransport is responsible for parsing the transport-specific properties provided in the endpoint URL and setting them up in the message context. Later we’ll explain how JMS-Sender handler takes those properties and does the actual JMS work.

Generated Proxy

The Generated Proxy is generated by the provided-by-Axis WSDL2Java utility. This component is used by the consumers to do both SOAP-over-HTTP and JMS calls. Even better, the component can be reused from any other registered transport implementation that we can come up in our architecture.

Axis Client Engine

The Axis Client Engine is provided by Axis. It can be configured with WSDO files that are written by us and provided to the engine via the generated proxy.

What is a WSDO file? A Web Service is deployed into Axis using an XML-based deployment descriptor file known as a Web Service Deployment Descriptor (WSDO). WSDO describes how the various components installed in Axis are to be chained together to process incoming and outgoing messages to the service. The WSDO can be used both on the client and server sides.

For our shipping service, here’s an example of how to provide the client WSDO to the Axis client engine and then just invoke the GetDistance service operation:

```
import org.apache.axis.configuration.FileProvider;
...
EngineConfiguration engineConfig = new FileProvider("client-config.wsdd");
ShippingServiceLocator shippingServiceLocator = new ShippingServiceLocator(
    engineConfig);
ShippingSoapBindingStub shippingService = (ShippingSoapBindingStub)shippingServiceLocator
```

```
Locator.getShipping();
...
GetDistanceResponse response = shippingService.getDistance(request);
```

Notice that in this example the name and location of client-config.wsdd is hard-coded in the source code, but for production, you should change it so it’s externalized and provided as a property, injected via the Spring framework or other mechanism.

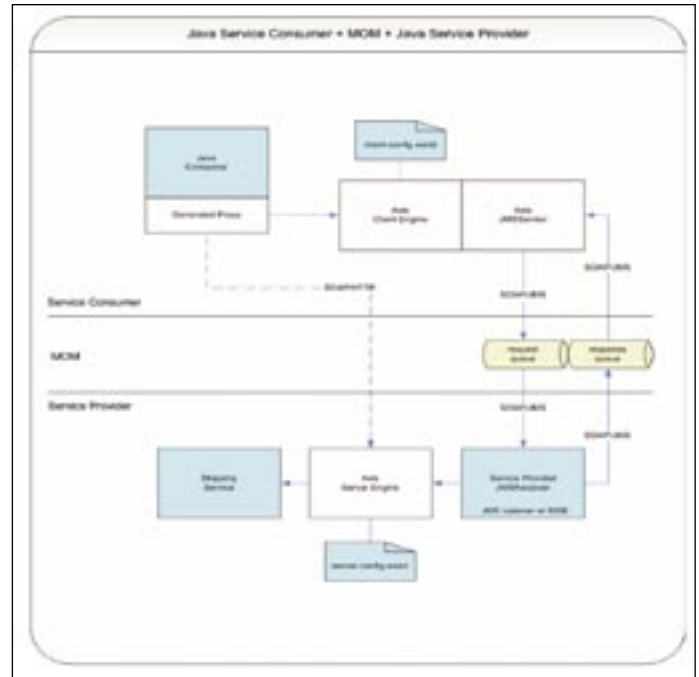


Figure 1: Traditional service reuse by categories

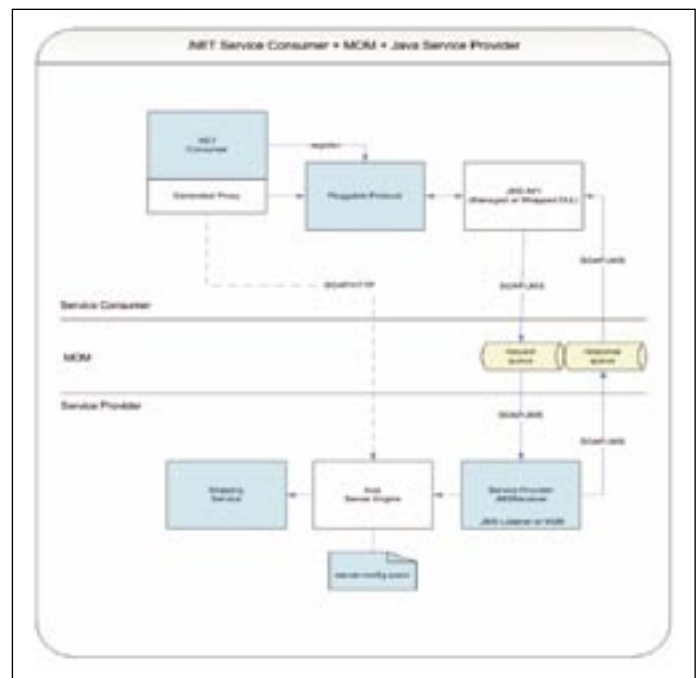


Figure 2: .NET Service Consumer + MOM + Java Service Provider

client-config.wsdd

The client-config.wsdd is used by the Axis client engine to specify the handler responsible for sending the message to the ultimate receiver. The following deployment descriptor replaces the default HTTPSender handler with the JMSSender handler that uses JMS to transport SOAP messages.

```
<?xml version="1.0" encoding="UTF-8"?>
<deployment
xmlns=HYPERLINK "http://xml.apache.org/axis/wsdd/"http://xml.apache.org/axis/wsdd/
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
<handler name="JMSSender"
type="java:org.apache.axis.transport.jms.JMSSender" />
<transport name="JMSTransport" pivot="JMSSender"/>
</deployment>
```

URL-based JMS Transport

The JMS URL syntax provides a vendor-neutral way of specifying JMS specific details like Topic and Queue destinations.

The query part of the service provider endpoint URL (the part after the question mark) is treated by Axis as key/value pairs and they are provided to the JMS vendor adapter factory to instantiate a specific adapter. The default JMS adapter in Apache Axis is JNDI, so if the vendor=JNDI isn't specified in the URL then the default org.apache.axis.components.jms.JNDIVendorAdapter will be used to locate the JMS connection factory and destinations. You can write your own adapter, just make sure it's available in the CLASSPATH and specified by the vendor property like vendor=com.momentumsi.jms.adapters.MyOwnAdapter.

If your JMS provider happens to be Tibco's EMS then a sample URL may look like this:

```
jms://tibjms/queue?java.naming.provider.url=tcp://localhost:7222&java.naming.
factory.initial=com.tibco.tijms.naming.TibjmsInitialContextFactory&ConnectionFact
oryJNDIName=QueueConnectionFactory&Destination=queue.test
```

Before making the actual call, the Consumer must change the service provider endpoint to the JMS one as shown in the example below:

```
...
String jmsUrl = "jms://...";
shippingService._setProperty(javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY,
jmsUrl);
...
GetDistanceResponse response = shippingService.getDistance(request);
```

Axis JMS Sender

The JMSSender handler is provided by Axis. It takes the properties from the message context that were parsed and prepared by the JMSTransport and does the actual work:

- Create JMS connection
- Create temporary JMS response destination
- Send the location of the temporary JMS destination together with the actual SOAP request to the specified JMS request destination

- Receive the SOAP response from the temporary JMS response destination
- Provide the SOAP response to the Axis Client Engine to be delivered to the consumer

Service Consumer Summary

We were able to reuse the unchanged code that was generated by WSDL2Java. We registered a transport implementation that handles jms:// URLs and we changed the endpoint URL to jms:// where we specified JNDI properties like JMS connection factory and destination. We provided a custom client-config.wsdd to the Axis Client Engine, where we specified an Axis-provided JMS handler that does the actual send and receive of SOAP requests and responses.

MOM (Message-Oriented Middleware)

As described above, the MOM layer increases interoperability, portability, and flexibility. It has no knowledge of the consumers or message provider and just does what it does best: serve as a backbone for an Event Driven Architecture.

MOM could be any available JMS provider like ActiveMQ, JBoss Messaging, Open JMS, Oracle AQ, Tibco EMS, WebSphere MQ, or for advanced processing, any available ESB provider like Apache ServiceMix, Tibco BusinessWorks, Sonic ESB, or OpenESB.

Service Provider Side

Service Provider JMSReceiver

Although Axis provides a simple org.apache.axis.transport.jms.SimpleJMSListener receiver, it's not intended for production. It's for testing and debugging purposes. So we have to write our own.

Our ShippingJMSReceiver is a standard JMS listener that extends javax.jms.MessageListener. Its basic purpose is to listen asynchronously for messages and then pass them off to our MessageProcessor. It can be deployed as a MDB (Message Driven Bean) to any application server, or even configured via Jencks to deploy inside Spring and provide a Message-Driven POJO.

```
public class ShippingJMSReceiver implements MessageListener
{
    public void onMessage(javax.jms.Message message)
    {
        BytesMessage bytesMessage = (BytesMessage) message;
        MessageProcessor msgProcessor = new MessageProcessor();
        msgProcessor.processMessageAsSoapRequest(getAxisEngine(), bytesMessage);
    }

    private AxisEngine getAxisEngine() throws Exception
    {
        FileProvider fileProvider = new FileProvider("deploy-shipping.wsdd");
        return new AxisServer(fileProvider);
    }
}
```

After the message is received, in the code above, the logic instantiates the standard AxisEngine, passes the deploy-shipping.wsdd to it, and invokes the MessageProcessor. Again, the name and location of the hard-coded deploy-shipping.wsdd file should be externalized to allow changes without code recompilation and redeployment.

The `MessageProcessor` works with a single SOAP request. It uses the `AxisEngine` to invoke the appropriate service operation. The beauty of this approach is that it reuses the `AxisEngine` that handles the SOAP messages without knowing how they were delivered. It's transport-independent.

```
...
Message soapMessage = new Message(bytesMessageReader);
MessageContext msgContext = new MessageContext(axisEngine);
msgContext.setRequestMessage(soapMessage);
axisEngine.invoke(msgContext);
soapMessage = msgContext.getResponseMessage();
...
```

The SOAP response is sent back to the JMS destination provided by the `JMSSender` from the consumer side.

```
...
Destination responseDestination = bytesMessage.getJMSReplyTo();
if (responseDestination != null)
{
    ByteArrayOutputStream out = new ByteArrayOutputStream();
    soapMessage.writeTo(out);
    JMSUtil.sendJmsMessage(responseDestination, out.toByteArray());
}
...
```

For simplicity's sake, we don't show any reasonable exception handlers or provide the code available in `JMSUtil` that contains the standard Java to-send JMS message to the specified destination.

Axis Server Engine

The `Axis Server Engine` is provided by `Axis`. As we mentioned, it has no knowledge of how the SOAP message was delivered. It just takes it and processes it, which will actually result in an invocation of the actual `GetDistance` operation of our `ShippingService`.

So how does `AxisEngine` find the `ShippingService`? Well, the `deploy-shipping.wsdd` provided to the `AxisEngine` is a standard `Axis` deployment descriptor. It describes handlers, services, operations, mappings, classes, and everything that `Axis` needs to find and invoke the operation as a regular Java method.

Shipping Service

We write the shipping service. It contains the actual business logic of the exposed service operations.

```
public class ShippingWebService
{
    public GetDistanceResponse getDistance(GetDistanceRequest request)
    throws Exception
    {
        ...
    }
}
```

As a regular `Axis` Web Service, the shipping Web Service requires the `deploy-shipping.wsdd` that we use to deploy the service under

`Axis`. To learn more about how to write `WSDD` files, see the `Apache Axis` documentation.

Service Provider Summary

The only component that we've added to handle SOAP messages is the `JMSReceiver`. It's plugged into the architecture to handle messages delivered through JMS. The rest of the components are exactly the same since they expose standard SOAP over HTTP services. That's why this architecture can handle requests delivered by both JMS and HTTP at the same time. Besides these two, by implementing appropriate receivers, the service provider can be extended to handle messages delivered by any transport.

The .NET Consumer of JMS

SOAP over HTTP is easy to consume from a .NET application. Visual Studio 2005 or `WSDL.exe` will automatically generate proxies based on a `wsdl`, so developers can get down to the business logic rather than worry about the plumbing. That's not to say it's as easy as pie. Not all SOAP is created equal, but addressing the issues around SOAP compatibilities would warrant an entire article (or several). Here we're focused on how to use the .NET infrastructure to consume Java SOAP services over JMS. Our examples were created for .NET 2.0 using Visual Studio 2005 and they assume a general understanding of how to generate HTTP Web Service proxies.

.NET SOAP over JMS

Boiled down, this JMS solution hijacks the SOAP request before it goes over the wire, sends it over JMS, listens for a response via JMS then returns the SOAP response back to the caller, allowing it to finish processing. We only pass the SOAP through the system – we don't have to do anything special to generate it, process it, serialize it, or deserialize it. We let .NET do the heavy lifting that it was going to do anyway.

Normally, a Web Service will use an HTTP URI. When the Web Service proxy makes http calls, it causes the `WebRequest.Create()` method to produce an instance of `HttpWebRequest`. The proxy generates the SOAP, hands it off to the `HttpWebRequest`, which sends it over the wire, gets a response, and then sends that to the `HttpWebResponse`. Finally, the proxy takes over again.

We're going to take advantage of the "pluggable protocol" feature in .NET to make using JMS almost transparent. We say "almost" because we don't want to supersede HTTP in all cases – just for certain services. First, we'll need to know what flavor of JMS we're using. For this article, we're focusing on `ActiveMQ` because it's freely available and already has a pure .NET API. The .NET API we use is from `Spring.NET`. There are other options, such as `OpenMQ` and `Tibco`. Even if there were no .NET APIs already, we could wrap a `dll`. If the API existed only in Java, there are technologies such as `JNBridge` that can bridge technologies.

Once the JMS API is selected, we need to create several components: an `ActiveMqWebResponse`, an `ActiveMqWebRequest`, an `ActiveMqWebRequestCreate`, and an `ActiveMqSoapStream` – a specialized stream for hijacking the SOAP. These classes are custom versions of the components used in the normal flow of HTTP Web Services. Then, of course, we need a consumer.

ActiveMqSoapStream

The specialized stream is where we do the fancy footwork to

hijack the SOAP. We don't want the proxy to realize it's dealing with a special stream. This class inherits `System.IO.Stream` and is mostly a pass-through to an encapsulated stream. The primary difference is that it overrides the `Close()` method called by the base `WebRequest`. Instead of closing the stream, this method rewinds the inner stream so our hijacking code can process the stream from the beginning. It also has an internal close method that our `ActiveMqWebRequest` will call to truly close the underlying stream when it's done with it, otherwise the stream would stay open indefinitely.

```
public class ActiveMqSoapStream : Stream
{
    private Stream m_Stream;
    public override void Close()
    { m_Stream.Position = 0; }
    internal void InternalClose()
    { if (this.CanSeek == true) m_Stream.Close(); }
}
```

ActiveMqWebRequest

The `ActiveMqWebRequest` is where the bulk of the work happens. It inherits from `System.NET.WebRequest` and implements the abstract methods and properties. There are a few custom properties for JMS-specific information such as the address, username, password, and queue name. We also have a field of type `ActiveMqSoapStream`.

```
public class ActiveMqQueueWebRequest : WebRequest
{
    protected ActiveMqSoapStream m_RequestStream;
    private string _password;
    private string _username;
    private string _queueAddress;
    private string _queueName;
    ...
}
```

For brevity's sake, I won't go into the details of the property accessors or pass-through methods. The methods we're most interested in are `GetRequestStream()` and `GetResponse()`. `GetRequestStream()` is where we replace the default stream with our own.

```
public override Stream GetRequestStream()
{
    m_RequestStream = new ActiveMqSoapStream(new MemoryStream(),
    true, true, true);
    return m_RequestStream;
}
```

`GetResponse()` is where we send the request, listen for a response, and then put the response in a return stream. This is the workhorse of the class. The first thing we do is access the SOAP stream as an array of bytes.

```
public override WebResponse GetResponse()
{
    byte[] responseBody = new Byte[m_RequestStream.Length];
```

```
m_RequestStream.Read(responseBody, 0, responseBody.Length);
```

Next we create the connections we're going to be using. We create a temporary queue as a response destination. This lets us have a request/response model instead of an asynchronous model. If we wanted an asynchronous model, we could listen to a permanent queue and then use the correlation ID to match the response to our request.

```
ConnectionFactory connectionFactory = new ConnectionFactory(
_queueAddress);
try
{ using (IConnection connection = connectionFactory.CreateConnection())
  { using (ISession session = connection.CreateSession())
    {
        //Create a temporary queue so we can listen for the response
        ITemporaryQueue queue = session.CreateTemporaryQueue();
        using (IMessageConsumer consumer = session.CreateConsumer(queue))
        {
```

Next, we attach the SOAP as a byte array to a message and send it to the queue/topic.

```
IBytesMessage message = session.CreateBytesMessage();
message.Content = responseBody;
//Give the address of our temporary queue
// so the service knows where to send the
response
message.NMSReplyTo = queue;
using (IMessageProducer producer = session.CreateProducer())
{ NmsDestinationAccessor destinationResolver =
new NmsDestinationAccessor();
    IDestination destination =
    destinationResolver.ResolveDestinationName(
session, _queueName);
    producer.Send(destination, message);
}
```

Now we wait for a response. We set a timeout (as a `TimeSpan`). Here it's a fixed value of 10 seconds for illustration, but it could easily come from a config file or passed as a property in the `ActiveMqWebRequest`-Create.

```
IBytesMessage response = (IBytesMessage)
consumer.Receive(
TimeSpan.FromSeconds(10))
as IBytesMessage;
stream = new MemoryStream();
stream.Write(response.Content, 0, response.Content.Length);
```

```
} } }
catch (NMSException e)
{ throw (e); }
finally
{ m_RequestStream.InternalClose(); }
```

```
ActiveMqQueueWebResponse resp = new ActiveMqQueueWebResponse();
```



```

Here we set the stream to our own.

resp.SetDownloadStream(stream);
return resp;
}

```

Our `ActiveMqWebResponse` class inherits `System.Net.WebResponse` and does little more than provide a setter and override the getter on the response stream, since the stream in the built-in `WebResponse` is read-only.

```

public class ActiveMqQueueWebResponse : WebResponse
{
    private Stream m_ResponseStream;
    private long m_lngContentLength;
    ...
    internal void SetDownloadStream(Stream vobjResponseStream)
    { m_ResponseStream = vobjResponseStream;
      m_ResponseStream.Position = 0; // Rewind the stream
      m_lngContentLength = m_ResponseStream.Length;
    }

    public override Stream GetResponseStream()
    { return m_ResponseStream;
    }
}

```

ActiveMqWebRequestCreate

`ActiveMqWebRequestCreate` is a class that is used to create instances of our objects instead of the built-in ones for a particular URI prefix. We'll see how this is used in the client application.

```

public class ActiveMqQueueWebRequestCreate : IWebRequestCreate
{
    ...
    public WebRequest Create(System.Uri uri)
    { ActiveMqQueueWebRequest request =
new ActiveMqQueueWebRequest(
uri, _queueAddress, _queueName,
_username, _password);
      return request;
    }
}

```

The .NET Client

In the client, we generate web service proxies exactly the same as if they were HTTP, only we don't necessarily have the option to hit a url that generates/serves a wsdl, so we use a file wsdl. We don't have to do anything special with the proxies. In this example, `ShippingService` is the service, and `GetDistance()` is its only operation. It accepts a `GetDistanceRequest` and returns a `GetDistanceResponse`. Before we call an operation on our service, we register our protocol to use our `ActiveMqWebRequestCreate` for a custom URI prefix.

Once this protocol is registered, it's available from that point forward within the application domain.

```

// Create a dummy URI
System.Uri urlRequest = new System.Uri(
"activeJms://ftp.momentum.com/activeJmsUri/I/do/not/exist.txt");
// Register it
bool isGood = WebRequest.RegisterPrefix("activeJms",
    new ActiveMqQueueWebRequest.ActiveMqQueueWebRequestCreate(
"tcp://localhost:61616", "myQueue", "un", "pw"));

```

We call our services the same way we would if it were HTTP. The only difference is that the URI uses a different prefix.

```

ShippingService.ShippingService svc =
new ShippingService.ShippingService();
svc.Url = "activeJms://localhost:61616";
ShippingService.GetDistanceRequest req =
new ShippingService.GetDistanceRequest();
req.startZipCode = "78728"; req.endZipCode = "50158";

ShippingService.GetDistanceResponse resp = svc.GetDistance(req);

```

In this specific implementation, each prefix has a fixed address and queue. Fancier implementations could support extra information in the URI query string similar to the Tibco one mentioned in the URL-based JMS transport section.

.NET Consumer Summary

The components above can be created once, and then leveraged for any number of services. Once the protocol is registered, we don't have to change the way we work with services, and for all the talk of SOAP, we didn't have to see any! There's nothing to prevent us from using both the JMS and HTTP web service implementations in the same application or integration tests.

Conclusion

We made our service consumers talk directly to the MOM layer. No extra hubs are involved, so the SOAP requests reach the middle-ware as quickly as possible. With appropriate JMS adapters and knowledge, you can easily implement a consumer in almost any language. Most important, we kept untouched the generated stubs and proxies at the consumer and reused the service engine on the server side. So in addition to the highly scalable and reliable, this approach is also fast and easy to implement. ■


About the Authors

Stanimir Stanev is a senior consultant at MomentumSI's Enterprise Architecture Solutions practice. He has many years of experience focusing on providing enterprise architecture and strategy expertise to companies looking to migrate to or maximize the advantages of SOA principles.

sstanev@momentumsi.com

Rob Bartlett is a senior consultant at MomentumSI's Software Development Solutions practice. He has over a decade of experience in technical roles, guiding major corporations in the design, implementation, and integration of business solutions.

rbartlett@momentumsi.com



Next-Generation Service Infrastructure & the Semantic Challenge

Computer science on the edge of a new generation

BY JEAN-PIERRE LORRÉ

Driven by SaaS market momentum, the growth of large service ecosystems involves radical changes in both enterprise Business Process organization and IT infrastructure to fit interoperability and agility requirements.

Many questions associated with the paradigm shift arise: how can we move to a network of services of Internet scale transparently available across the many devices we use to access information? What support will services need from an increasingly ubiquitous Internet? To meet these requirements, we propose to address the service infrastructure itself and provide an open, context-aware service bus, enabling both syntactic and semantic interoperability.

Currently, Service Oriented Architecture (SOA) technologies are being used successfully to solve the syntactic interoperability problem whereas the composition (e.g., orchestration) engines provide increased agility. Nevertheless, such classical SOA technologies and Enterprise Service Bus (ESB) concept technologies as those provided by editors or open source providers don't efficiently support large

collaborative business networks, since the service description is still limited (and doesn't provide semantic interoperability) and since the global context (i.e., security, reliability, etc.) isn't taken into account while composing services. Moreover nowadays Web Service standards provide a communications medium for distributed systems, but they can't yet ensure that the communicating parties "speak the same language," which is necessary for smooth, fully automated systems interoperation.

In general, developing large service ecosystems involves taking into account nomadic properties and quality of service requirements, particularly considering that these extended environments may use a large variety of communications infrastructures. Such non-functional requirements can be taken into account in a context-aware service composition process whereas probes can capture the execution context.

The main drivers for next-generation service infrastructure (see Figure 1) are then to:

- Enable a service to be used anywhere from any kind of device (pervasive technologies). These services and related information are stored in a lightweight distributed repository and deployed into a new generation Enterprise Service Bus (ESB) that encompass enterprises borders.

- Address end-to-end Quality of Service (QoS) requirements associated with service provisioning, with a special emphasis on dependability constraints and more specifically scalability, reliability, and security.
- Provide an agile framework supporting business-level Service Level Agreements (SLA) definition and monitor system behavior accordingly to enforce required non-functional properties of potentially composite service execution.

The aim of this article is to describe the components of this framework and show how we take into account the full set of challenges encountered by such large service ecosystems by extending classical ESB and SOA technologies thanks to semantic properties.

The first part is a short presentation of basic semantic service and service infrastructure technologies, followed by an explanation of how to use semantics to improve service governance, SLA definition, and relative enforcement mechanisms. A reactive composition framework is then offered to emphasize the main SOA advances for next-generation service infrastructure.

Semantic Services

In the Semantic Web paradigm, information on the Web is enriched with machine-interpretable semantics to allow its automated manipulation. An entity's semantics encapsulates the meaning of the entity by reference to a structured vocabulary of terms (ontology) representing a specific area of knowledge. Ontology languages support formal description and machine reasoning on ontologies; the Web Ontology Language (OWL)¹ was recently standardized by W3C.

Semantic Web Services employ such Semantic Web technology in the Web Services area: service functionality, Web Service inputs and outputs, their preconditions, effect, performance, and other QoS aspects; all are expressed in knowledge representation languages, referring to shared ontological vocabularies. This effort aims at the semantic description of Web Services towards automating Web Services discovery, invocation, composition, and execution monitoring:

- When searching for a service providing a specific functionality, ontology can provide synonyms of words, the taxonomic structure of service capabilities, relationships between service capabilities, etc.
- When trying to harmonize different data formats for two services that have to exchange messages, ontologies can provide elaborated conceptual data models for message descriptions that facilitate automated translation.
- When trying to compose complex business processes from given partial processes implemented by a number of services, automated planning algorithms from artificial intelligence can be employed, provided the semantics of the input services are formally defined. Such semantics can embody QoS attributes that allow a planning algorithm to take them into account.

Hence a number of research efforts have been proposed for the semantic specification of Web Services. The latest WSDL 2.0 standard not only supports the use of XML Schema, but also provides standard extensibility features for using, say, classes from OWL ontologies to define Web Service input and output data types. Actually, SAWSDL² is W3C's new candidate recommendation here, defining how semantic

annotations can be added to WSDL descriptions. Further, employing OWL, OWL-S³ is a high-level OWL ontology for Web Services. A similar recent proposal is the Web Services Modeling Ontology (WSMO)⁴ that's specified using the Web Service Modeling Language (WSML).

Semantic technologies can be employed further for describing non-functional service properties, such as service QoS, including dependability and security properties, as well as service context. This field is far from being standardized, which raises a number of issues, such as the selection of adequate non-functional properties and their effective description to allow discovery and potential composition of services that will satisfy the required properties.

Key considerations for a semantic service architecture are then:

- Semantic description of services in terms of both functional and non-functional properties; the latter include QoS as well as system and user context;
- Support for QoS specification and enforcement through the use of SLA;
- Service interoperability based on semantics;
- Semantic service registries publishing service descriptions;
- Service selection based on matching between functional and non-functional requirements of a business process and corresponding service properties;
- Support for complex services through service composition and adaptation taking into account both service semantics and behaviour;
- Ensuring QoS, dependability and security properties for service aggregations at the composite service level.

Highly Distributed Service Infrastructure

According to Gartner⁵, "An Enterprise Service Bus (ESB) is a new architecture that exploits Web Services, messaging middleware, intelligent routing, and transformation. ESBs act as a lightweight, ubiquitous integration backbone through which software services and application components flow."

Originally, the first commercial ESB products were mainly described as a way to integrate existing middleware services (J2EE application servers, message-oriented brokers, etc.) and products (such as B2B solutions) and to connect applications with the required protocol. More recently, since the advent of the SOA approach, ESB has also been presented as a way to create a SOA.

ESB editors clearly face two major challenges:

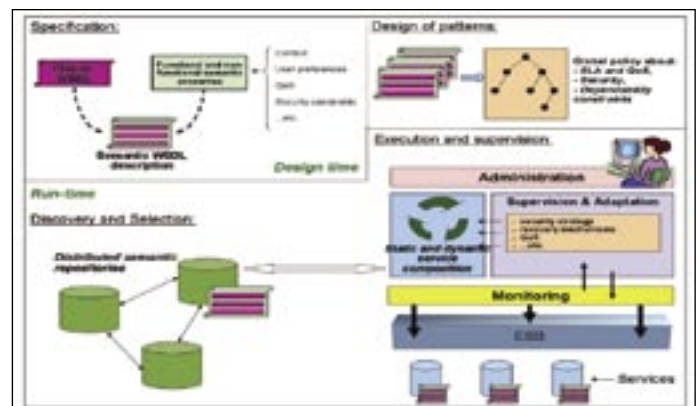


Figure 1: Semantic-based governance framework

- How to integrate heterogeneous technologies and products possibly produced by separate vendors in a way that sizes appropriately to each particular integration problem;
- How to provide the required features to fully address the specifics of SOA needs.

The Java Business Integration (JBI) standard (see the related section below) seeks to address the first challenge by adopting SOA principles. An ESB is built from JBI containers that can be seen as an integration framework, a host for Java connectors, an XSLT engine, or a mediation engine. JBI maximizes the decoupling between the JBI containers that all provide and consume services, while the ESB links the containers together allowing them to interact. Currently, it turns out that JBI-compliant ESBs are mostly open source ESBs that aim at promoting highly configurable and made-to-measure ESBs to fit business needs better.

Companies are currently struggling with the second challenge, since they realize that the ESB vendor's solution doesn't fit their needs. The reasons are manifold: for example, the ESB doesn't provide management models to control and enforce QoS at different levels and track consumer use; it doesn't fit into existing management and security frameworks; it's unable to connect to or evolve toward a highly distributed architecture that encompass the border of the enterprise. This problem will still exist as long as SOA technology editors don't address immediate and long-term business needs and concrete functional SOA.

Leveraging the ESB market momentum, the OW2 open source consortium (the old ObjectWeb) launched an ESB Initiative in June 2004 to facilitate the reuse and integration of several existing middle-ware components and respond to market requirements better.

JBI

PEtALS⁶ is the European flagship, JBI-compliant, open source Enterprise Service Bus developed by the OW2 community. It provides lightweight and packaged integration solutions, with a strong focus on distribution and clustering. Based on SOA principles, it offers a solid backbone for enterprise information system and acts as a bus where all data are exchanged in a technologically agnostic way.

The JBI specification promotes a plug-in architecture that enables the creation of tailored integration solutions by putting together best-of-breed integration components. The JBI specification is standardized in the JCP framework. JSR 208 describes JBI 1.0. The center part of the JBI specification is the Normalized Message Router (NMR). This NMR ensures loosely coupled communications between JBI components by providing standard SPIs that promote the exchange of XML documents between the components and loose references between the components via the use of their interface name.

Binding Components (BC) are "connectors" that interface the JBI bus with the rest of the information system. Binding Components enable both the exposition of external resources in the bus and the exposition of services available on the bus for their use by external consumers. Available BCs are: Filetransfer (send and receive files), FTP (put, get, or detect files on a FTP server), JMS (interact with an external JMS destination), Mail (send or receive files from or to an external mail service), SOAP (interact with external Web Services and expose JBI services as Web Services) and XQuare (lets users interact with databases).

Service Engines (SE) provide the integration logic. Available SE are: CSV (transforms a CSV document in an XML document), EIP

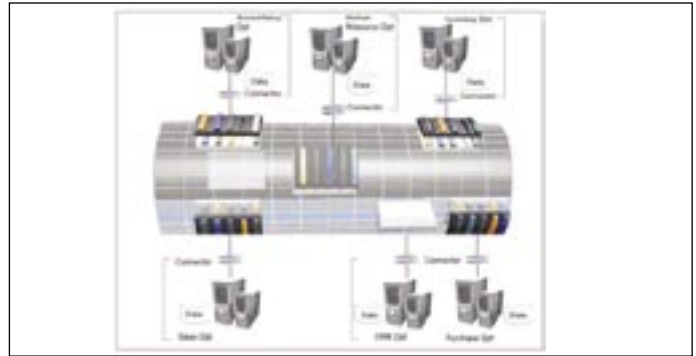


Figure 2: PEtALS Architecture

(implements Enterprise Integration Patterns), Forward (chain calls to service engines before forwarding the result to an output service), POJO (deployment of Java classes as services), RMI (access to the JBI bus implementation context from an external RMI client), and XSLT (processes XML transformations based on the XSL style sheet).

Semantic-Based Service Governance

Many companies are still in the early stages of SOA adoption and so the practice of SOA governance will be new territory for many IT professionals.

According to Wikipedia⁷, SOA governance is a concept used for activities related to exercising control over services in a SOA. SOA governance can be seen as an overlay on IT governance, but its focus is more organizational, since services are closely related to business activities. Loose coupling and the smaller granularity of services in a SOA also increase the demand on good governance. While the specific focus of SOA governance is on the development and use of services, effective SOA governance must cover the people, processes, and technologies involved in the entire SOA lifecycle.

Some key activities that are often mentioned as being part of SOA governance are:

- **Managing the portfolio of services:** planning the development of new services and updating current services.
- **Managing the service lifecycle:** meant to ensure that service updates don't disturb current service consumers.
- **Using policies to restrict behavior:** rules can be created that all services have to apply to ensure their consistency.
- **Monitoring the performance of services:** because of service composition, the consequences of service downtime or underperformance can be severe. By monitoring service performance and availability, action can be taken instantly when a problem occurs.

We use the following definition of SOA governance: "The ability to organize, enforce, and reconfigure service interactions in an SOA⁸." Consequently, the target SOA governance framework will provide a full set of software tools for SOA governance for managing the whole service lifecycle:

- **At design time:** a governance repository that stores information about services, SLA contracts, and other metadata such as semantic properties. Based on WS-Policy it allows service lookup and discovery based on metadata as well as service lifecycle management.

- **At runtime:** a JBI framework for policies' enforcement with a special emphasis on security and fault compensation and dynamic composition and routing.

The main advantage to using semantic properties for service governance is to obtain high-level information about functional and non-functional service's attributes. This allows more accurate ways to manage, select, and compose services to be implemented both at design time and at runtime.

The distributed repository stores metadata about services that include all the information necessary for governance processes: "classical" metadata such as WSDL but also SLA policies, composition models, transformations and mappings, and semantics. This repository offers an API that allows service lookup and discovery based on different flavors: UDDI style (white page, yellow page, green page), policy, and ontology-based semantic services.

The dynamic orchestration engine (see below) provides service composition mechanisms to dynamically react to external events to fulfil SLA contract and governance policies. Dynamic service composition uses semantic information to dynamically adapt service composition according to some events and automatically generate a plan.

A service policy enforcement mechanism takes into account necessary policies' enforcement as specified in SLA contract. The JBI approach sets up a pluggable strategy for governance based on a set of pluggable policy enforcement binding engines: it may be seen as an intermediate layer between the target service and the consumer.

SLA Definition and QoS Enforcement

The non-functional properties of services can be expressed in many different ways. These non-functional properties are commonly referred to as QoS properties. A trend is emerging at the junction between the SOA world and the Semantic Web of using ontologies to describe application domains by modeling their (functional and non-functional) properties.

In a service-oriented environment, it's advantageous for service consumers and providers to get guarantees regarding the services that they both require and offer. Usually these guarantees pertain to QoS. WSDL doesn't provide a way to express these guarantees so standards such as WS-Policy and WSLA (Web Service Level Agreement) exist to express additional non-functional attributes. SLAs are similar to the notion of contracts in component architectures, but transposed to service-oriented architectures. An SLA specification should let us correctly express the complex relationship that is

established between service consumers and service producers in a SOA environment.

It's foreseen that the semantic WS-Agreement specification could be a good starting point for expressing service level agreements in the PEtALS project, since its formal semantics would come up to our expectations.

WS-Policy⁹ is a W3C standard that offers mechanisms to represent the capabilities and requirements of Web Services as Policies. Web Services Policy is a machine-readable language for representing the capabilities and requirements of a Web Service. These are called "policies." Web Services Policy offers mechanisms to represent consistent combinations of capabilities and requirements, determine the compatibility of policies, name and reference policies, and associate policies with Web Service metadata constructs such as service, endpoint, and operation. It lets Web Services advertise their policies and Web Service consumers specify their policy requirements.

Infrastructure Monitoring

Infrastructure monitoring is also a crucial issue for such dynamic service governance to provide an SLA enforcement mechanism with sufficient input.

This requirement is addressed by a monitoring infrastructure that can be used for performance monitoring, availability, and QoS reporting. We deploy a dedicated framework based on specifying generic probes associated with an expected QoS level; given a particular service level and configuration we generate corresponding specific probes that are then deployed inside the service infrastructure.

Dynamic Composition

Service composition aims at enabling complex services built from a number of simpler services. The design of dynamic compositions using semantic services is a new challenge. Such an orchestration engine must take into account the semantic properties of each service to establish a coherent composition. The aim is to dynamically select the services that can preserve the global policy of the composition.

Processes being built today need business agility to adapt to customer needs and market conditions quickly. This would include incorporating new customers, partners, or suppliers used in a process. Web Services orchestration is about providing an open standards-based approach for connecting Web Services together to create higher-level business processes. Standards such as BPEL (Business Process Execution Language) and BPMN (Business Process Modeling Notation) are designed to reduce the complexity required to orchestrate Web Services, thereby increasing the overall efficiency and accuracy of business processes.

But, due to their limited expressiveness, such a high level of interoperability is hard to get with the current standard specifications. In fact, they're often dedicated to managing only the functional aspects of Web Services orchestration and on a somewhat low abstraction level. We propose to adapt the BPEL process engine to support:

- First and foremost, an ontology-based semantic selection of Web Services based on the functional requirements of the business processes and service consumers. To do so we use standards such as SAWSDL (Semantic Annotation for WSDL) and the OWL Web ontology language.
- A multi-step approach at different times (static/dynamic) to Web Services selection based on their non-functional properties (QoS)

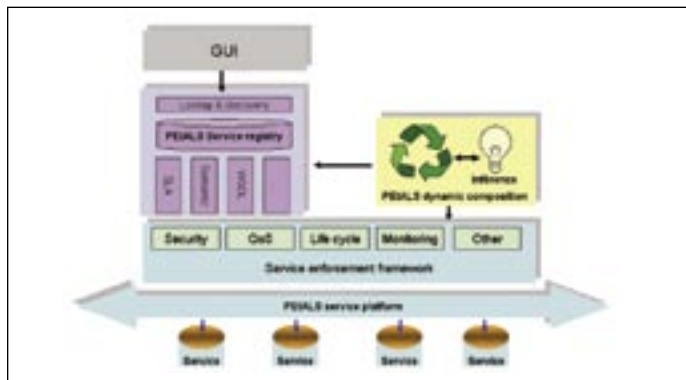


Figure 3: Governance Components

and the non-functional requirements of business processes. A two-phase protocol is specified that consist of:

- Statically filtering the list of registry-published services based on the non-functional requirements to obtain a set of acceptable services.
- Dynamically querying these services during business process evaluation to obtain their instantaneous and precise QoS values,
- Selecting the best service for Web Service calls using a QoS optimization decision process.

This engine is based on an autonomic encapsulation of the PVM¹⁰ (Process Virtual Machine) based on the OW2¹¹ Fractal Framework. The PVM is a general description framework for the process language. Developed by JBoss and Bull and available as an open source Java library, it's currently used to implement jBPL and Orchestra (the BPEL 2.0 engine). The Process Virtual Machine defines a common model that can be shared between all the graph-based execution languages. It also includes a strategy on how process constructs can be seen as software components. This enables support for multiple process languages and clearly shows how process technology fits right into software development projects.

Fractal is a modular and extensible component model that can be used with various programming languages to design, implement, deploy, and reconfigure various systems and applications from operating systems to middleware platforms and graphical user interfaces.

The Fractal component model makes heavy uses of the separation of concerns design principle. The idea behind this principle is to separate the various concerns or aspects of an application into distinct pieces of code or runtime entities. In particular, the Fractal component model uses three specific cases of the separation of concerns principle: namely separation of interface and implementation, component-oriented programming, and inversion of control.

The first pattern corresponds to separation of design and implementation concerns. The second pattern corresponds to the separation of the implementation concern into several smaller composable concerns, implemented in well-separated entities called components. The last pattern corresponds to the separation of the functional and configuration concerns: instead of finding and configuring themselves the components and resources they need, Fractal components are configured and deployed by an external, separated entity.

A Fractal component is composed of two parts: content that manages the functional concerns and a membrane of controllers that manages zero or more non-functional concerns (introspection, configuration, security, transactions, etc.). The content is made up of other Fractal components, i.e., Fractal components can be nested at arbitrary levels. The introspection and configuration interfaces that can be provided by the controllers allow components to be deployed and reconfigured dynamically. These control interfaces can be used either programmatically or through tools based on them, such as deployment or supervision tools.

Conclusion

Computer science is entering a new generation. The emerging generation starts by abstracting from software and sees all resources as services in a Service Oriented Architecture. In a world of services, it's the service that counts for a customer, not the software or hardware components that implement the service. Service Oriented Architectures

are rapidly becoming the dominant computing paradigm. However, current SOA solutions are still restricted in their application context to being in-house solutions. A service Web will have billions of services. While service orientation is widely acknowledged for its potential to revolutionize the world of computing by abstracting from the underlying hardware and software layers, its success depends on resolving a number of fundamental challenges that SOA doesn't address today.

We showed in this article how we're going to help realize a world where billions of parties are exposing and consuming services via advanced Web technology. The outcome of this work will be a comprehensive framework and infrastructure that integrates complimentary and revolutionary technical advances into a coherent and domain-independent service delivery platform:

- A distributed registry to store semantic information about services.
- A dynamic composition engine based on autonomic extension to a BPEL engine.
- A service policy enforcement mechanism that takes into account policies enforcement as specified in the SLA contract.
- A lightweight service infrastructure providing distributed ubiquity thanks to a peer-to-peer architecture.

This article is based on ongoing work supported by the SemEUse project, an initiative funded by the French government dealing with high-scale service distribution, SOA governance, and semantics for next-generation service infrastructure. Partners in this project are Thales, France Telecom, EBM WebSourcing, INRIA, LIP6, INSA Lyon, and INT.

References

Studer et al. Semantic Web Services Concepts, Technologies, and Applications. Springer. 2007.

1. <http://www.w3.org/TR/owl-ref/>
2. <http://www.w3.org/TR/sawsd/1>
3. <http://www.daml.org/services/owl-s/1.2/>
4. <http://www.wsmo.org/>
5. <http://www.gartner.com/>
6. <http://petals.objectweb.org>
7. http://en.wikipedia.org/wiki/SOA_Governance
8. Michael Wheaton – Sun
9. <http://www.w3.org/2002/ws/policy/>
10. <http://docs.jboss.com/jbpm/pvm/>
11. <http://fractal.objectweb.org/> ■

About the Author

Jean-Pierre Lorré is R&D manager of EBM WebSourcing in charge of next-generation SOA software products targeting a Web 3.0 service infrastructure including the European projects SOA4ALL that aim to support service infrastructure with billions of services and SYNERGY, which aims to enhance support of the networked enterprise. He also managed SemEUse, a French government-funded project that aims to develop a semantic ESB. Its main interests deal with Web 2.0 solutions based on SOA, ESB technologies, semantics, and open source models. EBM WebSourcing is a founding member of OW2, the open source consortium, and NESSI, the European Technology Platform dedicated to new wave service architecture. Jean-Pierre graduated from ISMRa (ENSI de Caen) in 1985 with a specialization in robotics.

Drag & Drop Data Conversion

Check out Altova MapForce® 2008 – the award-winning graphical data mapping tool from the creator of XMLSpy. Drag & drop to map, convert, and transform data between:



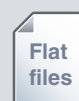
XML



Databases



EDI



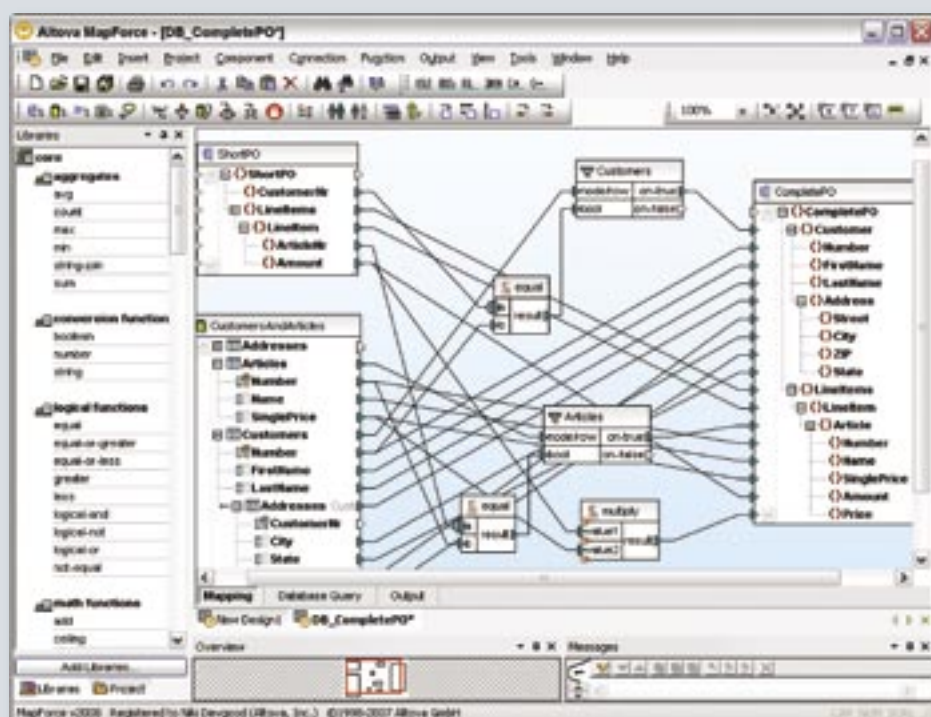
Flat Files



Web Services

- Drag & drop data mapping & conversion
- Support for all major relational databases
- Database query window with SQL editor
- FlexText utility for parsing flat files
- Support for EDIFACT and X12 EDI messages
- Connecting data to Web services
- Auto-generation of XSLT 1.0 and 2.0, XQuery, Java, C#, or C++ for royalty-free use
- Drag & drop Web services creation
- Extensible function library for filtering / processing data
- Visual function builder for custom functions
- Instant data conversion & output window

Once you have defined a data mapping in MapForce, simply click the Output Window to convert data instantly. Or, generate royalty free code – and deploy it with no additional fees or deployment adaptors required. With MapForce, you can implement data integration and Web services applications without writing any code!





DataServices WORLD

Welcome to DataServices World at SOAWorld 2008!

June 24, 2008 New York City



Ken North
Chair, DataServices
World at SOA World '08

DataServices World is about the confluence of databases, data warehousing, business intelligence, enterprise computing and Internet computing. Its focus is architectures and technologies for accessing data from heterogeneous data sources and providing that data to consumers such as components, services and applications.

Distributed processing, high-speed networks, powerful servers, components and collaboration define the landscape of 21st century computing. For building new systems and exploiting legacy applications and data, developers are looking to collaborative applications assembled from distributed components. The emergence of XML and web services spurred an increase in services-oriented architecture (SOA) adoption. SOA today can include web services, grid services, integration services, semantic web services, components and messaging systems.

Developers who've enjoyed success with component-based development are looking to new architectures with services as the new components. But even as the Service Component Architecture and other new technology gain tractions, some system characteristics remain consistent. Today's systems, like their predecessors, typically have a requirement for persistent information and databases.

Many organizations have a variety of persistent data stores, including SQL databases, geo-coded data files, spreadsheets, content management systems and XML. Services, applications, and mashups can consume and integrate data from disparate data sources. In an n-tier enterprise architecture and a service-oriented architecture, the logic for providing data from databases and other data sources resides in data access layers and data services layer.

Today's data services layers encapsulate logic for accessing data stores, typically using standards-based technology such as ODBC, JDBC, ADO.NET and Service Data Objects. These specifications define solutions for uniformly accessing and manipulating data from heterogeneous data sources.

At DataServices World, we'll uncover architecture and technology solutions for accessing, integrating and processing data from multiple sources while guaranteeing security and scalability. These solutions include robust, high-performance data access middleware, optimized databases, efficient protocol handling, tuned queries and state of the art data services. We'll be looking at technology of interest to CTOs, enterprise architects, system architects, information architects, developers, database gurus, consultants and analysts.



2008 Diamond Sponsor

REGISTER TODAY AND SAVE

www.dataservicesworld.sys-con.com