

SOA WORLDTM

MAGAZINE

www.SOA.SYS-CON.com

MAY 2008 / VOLUME: 8 ISSUE 5

Best Practices for SOA: Building a Data Services Layer

16 Leveraging SCA for
Standardizing the Composite
Application Ecosystem
SUDEEP MALLICK AND ASHWINI KUMAR JEKSANI

20 Choosing the Right Middleware
Stack for Your SOA in Healthcare
SANJAYA KARUNASENA

24 GO Service Ownership Framework
ANBARASU KRISHNASWAMY

28 The Unreliable Internet
MICHAEL GALPIN

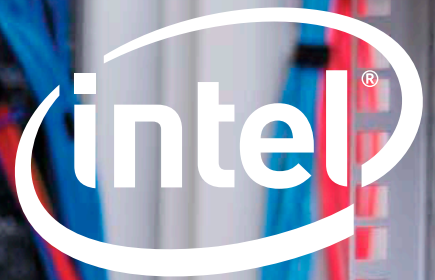


\$6.99US \$7.99CAN



05>

0 71486 03420 9



MAKE YOUR DATA CENTER A LEAN, GREEN IT MACHINE.

TODAY, EVERY DATA CENTER FACES THREE BIG CHALLENGES: SPACE, POWER, AND PERFORMANCE. And the pressure is on to deliver more for less. What's the answer? The Evergreen Data Center Framework from Intel and Tata Consultancy Services (TCS). This proven collection of today's newest technologies and best practices can help you design a data center infrastructure that doubles your performance—without expanding your power usage. It's built on techniques like virtualization and high-performance, power-efficient server platforms that can work together to transform your data center from power guzzler to lean, green IT machine.

For the whole story, attend "Evergreen Data Center from Concept to Completion" (see program guide for logistics) or contact Parviz Peiravi (parviz.peiravi@intel.com) or Sharad Isloor (sharad.isloor@tcs.com).

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.



Discovering Dr. Dolittle

WRITTEN BY SEAN RHODY

From the title, you might be thinking that I'm about to start this month's editorial with a reference to talking to animals and somehow tie that into SOA. Instead, what I actually would like to talk about is the pushmi-pullyu (I got the spelling from Wikipedia; I always thought it was "push-me pull you"). In the books, the pushmi-pullyu is an animal with two heads, each one going in the opposite direction.

What the pushmi-pullyu means to me in this particular context is two-way communication and the enfranchisement that it represents. In particular, I'm talking about the way Web 2.0 enhances SOA and further enables changes in our communication model from a push to a push-pull model.

You might think we already have a two-way model, but in most cases there's not really an equality relationship between the push and the pull. For example, a corporate site interacting with a consumer ought to be as much about the consumer's needs as the corporation's. Why, you ask? Because Web 2.0 is about social computing, and is as much a response to the unfriendliness of static web pages as it is a technology movement.

SOA discussions all tend to follow the same pattern – establish fundamentals, set up security, look at scaling, set up governance, and we'll get to BPM later. All of these issues are worthy of discussion, but they also tend to shortchange an area that I think needs even more focus – user interaction.

I don't just mean eliciting a response from a user either. I mean actually allowing the user to make decisions and guide his path through the services offered by a corporation. Social networking sites such as Facebook or MySpace are cases in point.

Traditional Internet marketing and personalization concentrated mostly on segmenting users from a corporate viewpoint. These are our high dollar users; they get more bandwidth and discounts. These are our low dollar users; they get a slower response. In some ways it made sense – if you ignore the fact that the user is an intelligent human being and can often resent being seen as merely a dollar sign or a variable in an equation.

The impact of social computing and the human network effect are already changing the way we have to look at end users. Far from viewing them as simpletons who need our guidance, we need to enable them as intelligent beings who can chart their own course through our services, and who enjoy the freedom to associate with one another on their own terms. In short, treat people like grown-ups. Now there's a frightening concept.

Yet it pays dividends every single time. Brand loyalty, word of mouth, customer adoption – all are driven by focusing on taking a service, and making it extremely easy for a user to consume it – in whatever way they may find appropriate.

Social networking sites could have static groupings – default settings that you could choose to enroll in. But that limits the participation of others – because it doesn't allow for open enrollment and dynamic creation based on user-defined groups – and creates a one-way conversation between the corporation and the user. What's really needed is the ability to work both ways and allow the user as much freedom as the corporation enjoys (or at least almost as much).

The lesson is clear for us – to make SOA truly meaningful we have to give users a means to interact with services in ways that they choose. We need to break down the walls that we've worked so hard to put up and find the courage to empower the end user (and the strength and intelligence to still have a secure environment without security checks interfering with individuality). If we can achieve that, we'll have made the whole SOA journey worthwhile.

That's our focus for this month – Web 2.0 and SOA. Now I've got to run; I've got a con call with a chimpanzee on line one and the lions want to discuss a shared governance model. ■

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](#)



8

3 **Discovering Dr. Dolittle**

SEAN RHODY

6 **News**

8 **Best Practices for SOA: Building a Data Services Layer**

JASON BLOOMBERG AND JOHN GOODSON

12 **SOA or DOA**

HON WONG

16 **Leveraging SCA for Standardizing the Composite Application Ecosystem: Rationale and a Use Case**

SUDEEP MALICK AND ASHWINI KUMAR JEKSANI

20 **Choosing the Right Middleware Stack for Your SOA in Healthcare**

SANJAYA KARUNASENA

24 **GO Service Ownership Framework**

ANBARASU KRISHNASWAMY

28 **The Unreliable Internet**

MICHAEL GALPIN

34 **SOA Issues Are People Issues...Not Technology**

DAVID LINTHICUM



28



ACCELERATE PROCESS IMPROVEMENTS. WE'LL SHOW YOU HOW.

Leverage existing assets. Integrate silos of information. Improve processes — Faster

Speed up processes and shift your SOA into top gear with Software AG Business Infrastructure Software. We'll help you drive down integration costs, drive up the value of existing applications and deliver new applications and services — faster. You're destined for success when you can manage and access critical data instantly and also monitor business operations in real time.

See how fast you can reach your goals, visit Software AG at SOA World 2008! www.softwareag.com

News

Volvofinans Drives New IT Strategy with Software AG Technology

(New York) –Software AG has announced that Volvofinans has selected CentraSite as its enterprise platform for service-oriented architecture (SOA) governance and meta-data management. CentraSite will serve as the foundation for the new IT infrastructure being implemented by Volvofinans to accelerate the delivery of new products and services.

With total assets of approximately 2.5 billion, Volvofinans provides financing and related services for Volvo and Renault vehicles throughout Sweden. Volvofinans currently manages nearly 250,000 loans and leases, including 350 corporate programs, with an additional one million Volvo credit cards in circulation as well.

Volvofinans had been moving toward an SOA-based IT infrastructure for a number of years. This would allow the company to more quickly and cost-effectively create new processes and applications from reusable service components. As a result, Volvofinans could more easily offer its 70 dealers and its fleet administrators customized and personalized IT services based on their size or the mix of different automobile brands being sold.
www.softwareag.com

StrikeIron and Cortera Partner to Deliver Live Credit and Business Demographic Data over the Web

(Research Triangle Park, NC) – StrikeIron has announced that U.S. business information services provider Cortera's Credit Pulse and Business Vitals Web Services have been added to the StrikeIron Web Services Marketplace.

The Cortera services join StrikeIron's 100 plus Web services from providers including D&B, Gale, MapQuest, Midnight Trader, Tax Data Systems, Walls Street Horizon, and Zacks. Using the Internet as a platform, StrikeIron removes the complexity involved in managing diverse data formats from multiple data sources and provides access to data that can be utilized by customers in their Web sites, enterprise mashups, business processes, and applications.
www.cortera.com
www.strikeiron.com

SAP Accelerates the Path to SOA for Customers

(Walldorf, Germany) – SAP AG has announced a design and development governance offering for enterprise service-oriented architecture (enterprise SOA.) The new offering will enable SAP customers to experience tangible business benefits as a result of an enterprise SOA strategy. Growing adoption of the SAP enterprise SOA strategy – the design and reuse of rich enterprise services to enable efficiencies in business performance – has led to customer requests for training and education involving SAP's proven design and development governance methodologies to help them speed their path toward enterprise SOA.
www.sap.com

TIBCO ActiveMatrix to Provide Predictive Service Level Agreement Management for SOA

(San Francisco) – TIBCO Software has announced it has added new functionality to its ActiveMatrix Service Oriented Architecture (SOA) platform to automatically predict and fix IT service problems before they slow down access to business information. The new TIBCO ActiveMatrix Service Performance Manager capabilities mark the next wave of SOA infrastructure software technology from TIBCO.

As SOA continues to grow in use and sophistication with expanded use of individual services across the enterprise, customers have the challenge of understanding and managing the dependency and re-use of services that may have originally been built for one specific use and are now depended on by multiple services, creating potential weak points in the system.

ActiveMatrix Service Performance Manager understands business Service Level Agreements (SLA) and can monitor actual latency within context and time. It predicts and solves problems by taking action to, for example, replicate an SOA service module or bring on-line additional servers to automatically head off slowdowns and service interruptions and sends a notice of the action taken.
www.tibco.com

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.

Cost Effective SOA Management

small budget

Big Questions

mission-critical services

major Pressure

short timeline

reduced staff



A flexible Web Services Management solution for the enterprises that need the following capabilities

- ▶ Runtime Governance, Security & Policy Enforcement
- ▶ Visibility, Management and monitoring of the SOA environment
- ▶ Agentless visibility into SSL and non SSL Web service traffic
- ▶ Multiple Deployment options to fit Your environment
- ▶ Cost effective

Download a free trial at www.managedmethods.com



Managed Methods
.....
www.managedmethods.com
.....

Best Practices for SOA: Building a Data Services Layer

**Choosing best-of-breed data access
middleware is an important enabler of SOA**



BY JASON BLOOMBERG AND JOHN GOODSON

These days nearly every sizable organization has either implemented some form of SOA or has it on their roadmap. They quickly find that SOA efforts tend to expand like spider webs, eventually touching every corner of IT as well as the business itself. Due to the vital role that data plays both in business and systems operations, database architectures, information specialists, data integration experts, and anyone responsible for data persistence in an organization are increasingly being called on to contribute to their organization's SOA initiatives — whether or not this was intended at the onset.

Information locked away inside monolithic application silos has proven to be a stubborn obstacle to the flexibility that modern businesses require. If businesses are to have any hope of building flexible services that offer the performance and agility needed to succeed with SOA, those businesses must solve the technical challenge of accessing information — that is, data — across application platforms and their organization as a whole.

System architects who fail to devote sufficient planning to data

access issues and attempt to layer a service-oriented approach on top of their existing data sources often find that providing flexibility above the service abstraction requires complex changes at the data source level, impeding the agility they sought and thereby undermining one of the core rationales for implementing SOAs in the first place.

In traditional distributed architectures, developers write a data access code, which they might then seek to make reusable. However, if a problem exists with this data access code, that problem essentially becomes propagated with adverse impact on any application that requires access to that particular data source. Furthermore, whenever anything changes — including the underlying database, the data model, or the version of the coding environment being used — the data access code must be updated everywhere it appears.

Considering that data sources can range from all kinds of structured data stores (such as relational databases, mainframe data sources, and enterprise applications) to semi- or unstructured data such as Web pages, PDF documents, office application files, XML documents, e-mail, media content, print streams, or a wide variety of





content and data feeds and formats, it becomes clear that accessing and processing all these disparate types of information from so many disparate sources via the tightly coupled approach of traditional distributed data access would constitute a technical support challenge of monumental proportions.

Properly architected SOA presents both business functionality and data as abstracted services. Applied to data access, if access is abstracted as data services and the access code moved into supporting infrastructure, then problems can be addressed and changes can be supported throughout the environment in a much more loosely coupled and flexible manner. In essence, the data services layer provides a single abstracted point of access for all data access, update, and creation operations, providing a holistic view of the data models that the underlying persistence layer relies on. It acts as a bridge between the business services and the underlying data persistence layer; business users needn't concern themselves with whether the data they are consuming originated in the database, an enterprise application, a file system, another company, or anywhere else for that matter. This promise of ubiquitously accessible data freed from the constraints of its sources is liberating for companies as they struggle with the challenges of systems integration.

A data services layer must provide an interface that exposes a standard set of reusable data services for reading and writing data, independent from the underlying data sources. The loose coupling it enables between applications using the services and the underlying data source providers lets database architects modify, combine, relocate, or even remove underlying data sources from the data services layer without requiring changes to the interfaces that the data services expose. As a result, the architects can retain control over the structure of their data while providing relevant information to the applications that need it. Over time, this increased flexibility eases the maintenance of enterprise applications.

Before the advent of SOA, developers built the capabilities that the abstracted data service layer could provide using manual coding, tightly embedding that code into the application under construction. Embedding such data access and data abstraction code directly into applications limits the flexibility and reusability of the resulting applications and, as a result, enterprises looked to traditional middleware such as Extract, Transform, Load (ETL), and Enterprise Application Integration (EAI) products to provide the capabilities of the data service layer from a middleware perspective. The ETL approach is best suited to static applications that don't require the flexibility that SOA can provide. It can be costly, as well, and requires a high management

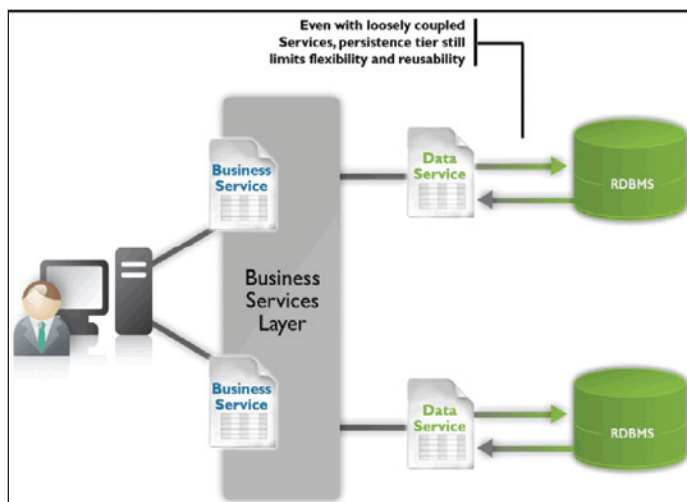


Figure 1: Loosely coupled services abstracting replicated data

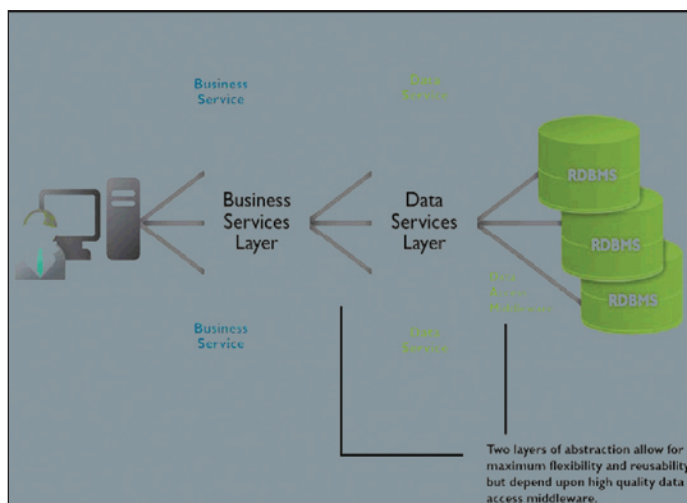


Figure 2: Data services layer in an SOA implementation

overhead. The EAI approach centralizes data interaction logic, but still fails to provide the flexibility many organizations require from their data services layer.

Even when an organization is implementing true SOA, however, a poorly architected data services layer can lead to performance issues. In many cases, each application has its own database which contains a duplicate copy of business reference data such as customer information, product information, and inventory levels, as shown in Figure 1.

The organization must synchronize such databases on a regular schedule, which can lead to stale or inconsistent data between applications. In this case, even when an organization has exposed application functionality as loosely coupled services the persistence tier still limits the flexibility and reusability of the services. Incorporating a data services layer into the SOA implementation addresses this problem. Data services offer transaction and connection management to multiple applications, as shown in Figure 2.

The data services layer manages the relationships between the data services and ensures that each application is aware of all data changes, regardless of the cause of the change. Leveraging data services as infrastructure services beneath the business service abstraction increases reusability and flexibility, and shortens development and rollout time for new services. A well-architected data services layer relies on consistent, high-performance data access middleware that leverages industry-standard APIs such as ODBC, JDBC, ADO.NET, and SDO.

To sum it up, building a data services layer as part of an SOA implementation provides the infrastructure necessary to reap the full benefits of SOA. Specifically, it provides the following solutions to common data problems:

- **Reduced reliance on hand-coded persistence logic** — By building abstracted, shared data services, it's possible to leverage best-of-breed data access middleware to support the data services as part of an architected plan.
- **Inflexible integration is addressed** — Instead of the point-to-point or hard-coded integration of traditional integration approaches, the data services layer provides for loose coupling at the persistence tier.
- **Data query bottlenecks removed** — The data services layer enables organizations to implement content-based routing techniques to avoid bottlenecks.
- **Improved data consistency** — The data services layer functions as a single locus for data access in the enterprise, helping an organization drive toward a single source of truth for its data. Implementing a data services layer helps ensure that applications always pull data from the correct sources and consistently provide appropriate information back to all applications.

Without doubt, architects who focus on SOA must take a hard look at solving the data access challenges in order to plan a successful, loosely coupled architecture. Accessing data in the various stores across the enterprise in the most efficient, flexible manner is the basis for building data services, and is thus critical for building all the services in the SOA implementation.

The Importance of Best-of-Breed Data Access in Building a Data Services Layer

Ask just about any business executive about the core value IT provides to the business, and they're likely to respond that data (in the form of useful information) rather than applications form the lifeblood of their business. In essence they're correct; information technology is by definition all about information. But actually the two — data and applications — can only be separated in theory, not in

fact. Data by itself is often inaccessible and/or unintelligible without the applications that process it, and most applications in turn serve no real business purpose without data.

While establishing a data services layer is essential if you want to fully apply SOA best practices to data, it's equally important to realize that the foundation of that data services layer consists of data access. Data access, in turn, depends on standards such as ODBC, JDBC, ADO.NET, and SDO. Even if you're using a persistence layer like one built with the open source Hibernate toolkit, for example, high-quality data access middleware is critical. A slow JDBC driver under Hibernate or a slow ADO.NET provider under Microsoft's Language Integrated Query (LINQ) will inevitably impede the services. The results can be far reaching. Accessing the data in the various data stores across your organization in the most efficient, flexible manner is a core reason for building data services. It's therefore critical for building all the services in your SOA implementation.

Performance speed is not the least of the issues that can be affected by data access middleware. And, because the data services layer essentially virtualizes data access, those services can hide a plethora of potential data access pitfalls. Other such pitfalls can include:

- Scalability issues
- Database platform, application, and version incompatibilities
- Differences in how the various data sources handle the details of standard data access operations such as create, read, search, update, and delete
- Data source security priorities, which might vary from database to database, table to table, or even row to row
- Data mapping challenges arising from semantic differences among heterogeneous data
- Problems arising from the mixture of structured and unstructured data — such as file formatting issues
- Differences in versions of SQL

SOA can actually serve to magnify limitations in scalability, performance, and interoperability imposed by data access because organizations are both more likely to reuse the underlying code and leverage existing data services across many services and composite applications.

Your first step then in resolving these data-related issues is to build shared, centralized data services, which will result in an adaptable and easily maintained SOA implementation. This way data access logic only appears in one place, no matter how many applications consume it. The result is that, instead of scattering data-related services invocation code throughout each business service, centralized data access helps you create an environment that enables a best-of-breed data access solution to address all those data access issues.

This last point — the use of best-of-breed access solutions — can't be stressed strongly enough. Even many people involved in data management rarely give data access middleware more than a passing thought. One reason for this is that commercial databases all include connectivity drivers bundled with the database software, which are often used by default. The fact that these "free" drivers may be less than optimal for a given IT environment typically doesn't come up unless and until either a technical or a maintenance issue becomes serious enough to be traced back to the data connectivity layer. For reasons we've described, such issues can be compounded in an SOA and can also be extremely difficult to diagnose.

Quite simply, it is ill advised to rely on database-bundled data access middleware for your data services layer in an enterprise SOA. The same holds true for freely available open source drivers. As the

very foundation of your data services layer, the type of data access you use deserves serious consideration and careful planning. Dynamic environments where multiple services reuse data access code and new services regularly go into production require that such code meet rigorous requirements.

Your best bet is to go with third-party data access middleware, preferably from a supplier whose core business and expertise consists of data connectivity. Look for specific attributes for your data access middleware, including capabilities for boosting query performance. Such attributes can include connection pooling as well as support for tunable data access performance such as adjusting network packet size. For scalability and high availability, data access middleware should be multithreaded and thread-safe, and offer client load balancing and failover to alternate servers.

It's also important for data access middleware to support different types and versions of databases as well as all the subtle variations in SQL they support. Such support for heterogeneity should also extend to multiple computing platforms, chipsets, and operating systems. In addition, for the best performance and flexibility, data access middleware should offer wire protocol drivers to avoid the overhead and maintenance issues of off-the-shelf database drivers. Wire protocol drivers obviate the need for database client software and libraries, simplifying installation and administration as well as offering a much more efficient and high-performing operation. Such drivers must support the full panoply of relevant standards, including JDBC, ODBC, ADO.NET, and – over time – SDO.

Finally, you should incorporate data access middleware in a comprehensive IT security strategy that covers both network security and database security, with secure communications and secure code.

It should integrate with multiple solutions for authentication and authorization, too.

It's important to choose best-of-breed data access middleware as a critical building block for any SOA initiative you undertake in your business. Data access is a fundamental building block of SOA, and if you fail to make good choices there, the entire services infrastructure will suffer for it. Fundamentally, regardless of the SOA infrastructure that runs above the data services layer, there's no question that data access remains a key building block technology for SOA. ■

About the Authors

Jason Bloomberg is managing partner at ZapThink, LLC and co-author of the book *Service Oriented or Be Doomed! How Service Orientation Will Change Your Business* (John Wiley & Sons, 2006) released in February, 2006.

John Goodson is vice-president and general manager of DataDirect Technologies, a comprehensive provider of software for connecting disparate critical business applications to data and services.

Data Services Use Case Scenario

A large brokerage house had demanding requirements for reliability, performance, and scalability, processing thousands of transactions per second at peak volume. Via data services technology, it delivers immediate responses to its customers and is able to scale to meet the needs of its business with near-perfect uptime. As part of its SOA initiative, it implemented dozens of business services that utilize the same data services. The consuming applications share a common data model as well as common data. The IT team selected data access middleware that provided optimized updates, load balancing, and failover.



Next Generation Data Center Management

PlateSpin provides a unified suite of solutions to help enterprises adopt, manage and extend their use of server virtualization in the data center. From anywhere-to-anywhere Workload Portability™ and protection to data center optimization and virtual infrastructure management, PlateSpin's unique solutions have helped over 5,000 enterprises worldwide optimize their data centers, reduce IT costs and improve the speed and quality of server consolidation, hardware lease migration and disaster recovery initiatives.

Visit us at our booth to find out more.

www.platespin.com



© 2008 PlateSpin ULC. All rights reserved. PlateSpin and the PlateSpin logo are registered trademarks of PlateSpin ULC. PlateSpin conversion and optimization technology and related products are patent pending. PlateSpin is a Novell company.

SOA or DOA

Production Management for Performance Assurance

BY HON WONG

Web applications built on a Service Oriented Architecture (SOA) promise to greatly improve IT efficiency and business agility. SOA establishes data and protocol standards so that existing internal and third-party application modules or services can be reused and orchestrated into business applications. Unfortunately, while SOA enables the rapid implementation of business applications, it also greatly increases the complexity of managing performance when these applications are deployed in production — often diminishing the benefits of the SOA adoption. Without an effective way of monitoring application performance, and quickly diagnosing and correcting problems, there is a high likelihood that an SOA application could be dead on arrival (DOA).

There are several aspects to implementing SOA that makes it difficult for IT to manage application performance:

- **Slowest Common Denominator:** The service level of an SOA application is limited by the service level achievable by the worst-performing services it touches on the network. For maximum flexibility, services can even be supplied by third-party vendors, and may be running on different computing platforms. This makes it virtually impossible for IT to characterize the performance of constituent services or control the numerous moving parts that can affect application delivery and performance. Reusing services also means that performance flaws found in common services are replicated across applications, creating multiple points of failure. As a result, it is very difficult to determine quantitatively if the service level exceeds end-user expectation or, at a minimum, meets the performance targets stipulated in service level agreements (SLA).
- **Rube Goldberg Machines:** For a Web-delivered SOA application, it is also difficult to determine quickly the existence and source of service delivery problems. Culprits can include any of myriad “moving parts” in the delivery mechanism, such as the client’s PC, the Web cloud, the data center, the composite services, and/or third-party service provider’s services or infrastructure. No list of “likely suspects” can possibly be exhaustive enough for such a complex environment.
- **The Tangled Web:** The task of diagnosing SOA application performance problems is further complicated by the lack of a predictable or pre-determined path a particular transaction takes when traversing through the application or infrastructure. Unlike client/server computing where, for example, an inventory lookup

is routed through a defined network segment and is served by an ERP application installed on a fixed physical server, that same inventory lookup in an SOA environment can be routed dynamically through the Web cloud, and be served by multiple virtual or physical servers on multiple logical tiers of the infrastructure. A third-party Web Service call might also be initiated to a supplier’s infrastructure to account for any inventory en route from the supplier’s factory. This complex orchestration involving a non-deterministic transaction path makes recreating and diagnosing performance problems very difficult.

To tackle these challenges so that the SOA application is not DOA, IT needs two capabilities at their disposal. First, IT has to be able to monitor application performance as experienced by the real user because that is the only place where the performance of all of the constituent services’ is felt. Furthermore, if a service-level violation is detected, they have to be able to quickly trace the offending transaction and pinpoint the cause of the performance problem. Capitalizing on these capabilities in a systematic way allows IT to:

- Improve service levels by discovering performance bottlenecks and shortening the time to problem resolution.
- Lower the costs (and frustrations) of operation management by eliminating unnecessary triage meetings and fruitless problem recreation attempts.

So how do existing end-user monitoring techniques fare in delivering the capabilities IT needs to avoid application DOA? Not well. Let’s take a quick survey of existing monitoring techniques as it applies to SOA performance management:

- Sniffers or other packet capture appliances can estimate the round-trip response time of a packet algorithmically, but lacks the ability to measure the response time of transactions whose path does not pass through the point on the network where the appliance is installed. Take mashup applications as an example – critical data objects are supplied by third-party applications, potentially bypassing any sniffers installed in front of the Web servers. In the data center, the sniffers are also blind to Web Services calls among servers or third-party services that forms the basis of an SOA application.
- Server monitoring tools can only report on the transaction response time of the infrastructural silo they are monitoring. For example, popular J2EE application server monitoring tools measure only the response time on transactions that involve the

BPM & SOA— Building Blocks for the Enterprise



Contrary to popular perception, business and IT can work together. And BPM and SOA are the building blocks that make this happen. SOA enables asset re-use and BPM accelerates SOA implementations with greater business-IT synergies.


Interstage® Business Process Manager is an award-winning BPM Suite. CentraSite™ is a completely standards-based Registry-Repository.

For more information on
Fujitsu's BPM and SOA offerings,
please visit us at

www.fujitsu.com/interstage

or contact

Interstage@us.fujitsu.com

The Fujitsu logo consists of a red infinity symbol above the word "FUJITSU" in a bold, red, sans-serif font.

application server. Transactions served directly by the Web or third-party servers, which never touch the application tier, cannot be managed by the J2EE monitoring tool.

- Traditional Web site performance monitoring services can detect whether an SOA application is available or not. It cannot report on performance as experienced by real users or provide actionable information that pinpoints the cause of problems to guide corrective effort.
- Pure-play SOA management products can help IT model the interdependencies among various services and provide limited transaction path information, but are often blind to the health of the infrastructure that supports the orchestration. More importantly, they have no visibility into the ultimate performance as experienced by the end user.

In terms of providing “real” actionable information for managing SOA performance, these legacy tools are deficient not only in the type of performance data they collect, but also in where the data is collected. It is critical to define application performance as response time as perceived by the end user instead of server, network, J2EE, database, or other silo-oriented metrics. There is no argument against the fact that the experience of the end user is the only thing that matters. Moreover, for mashup applications where the Web page is served by multiple servers or third-party data centers, or when the application is delivered using content delivery networks, the application might not even come together until the content arrives at the browser. As a result, the only valid measurement of good or bad SOA application performance is the one measured directly at the real user’s browser.

To deliver real user monitoring and transaction tracing capabilities to avoid SOA going DOA, IT needs three integrated functions in their SOA performance management tools:

- **Detect:** “You cannot manage what you cannot measure.” Having a quantitative way to determine whether the SOA application meets service level requirements is the first step in SOA management. In other words: “Is the right application response (data, page, action, etc.) delivered to the right user in the right amount of time?” There are numerous QA techniques to assure that the right application response is delivered. Furthermore, most organizations have the necessary security to assure that the right person is receiving the information. But assuring that the information is delivered at the right time to the end user through the complex Web-based SOA infrastructure is another matter. Having the ability to non-intrusively monitor application performance as experienced by real users is an absolute necessity because it is the: (1) only way to accurately detect problems experienced by real users of SOA applications for service-level assurance and reporting, and (2) it provides a key driver for making process or application response time improvements. The starting point of such monitoring is the end user’s browser, where the application truly “comes together.” It is at the browser that IT can take into account “last mile” circumstances and identify whether an incident has occurred that will affect user satisfaction. Data collected by legacy tools that focus on monitoring a particular technology silo — like network routers, Apache Web servers, WebSphere application servers, or .NET frameworks — cannot be extrapolated to determine what the actual end users of complex

SOA applications are experiencing in the browser.

- **Isolate:** Once the application performance as experienced by the end user is known, it has to be correlated with the performance profile of all the infrastructure and application components involved in the delivery of the SOA-based response. Since composite applications (1) are made up of services that are “black boxes” whose performance cannot be controlled or tuned by those orchestrating the application, (2) run on physical or virtual infrastructure components that are not entirely under the control of IT operations, and (3) may have different parts of a transaction served by different data centers or servers including third-party service providers, it is important that the performance of each transaction is reported and correlated across all infrastructure tiers, third-party data centers, and application components. Performance correlation can be achieved by painstaking log file analysis and heuristics to match up IP addresses and request times across various tiers, but this methodology is error-prone and difficult even if access to all of the logging information is available, and impossible if the transaction touches a tier outside the data center where log files are unattainable. Another simpler mechanism is to tag each transaction originating at the end-users’ browser non-intrusively and dynamically trace it through the entire infrastructure, logging appropriate performance data at each tier. Such an end-to-end view of performance based on the real user’s experience offers the bird’s eye view needed to pinpoint incidents, errors, bugs, or bottlenecks that impact end-user response time.
- **Optimize:** A holistic browser-to-database view of transaction performance provides actionable information so that ad hoc or trial-and-error approaches are no longer needed to identify and respond to performance problems. Without actionable information, IT incident response teams will likely spend more time debating the cause and attempting to re-create the problem than implementing a fix and restoring the business function. By analyzing correlated transaction performance information over time, IT can identify leading indicators of performance concerns so they can proactively resolve them before an incident impacts user satisfaction or business productivity. Furthermore, the information also helps identify areas for performance improvement in the infrastructure, services, and application.

Having these three functions integrated into a single SOA performance management tool gives IT an early respond system to detect and react to end-user performance problems before they impact thousand of users or lead to costly site outages. Information on business impact or performance bottlenecks should be fed back to the operations staff for infrastructure or process improvement, and to the developers for application optimization.

Yes, SOA can greatly enhance business agility and lower cost of application development. However, without a real user-oriented approach to manage SOA deployment and production management systematically, it is highly likely that the SOA application will be DOA.

About the Author

Hon Wong is founder and CEO of SymphonIQ and also founded NetIQ, where he served on the board of directors until 2003. He has co-founded and served on the board of several other companies, including Ecosystems (acquired by Compuware). He provides Web Application Performance Management (APM) insights and more on his blog, *The Web APM Blog*.



GUARANTEED EFFICIENT ENTERPRISE



\$150,000 THERMAL GUARANTEE
AGAINST HARDWARE DAMAGE TO YOUR SYSTEM

**WORLD'S ONLY
THERMAL GUARANTEE**

Introducing the Efficient Enterprise:™ more power, more control, more profits

Can your legacy system say the same?

Your service panel limits the amount of power available.
Your budget limits the amount of money. You have to
stretch every bit of both as far as you can. What you
need is the APC Efficient Enterprise.™

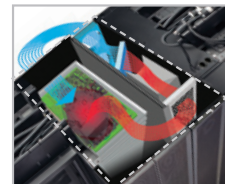
The APC solution offers modular scalability so that you
pay only for what you use; capacity management so that
you know where to put your next server; and dedicated
in-row and heat-containment systems that improve
cooling and thermal predictability. An Efficient Enterprise
earns you money through the pre-planned elimination of
waste. For example, simply by switching from room to
row-oriented cooling, you will save, on average, 31% of
your electrical costs.

Our system reimburses you

Whether you're building a new data center or analyzing
the efficiency of existing systems, your first step is
knowing where you stand. Take the online Enterprise
Efficiency Audit to see how you can reap the benefits of
a smart, integrated, efficient system: more power, more
control, more profits.



CAPACITY MANAGEMENT



CLOSE-COUPLED COOLING



P = Power C = Cooling R = Racks

CONSERVE POWER



CONTAIN THE HEAT



Visit the **APC Technical Session: "Maximize Your Virtualization Payoff;
There's More Than You Think"** Domenic Alcaro, Director - Enterprise
Segment Northeast — **Grand Ballroom**

APC
by Schneider Electric

Leveraging SCA for Standardizing the Composite Application Ecosystem: Rationale and a Use Case

Catalyzing the adoption of composite applications via standards

BY SUDEEP MALLICK AND ASHWINI KUMAR JEKSANI



Composite applications are the new breed of applications that are built rapidly by composing ready-made configurable and customizable service components together. This is similar in spirit to creating new recognizable objects by snapping together pre-fabricated Lego blocks in unforeseen ways. The advent of composite applications could be considered a logical progression from Web Services, component-based development, and packaged application platforms. Service-enabling existing legacy applications into standalone modular service components with stable interfaces greatly enhances the reusability of enterprise information assets. Depending on the business context, these reusable service components could be re-configured with different properties, giving them situation-specific

behavior. They could then be linked together in meaningful ways to deliver new business applications enabling new business processes. In some cases some small yet critical implementation aspect of the service could even be redeveloped (via custom scripts) to provide the requisite behavior – which is similar to packaged application-style customization. Reusability, configurability, and assembly are key to composite applications. Although the idea of assembling configurable components has been around for quite sometime, the increasing maturity of tools and platform suites from leading vendors is pushing its adoption into the mainstream. SOA plays a critical role in realizing this vision.

We could define composite applications as “rapidly composable applications that enable business workflows through the assembly of customizable and configurable pre-fabricated assets that can be deployed at runtime to access the diverse enterprise resources and applications required for executing the business scenario while leveraging standardized-based process, data, usage, and interaction models.”

Challenges

Building composite applications may look like an attractive idea,

however, it's rife with challenges:

- The first challenge is the identification of reusable, configurable service components amenable to assembling into full-fledged business processes or workflows.
- The second challenge is the identification of the context-specific variations of these reusable service components. Depending on the business and technical context, these aspects of the service components would have to be specified before deployment. For example, the same service that provides analytical data on product sales to another consuming enterprise application over RMI/IIOP could be reconfigured and re-provisioned to provide data over SOAP/HTTP to the mobile devices of field sales personnel.
- The third challenge is to govern the lifecycle of such modular assets – versioning, testing, provisioning, demand and supply management, ownership, usage policies, etc. – typical concerns of SOA management and governance.
- The fourth challenge is handling the heterogeneity and complexity at runtime. The service components would ultimately have touch points with multiple existing enterprise applications resulting in complexity in terms of transactions, security, performance, etc.

Constituents of a Composite Application

The typical constituents of a composite application are:

- Service components exposing functionality from existing enterprise applications
- Connectors and adapters that make connections possible between disparate systems
- Document handlers and transformers that enable conversion among different document formats (XML or non-XML, industry vertical open standards-based or otherwise, structured or semi-structured, etc.)
- Business process templates, sub-process templates, workflows
- Data and message schemas specifying a common information model (CIM) for enterprise entities, schemas for component/service to component/service interaction messaging

- Business rules, business policies, SLA definitions, UI assets, business event metadata and executables, workflow roles

So a comprehensive gamut of ready-made and configurable items can go into making a composite application. These individual constituents may have been tried and tested individually during development as well as in different business use cases. However, whenever a new composite application is built from these existing components, it has to undergo rigorous testing and validation iterations.

SCA & Composite Applications

Service Component Architecture (SCA) 1.0 is a new standard specification that has been created to address these concerns of building composite applications. It enables a standardized way of specifying, configuring, customizing, and composing the constituents through interconnections. The process of composition is recursive. SCA is an umbrella of specifications, the two fundamental specs being:

- **SCA Assembly Model v1** – It defines the building blocks and the composition mechanism. It covers concepts such as domain component, implementation, interface, composite, reference, wire, service, component type, binding, recursive composition, and extensibility aspects of the SCA model, packaging, and deployment. The SCA components and composites are specified in Service Component Definition Language (SCDL) files (XML-based). Components are implementations that have been configured using properties and adhere to a service interface definition that is the only portion of the component exposed to the external world. The service is exposed using different bindings accessed over compatible wires or protocols by service consumers. The components could be composed into composites that could in turn be treated as components hiding internal composition details. The components themselves could be consumers of services from other components through references over compatible wires. These components, composites, and their interconnections are enacted inside an SCA domain that uniquely recognizes these entities and their configurations inside that domain.
- **SCA Policy Framework v1** – It defines the building blocks for specifying the non-functional aspects of the components and composites. It covers concepts such as policy, policy sets, intents, mechanisms of attaching the policies and intents to SCA components and composites, roles and responsibilities involved in building composites. In its current version this

specification covers two important non-functional aspects – policy specifications of security and reliability aspects. There are two types of policies – implementation (e.g., authorization required when accessing the service implementation) and interaction (e.g., the security and reliability of message delivery during the interaction). Usually these policies are framed as WS-Policy assertions. Intents are abstract policy intentions such as “atLeastOnce,” “authentication,” “integrity,” “confidentiality,” “confidentiality. Message,” “confidentiality. Transport.” Corresponding policies are concrete implementations of intent such as the particular authentication technology like the X.509 certificates that JMS provided at least once delivering the message. The policies can be grouped into policy sets and intents into profile intents.

The components and composites might embody implementations in diverse platforms and yet would still need to be connected to each other to accomplish a complex cross-functional business process in an interoperable manner. Since SCA aims to provide a standard for composition platforms, it has specifications for standardizing component descriptions and connectivity from various implementation platforms. SCA currently covers the component and composite specification for the following platforms:

- Java
- C
- C++
- BPEL
- COBOL

The specifications for the above cover both service consumer- and provider-side implementations (called Client and Implementation). SCA also covers component implementation aspects in detail on Java and the Spring Framework.

To provide interoperable connectivity among components and composites, SCA v1 currently targets some widely adopted bindings (and hence natively supports the corresponding “Binding Types”) such as:

- Web Services
- JMS
- EJB Session Bean
- JCA

The SCA specification at the moment also allows specification of service component interfaces (and hence the corresponding “Interface Types”) in two languages:

- WSDL PortType
- Java interfaces

The concept of component type is fundamental in SCA. SCA can introspect a component implementation and discern its component type (service interface, reference, and other configurable properties) provided the SCA runtime natively supports the corresponding implementation type. In case SCA can't verify the implementation through introspection, it can use an accompanying component-type file (that has to be provided by the implementation supplier) to discern the “intentions” of the component – services, interfaces, dependencies, and properties.

Extensibility Mechanism of SCA

The extensibility mechanism in SCA is extremely powerful and

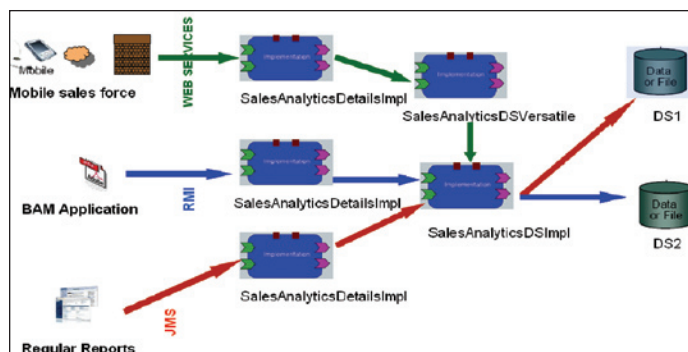


Figure 1 The architecture of a multi-channel sales analytics service component-based composite application.

allows custom support for new interface types, implementation types, and binding types. However, until they're standardized it would remain proprietary, limiting true interoperability. In any case, one can imagine that within the boundaries of the same enterprise and even among a limited set of business partners, these extensions could be standardized. For example, in case of Apache Tuscany, a popular SCA platform, the steps for adding an implementation type would involve implementing a few interfaces having lifecycle methods for the Tuscany SCA runtime to invoke and activate these extension modules as well as the standard SCA. Moreover, SCA does not require creation of new rules for implementation and binding platforms. It leverages already existing standardization efforts in these diverse platforms. For example, it builds on the Java Annotations concept prevalent in the Java EE platform to specify SCA-specific annotations. A Java developer conversant with the Java annotations concept and style of development would find it easy to adopt SCA annotations for the Java platform.

SCDL Compared to WSDL

WSDL, like SCDL, is a service interface description language for Web Services, hence there's a chance for confusing it with SCDL. While WSDL provides an interface for Web Services, WSDL isn't at all concerned with the details of service implementations other than providing the concrete endpoint where the service is actually accessible. SCDL addresses the broader requirement of configuring and wiring heterogeneous implementations together (with different kinds of interfaces and wiring mechanisms), Web Services being just one of the interface types.

For example, one could have a C++ implementation with a CORBA IDL interface description. It may be desirable to configure the C++ implementation with different database connectivity protocols and make this configured implementation (a component in SCA) available to a Java implementation. The Java service component could talk to the C++ component using a Web Services protocol. The C++ could even be referred to by a COBOL implementation. The requirement here then is to configure and wire together these diverse implementations without opening the implementations up. The runtime on which these components would execute would have to address multiple challenges – it would have to introspect these implementations (C++, COBOL, Java), find out if the implementation exposed the promised and published interfaces (CORBA IDL, Web Service interface), if these implementations can actually be properly configured (database connectivity), if the source and target components interfaces are compatible, and finally if the protocol binding the interfaces demand can be made available between these components. An SCA runtime would address these challenges exactly.

SCA enables the description of component-to-component interdependency. It specifies the component architecture, which WSDL isn't concerned with. Secondly, SCA promises to provide tremendous flexibility in being able to specify, configure, and wire components together – implementation-to-implementation flexibility (varying with properties), implementation-to-interface flexibility, and interface-to-binding flexibility.

Listing 1 SCDL file

```
<?xml version="1.0" encoding="UTF-8" ?>
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0" targetNamespace="http://SAService" xmlns:hw="http://SAService" name="salesanalyticsdetails">
  <!-- Service exposed with SOAP/HTTP (Web services) binding -->
  <service name="SalesAnalyticsSvc" promote="SalesAnalyticsServiceComponent" requires="authentication">
    <interface.wsdl interface="http://salesanalyticsdetails#wsdl.interface(SalesAnalyticsDetails)" />
    <binding.ws uri="http://localhost:8090/SalesAnalyticsSvc" />
  </service>
  <!-- Service exposed with RMI binding -->
  <service name="SalesAnalyticsRMISvc" promote="SalesAnalyticsServiceComponent">
    <interface.java interface="com.sm.server.SalesAnalyticsDetailsIntf" />
    <binding.rmi host="localhost" port="8100" serviceName="SalesAnalyticsRMISvc" />
  </service>
  <!-- Service exposed with SOAP over JMS binding -->
  <service name="SalesAnalyticsJMSvc" promote="SalesAnalyticsServiceComponent">
    <interface.wsdl interface="http://salesanalyticsdetails#wsdl.interface(SalesAnalyticsDetails)" />
    <binding.ws wsdlElement="http://salesanalyticsdetails#wsdl.binding(SalesAnalyticsDetailsSoapJmsBinding)" uri="jms:/queue.sacrete?
      transport.jms.ConnectionFactoryJNDIName=QueueConnectionFactory&
      java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory& java.naming.provider.url=tcp://localhost:61616" />
  </service>
  <component name="SalesAnalyticsServiceComponent">
    <implementation.java class="com.sm.server.SalesAnalyticsDetailsImpl" />
    <reference name="salesAnalyticsDSV" target="SalesAnalyticsDSVersatile" />
  </component>
  <!-- The mobile framework intervening service component. It can be configured for normal or mobile devices -->
  <component name="SalesAnalyticsDSVersatile">
    <implementation.java class="com.sm.server.SalesAnalyticsDSVersatile" />
    <reference name="SalesAnalyticsDS" target="SalesAnalyticsDS1" />
    <property name="Device">Mobile</property>
  </component>
  <!-- The same SalesAnalyticsDSImpl could be configured to use different data sources (in this example, file systems for simplicity) -->
  <component name="SalesAnalyticsDS1">
    <implementation.java class="com.sm.server.SalesAnalyticsDSImpl" />
    <property name="DataSource">D:\Research\Datafile1.txt</property>
  </component>
  <component name="SalesAnalyticsDS2">
    <implementation.java class="com.sm.server.SalesAnalyticsDSImpl" />
    <property name="DataSource">D:\Research\Datafile2.txt</property>
  </component>
</composite>
```

Business Use Case

Let's consider a sales analytics application (Figure 1) that supplies analytical data on sales history to other enterprise applications that consume this information to generate regular reports for managerial decision making, implement BAM, and provide field sales personnel with the latest sales reports. This requires data input with different degrees of recency and latency. The regular reports require weekly inputs in batch mode and hence the sales analytics application could supply the data asynchronously over JMS. The BAM application requires real-time updates and hence would require data to be provided over a synchronous RMI binding. The field sales personnel would have to be provided with daily updates on their devices. Hence, the data has to be formatted from the sales analytic application for mobile device form factors and would have to be provided over the Internet with enhanced security levels. The enterprise decides to use Web Services over a secured HTTP protocol to do this. Additionally, to enable mobile form factors the existing sales analytics application would have to be front-ended with another component from a deployed mobile-enabling

framework. The architecture of this setup is depicted in Figure 1. The sales analytics component exposing the service functionality is written in Java.

The same Java component `SalesAnalyticsDetailsImpl` (Listings 1 and 2) has been configured to use different data sources (having different data recency – `SalesAnalyticsDS1`, `SalesAnalyticsDS2`) and then is exposed as service components `SaleAnalyticsServiceComponent` using different binding protocols – RMI, JMS, Web Services. There could be an intervening mobile-enabling framework service component `SalesAnalyticsDSVersatile` between `SalesAnalyticsServiceComponent` and `SalesAnalyticsDS1` (or `SalesAnalyticsDS2`) that could supply mobile-formatted data from the data sources. The SCDL file providing this entire composite configuration on open source Apache Tuscany is shown in Listing 1.

The Java interface and implementation contain no references to individual binding details or security details. These are taken care of by the SCA runtime through the SCDL configuration. The Java interface and implementation files are shown in Listings 2 and 3.

Listing 2 Java interface `SalesAnalyticsDetailsIntf`

```
package com.sm.server;

/**
 *
 * @author XYZ
 */
import org.osoa.sca.annotations.Remoteable;
import org.osoa.sca.annotations.Service;

/**
 * This is the business interface of the SalesAnalyticsDetails service.
 */
@Remoteable
@Service
public interface SalesAnalyticsDetailsIntf {

    public String getSalesAnalyticsDetails(String period, String productID);
}
```

Listing 3: Java implementation `SalesAnalyticsDetailsImpl`

```
package com.sm.server;

/**
 *
 * @author XYZ
 */
import org.osoa.sca.annotations.Property;
import org.osoa.sca.annotations.Reference;
import org.osoa.sca.annotations.Service;

@Service(SalesAnalyticsDetailsIntf.class)
public class SalesAnalyticsDetailsImpl implements SalesAnalyticsDetailsIntf {

    @Property
    public String DataSource = null;

    private SalesAnalyticsDSV salesAnalyticsDSV;

    public String getSalesAnalyticsDetails(String period, String productID) {
        String details = null;
        //business logic below
        //...
        //...
        return details;
    }

    public String getDataSource() {
        return DataSource;
    }

    @Reference
    public void setSalesAnalyticsDSV(SalesAnalyticsDSV salesAnalyticsDSV) {
        this.salesAnalyticsDSV = salesAnalyticsDSV;
    }
}
```

Conclusion

Composite applications are the emerging face of SOA for developing IT applications. SCA is a good starting point for standardizing the necessary infrastructure for realizing true flexibility in composite application development. The true power of SCA lies in providing flexibility at multiple levels: the implementation level, interface level, and binding level. Further, as more and more implementation types and binding types get supported by SCA, the true power of composite applications will be unleashed with mainstream penetration.

References

- *Service component architecture specifications:* <http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications>.
- *Apache Tuscany:* <http://incubator.apache.org/tuscany/>.
- *Tuscany: Adding a New Implementation Type:* <http://incubator.apache.org/tuscany/mike-edwards-ramble-through-adding-a-new-implementation-type.html>.
- *An overview of Service Component Architecture:* <http://www.ibm.com/developerworks/webservices/library/ws-soa-scadev1/>.

About the Authors

Dr. Sudeep Mallick is a senior technical architect and researcher with SOA Center of Excellence, SETLabs, the R&D arm of Infosys Technologies, Ltd. He has been published in numerous international journals and has spoken at conferences on service-oriented computing, software engineering, and enterprise architecture. He is also the author of a book on enterprise IT architecture. His current areas of interest include service-oriented software engineering, SOA governance, and composite applications.

sudeepm@infosys.com

Ashwini Kumar Jeksani is a software engineer working for SOA Center of Excellence, SETLabs, the R & D wing of Infosys Technologies, Ltd. His areas of interests are SOA-enabling technologies such as ESB, composite applications, SCA and Web Services. His current area of work is on SAP enterprise SOA (eSOA).

Ashwini_Jeksani@infosys.com

Choosing the Right Middleware Stack for Your SOA in Healthcare

Consider the non-functional requirements

BY SANJAYA KARUNASENA



When there are lots of solutions, choosing the right products to build your software is really challenging. Identifying the right combination of middleware products to build your SOA is key to its success. This article discusses an approach to using non-functional requirements in choosing these products, using SOA in healthcare as an example.

Software applications that are fragmented across departments, legacy systems that cannot be retired, and integration challenges due to heterogeneity are no exception in the healthcare industry. Business processes and sophisticated medical data that go from physicians and hospitals to health plans to pharmaceutical companies to insurance companies make the implementation of IT solutions highly challenging. Service Oriented Architecture (SOA) has been identified as a perfect way to meet these challenges and many companies in the healthcare business are moving towards SOA. Choosing SOA as the architectural style is just the first step in implementing a successful enterprise solution. Employing the right products in your SOA middleware stack is another key step in this process.

When architecting a solution, identifying key non-functional requirements is very important. A good enterprise architect always considers the non-functional requirements in making important architectural decisions. There are many non-functional requirements and it's impossible to satisfy all of them to the fullest extent. Business requirements play an important role in prioritizing non-functional requirements. An architectural analysis technique like ATAM could be very useful here. These key non-functional requirements play a major role in choosing which middleware products is used in your solution.

Non-Functional Requirements

Listed below are a few important non-functional requirements in the healthcare industry that need to be considered when evaluating products to be used in your SOA middleware stack.

Security – The confidentiality, integrity, and availability of patient data has to be maintained in creating, receiving, maintaining, and transmitting. You may want to use proprietary access controls and authentication protocols to achieve high security. The chosen products should provide extension points to plug in to these proprietary solutions.

The application of security policies and managing access control at each service level is important. Every request must be audited

and audit trails have to be maintained so, when an incident occurs, its cause can be identified without failure.

High Availability – In an emergency, not having some important data available for few minutes could cost a patient his life. So the high availability of critical systems is a must.

To provide high availability, the service container and the associated products should support clustering and load balancing.

Reliability – The level of fault tolerance, fault detection, and recovery plus the responses to unexpected failure modes provided by the middleware should be evaluated. When the middleware is clustered, it could enforce certain design constraint on your services. For example, to what extent does the state information get replicated across the cluster? Some of these constraints could make your functional design overcomplicated and difficult to maintain.

Another aspect to look at is whether the middleware can be used to make your legacy applications more reliable. One of our clients has a mainframe with a hard limit on concurrent connections. We make use of WSO2's ESB to expose the mainframe services as Web Services, while throttling connections to the mainframe across the cluster.

Performance – Performance can be looked at in two different ways: the performance of the middleware and the level of support the middleware offers to boost your services' performance.

A mistake made by most of the developers is to think that they can consume the entire response time stated in the requirements document for a use case when the actual response time available to the developers is the response time in the use case less the message transmission time plus the message processing time of the middleware.

The ability to cache data in the middleware layer is important. How well the middleware can manage its cache across the cluster is an important factor to consider. This caching capability can be used to improve the performance of legacy applications as well.

Interoperability – The key to interoperability is to adopt open standards. The most widely used and accepted open standards in the industry are from the Web Services community. Therefore, it makes sense to choose middleware based on Web Services technologies when it comes to implementing interoperable services.

Integration with Other Systems – There will be requirements to integrate with mainframes, databases, and other internal applications. The middleware that can operate across multiple transports (message/file transmission protocols) to send and receive mes-

sages is important when evaluating integration options. HTTP, JMS, SMTP, XMPP, RMI/IIOP, and (S)FTP are popular message/file transmission protocols.

Cost – How much of your IT budget can you allocate for third-party software is also an important aspect to look at. However, in the Web Services spaces, it's evident that some of the free and open source solutions are actually better than their proprietary counterparts. So you shouldn't hesitate to give serious consideration to FOSS in the enterprise. Most of these software products are backed by organizations that provide development and production support.

SOA Governance – The importance of SOA governance is becoming evident with the growth of SOA applications in the enterprise. Having the ability to effectively govern the enterprise software is critical to the long-term success of SOA. Maintaining quality, performance, and the applicability of services, consistency in discovery and reuse, managing versions, assessing and managing changes are some of the key disciplines.

A registry toolkit that can be associated with the middleware products is required to satisfy this need. Such a product should be able to store, tag, comment, link, find, and organize the service descriptions and related artifacts. It should be equipped with a well-defined security model and permissions.

Sample Solution

Figure 1 illustrates a solution that we have proposed as a SAO

middleware stack for a healthcare client.

The following is a list of key products we have used to construct the stack and the feature in each product that helps satisfy non-functional requirements.

WSO2 ESB

- Caching
- Logging and Auditing
- Statistics and Management
- Security Control
- Connectivity (SOAP, REST, SFTP)
- Transport (HTTP/JMS/SMTP)
- Interface (WSDL/Schema/Policy)
- Message format (SOAP/POX)
- QoS (WSSecurity/RM) and optimization switching (MTOM/SwA)
- Non-blocking http(s) transport for ultra-fast execution and support for large numbers of connections
- Integrated Registry/Repository, facilitating dynamic updating and reloading of configuration and resources
- Load-balancing/Failover and Throttling support
- Highly flexible, supports XSLT, XPath, Java, Ruby, JavaScript configuration models
- Non-blocking IO and streaming XML
- Scales to support thousands of concurrent connections
- Transformation (XSLT) at twice the rate of a leading proprietary competitors

A blue-themed advertisement for Certeon. The background features a stylized globe and abstract digital lines. In the foreground, three white silhouettes of people are shown from the waist up, reaching up to support the globe. The Certeon logo is prominently displayed in the center-left. Below the logo, the text 'Acceleration • Virtualization • Manageability' is written in a serif font. Further down, a promotional message invites visitors to the 2008 Virtualization Conference & Expo in New York City. The website address 'www.certeon.com' is at the bottom left.

certeon®

Acceleration • Virtualization • Manageability

Come see what's new at the Certeon booth.
Certeon is a proud sponsor of the
2008 Virtualization Conference & Expo
June 23 – 24, 2008 • New York City, NY

www.certeon.com

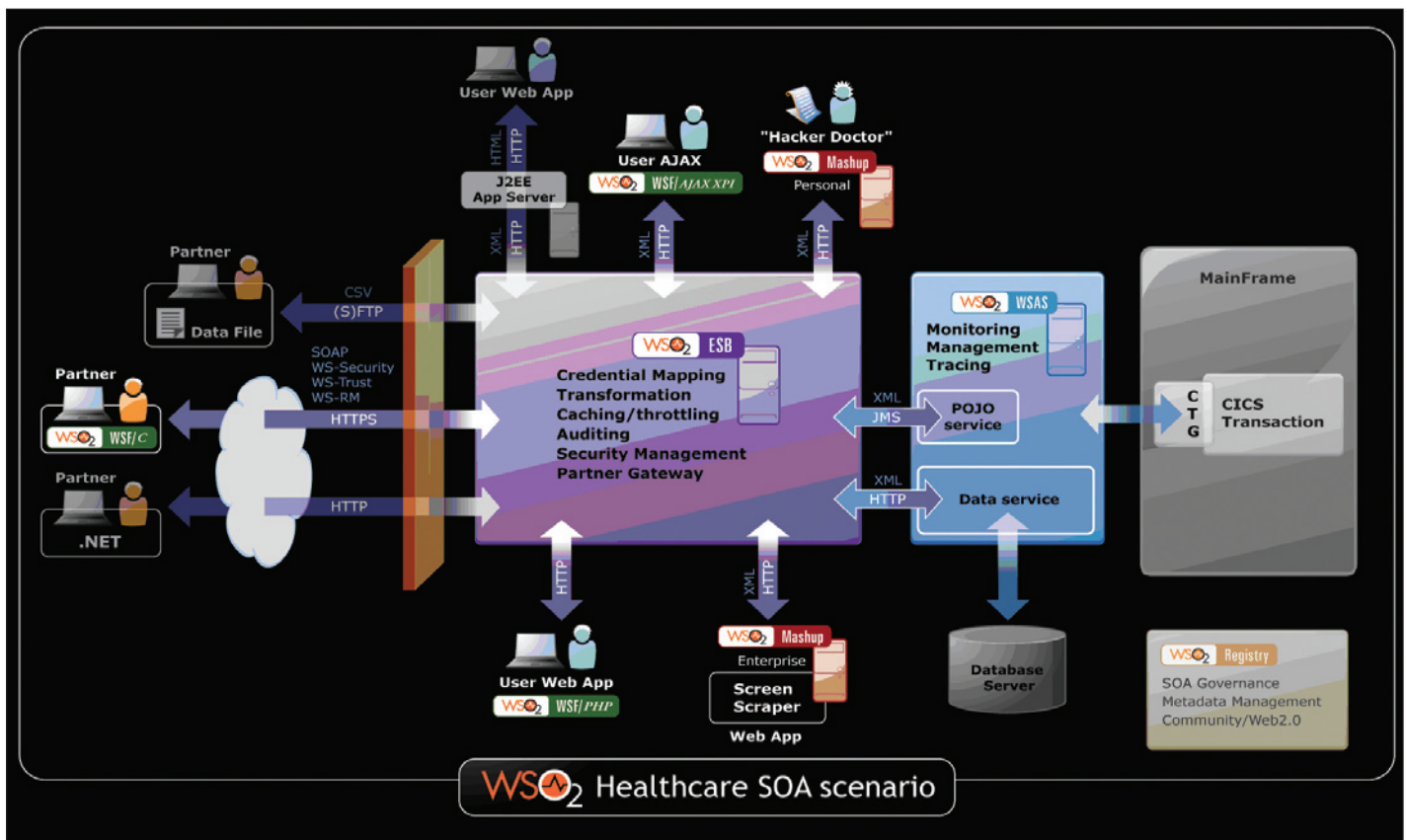


Figure 1

WSAS

- Very good support for SOA and WS-*
- SOAP 1.1, 1.2, MTOM, WSDL 1.1, 2.0
- WS-Security, SecureConversation
- WS-Trust
- Clusterable, highly available
- Built-in test tools (TryIt), point-and-click security configuration
- Multiple service implementation types (JSR181, POJO, Axis2 AAR, Axis1, Data Services)
- Ability to expose SQL queries as services using WSO2 Data Services
- Support for gracefully shutting down the transports
- Version controlled configurations via SVN

WSO2 Mashup Server

- Enable doctors to code using services
- Properly authenticated and authorized
- Services can be combined, orchestrated. and mashed up

WSO2 Registry

- SOA Governance
- Store, tag, comment, link, find, and organize the important artifacts in the enterprise
- Well-defined security model and permissions
- Each department can manage its own domains in the wider SOA
- Bridge the Social SOA (users, business domain owners) with the Technical SOA (developers, systems, clusters)
- Manage policies effectively

- REST-based model ensures wide compatibility with any HTTP client
- Full Web UI with a structured, smart, wiki-like experience, plus AtomPub remote API
- Full version control, rollback
- Dependency management
- Understanding the links between resources
- Built-in user management, permissions and security

Conclusion

Making use of Web Service technologies is the most widely used approach in implementing a SOA. Selecting the right Web Services middleware plays an important role in making your SOA dream a reality. Non-functional requirements that play a major role in many architectural decisions should be used in deciding which middleware products to use.

References

- [1] Architectural Trade off Analysis Method: http://www.sei.cmu.edu/architecture/ata_method.html
- [2] Medical Connectivity Consulting, Improving patient flow through technology: <http://medicalconnectivity.com/2007/07/03.html>

About the Author

Sanjaya Karunasena is a senior software architect with WSO2, Inc.

sanjayak@wso2.com



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

GO Service Ownership Framework

Capturing the aspects of SOA service ownership

BY ANBARASU KRISHNASWAMY



As the field of service-oriented architecture (SOA) evolves, it brings interesting challenges that should be addressed in order to drive its adoption and realize the benefits it has been promising. It took a while for many to understand that SOA is not purely a technology issue.

SOA is an IT strategy and it requires a shift in the way we think about how IT is aligned with business and how an organization can support and drive SOA within the enterprise. The popular IS Strategy Triangle framework emphasizes the need for alignment between business, IT and organizational strategies.¹ Figure 1 shows the extension of the IS Strategy Triangle and how SOA fits in. Business strategy should drive the SOA strategy, which is an IS/IT Strategy and SOA strategy must support the business strategy.

In order for the SOA strategy to be effective, the organizational capabilities and processes should be aligned with the business and SOA strategies. A number of surveys have identified SOA organization and governance-related issues as a top inhibitor for SOA adoption.

Effective adoption of SOA requires changes to the traditional structure and processes of an organization. The implementation of SOA strategy might introduce new organizations, governance bodies, roles, responsibilities, and other governance issues. It may change the way business capabilities are built and rolled out to the business. These changes introduce challenges around the ownership of assets. The purpose of this article is to develop a framework for identifying ownership issues around SOA assets. Every organization's culture and organizational capabilities are different. The framework discussed in this article is generic enough to be customized for any organization.

The effectiveness of the governance processes depend very much on the definition of roles, the clarity around the ownership of assets and processes, and the degree of empowerment of the governing bodies. SOA introduces several challenges around the ownership of SOA assets for a couple of reasons:

1. Services are built to address enterprise concerns. Individual business units no longer own all parts of a service, as doing so would completely defeat the purpose and benefits of SOA. The decisions made around SOA should be based on the best interests of the whole enterprise, not the interests of one particular group within the enterprise.

2. Services are made up of several parts that are generally distributed in nature. This distribution is multi-dimensional. For example, a service may span physical (deployed across several hardware layers) and organizational boundaries (different organizations may be involved in the development and deployment of the service).

When determining the ownership of SOA services, it's important to clarify the purpose of identifying ownership, and to understand the underlying assets exposed and the boundaries around Services.

Purpose of Identifying Ownership

The purpose of identifying ownership is multifold. Every organization has different reasons or objectives as to why it should identify the owners. These objectives drive how ownership is defined and shared across organizational entities.

In general, the common reasons for identifying ownership are listed below.

Single Point of Accountability (SPOA)

Ownership identifies a single point of accountability (SPOA). Property rights have been identified as the key to economic development and property rights lead to prosperity.² The property rights within an organization play a critical role in leading it to prosperity. O'Driscoll's article refers to the following passage from the work of Mises.

Ownership of the means of production is not a privilege, but a social liability. Capitalists and landowners are compelled to employ their property for the best possible satisfaction of the consumers. If they are slow and inept in the performance of their duties, they are penalized by losses. If they do not learn the lesson and do not reform their conduct of affairs, they lose their wealth. No investment is safe forever.³

This observation applies perfectly to SOA as well. The owners of service assets must ensure that these assets serve in the best interest of the consumers and the enterprise. Owners are expected to be held accountable for the part they own. Without a sense of ownership, the organization can not achieve a state of prosperity. As organizations achieve higher levels of SOA maturity, they would most likely have a "business-like" model where providers of a Service serve consumers and are measured on the value they create through the services. Owners must understand that they have a liability to the enterprise and must strive to serve and satisfy the consumer. Failing to do so would diminish the value they provide to the enterprise, raising questions about their existence.

Subject Matter Expertise

Ownership provides Subject Matter Expertise (SME) to define, design, develop and maintain the services. Owners are typically

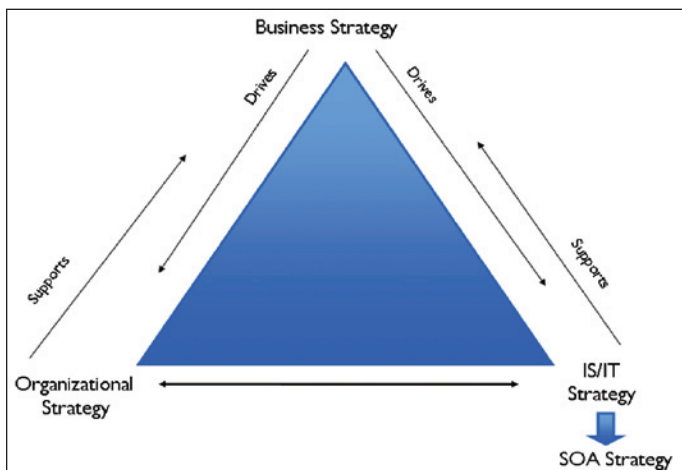


Figure 1 IS Strategy Triangle and SOA

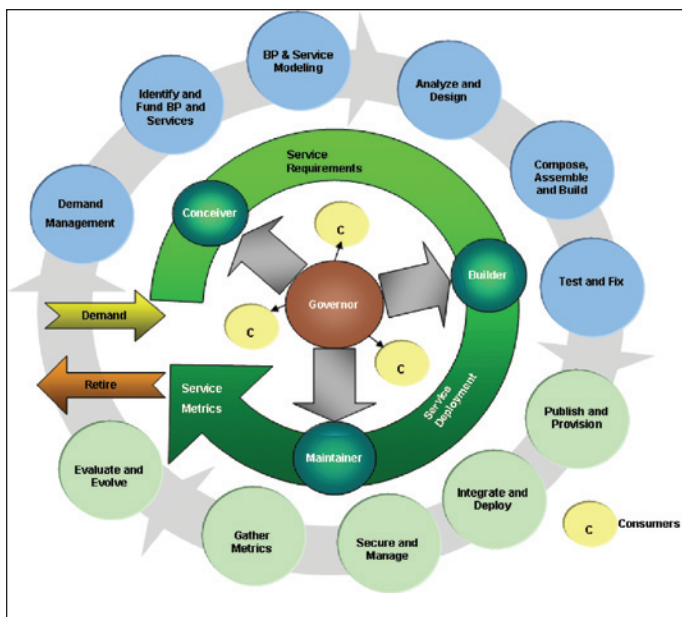


Figure 2: GO (Governor Owner) Service Ownership Framework

experts in some or all parts of the assets. For example, some may be experts in the functional area for which the capability is built, while some may be experts in the technologies used to build the business or technical capabilities.

Asset Protection and Evolution

Services need to be protected and promoted for the benefit of the enterprise and the owner(s) plays a key role in that. The protection of assets really means evangelizing and promoting the reuse of it in order to realize business value. If the assets are not protected, duplicate capabilities will be built, eventually forcing them to retire. The owner(s) typically protects the services to ensure that they can cater to the consumer and evolve them to support the changing business needs.

Funding

Ownership in most cases clarifies the source or use of funding.

Owners either fund or recover the cost of building a service, although there are occasions when other consumers may pitch in. In light of the enterprise value added by the service, a funding model should be developed to assist the owners with managing the cost of building and running the service.

Change Control

The owner(s) is responsible for controlling any changes to the service and generally has the authority and knowledge to validate changes to the service. They ensure that the changes cause improvement without affecting the semantics of the service.

GO (Governor-Owner) Service Ownership Framework

Services are complex entities that span across organizational and technical boundaries. The distinct technical and process components of the service should be identified and assigned ownership appropriately for the reasons stated earlier. A single owner model will not work for SOA services. One entity or group may not be able to own and govern the services due to the distributed nature of service components and life-cycle processes. It's necessary to distinguish the role of owner(s) from that of governor, which is discussed later.

From the time IT demand is created for a business capability to the time a service is deployed to provide that business capability, the service goes through several stages. During this development process, different groups perform the tasks required to evolve the service to the next level. The service could have different owners at different points in this life cycle. The coexistence of different versions of the same service introduces additional complexities. A great deal of coordination is required to ensure a smooth hand-off between these organizational entities. Also, there are other stakeholders in the process that need to be involved at the right times. This warrants the need for a governing body that would independently orchestrate these efforts. We call this governing body the "governor" in this framework.

Figure 2 explains the service ownership using the GO service ownership framework. As IT demand is generated, concepts are prioritized and appropriated for implementation. As part of the concepts and requirements phase, services and business processes are identified and modeled. These activities are typically performed by the business unit. Then they go through the design, construction, and testing phases. A service provider typically performs these tasks to build the services. Then the service begins its operational life cycle, where it is deployed, managed, and monitored. The metrics gathered are fed back to the business to improve the services further or to generate new ideas. As new versions of the services evolve, the old services will be retired.

The various components of the service should be considered for the purpose of ownership. You need to ask a few questions with respect to these components. Should these components be owned by the same entity or does it make sense to have different units own different parts of the service? Who has the final say on the design decisions around these?

Service Contract

- The contract includes the business requirements or semantics of the service and non-functional requirements.

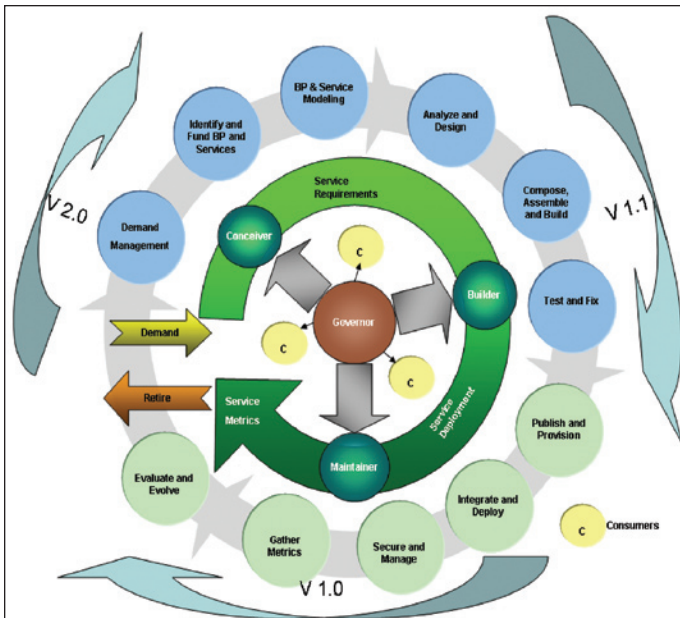


Figure 3: Co-existence of service versions

Service Interface

- Interfaces are the entry points to the services. The operational infrastructure exposes the interfaces for access by the consumers.

Service Implementation

- Implementation of the service may include the following
 - Underlying assets
 - Business (or utility) logic implementation
 - Policies
 - SOA infrastructure components like registry and ESB
 - Physical infrastructure such as the hardware servers, network components etc...

We need to make sure that each one of the components has rightful owner(s) identified. The GO Framework breaks service ownership down further into the following ownership types to address the ownership needs of these components:

- Semantic Ownership (Conceiver)
- Development Ownership (Builder)
- Operational Ownership (Maintainer)

Each of these ownership types are discussed in detail below.

Semantic Ownership (Conceiver)

The lifeblood of the service is its business (or technical) semantics. The semantics include the meaning of the underlying assets, information, business rules, or algorithms. Semantic ownership of the service typically lies with the owner of the underlying asset. The business (or the business representative) typically holds this ownership for business services. The business is in the best position to judge the impact of any semantic changes or enhancements. Since this owner conceives of the service, we will call it the Conceiver. In the case of shared services that are used by multiple business units, identifying the rightful owner may not be easy. In such cases, the governor would need to determine the right semantic owner under

the direction of the SOA leadership team or steering committee. In certain cases, related services can be grouped into portfolios of functional categories. These services would typically have an underlying semantic relationship and the same semantic owner. In most cases, semantic owners are the initial consumers of the service as well as the ones in need of the service functionality.

Development Ownership (Builder)

Development ownership refers to the responsibility of designing, developing, and provisioning the services. The development owner is responsible for the technical aspects of the service that include the following:

- Service interface definitions
- Schema definitions
- Technical implementation
- Policies
- Service architecture
- Implementation of nonfunctional characteristics

The development owner can modify the nonfunctional characteristics of a service without changing the semantics of the underlying assets.

Service providers would usually fill this role. They work with the semantic owners to understand the service requirements and implement the service. In some cases, they may need to work with a different group to implement all or part of the business or access logic. Since this owner builds the service, we will call it the Builder.

Operational Ownership (Maintainer)

The operational owner is responsible for maintaining the service in the operational life cycle of the service. The operational owner owns the “runtime instances” of the services. Changes to the runtime instance and configuration would be controlled by this owner. The development owner typically hands over the service to the operational owner when the service is ready to be transitioned from the development life cycle to the operational life cycle. The operational metrics of the services are gathered and provided back to the business for continuously improving business capabilities. Since the operational owner is responsible for maintaining the services, we will call it the Maintainer.

Governor

Service ownership can not be discussed without talking about the role of the governor, which is typically an organizational role. A centralized organizational body should be assigned these responsibilities to protect the interests of the enterprise. The governor role has the following responsibilities.

- **Governance:** The governor should handle the governance and coordination among the various stakeholders and owners. As discussed earlier, a service changes hands many times throughout its life cycle. The development and release of the service should be synchronized with the development and release schedules of the consumer as well. On top of that, it should be ensured that the service follows enterprise standards and best practices. The governing body is responsible for these activities and it may leverage other groups within the organization to execute these.
- **Promote Reuse:** Sometimes, owners may not be interested in promoting service reuse as they may be measured on delivery but

not on enterprise value or cost savings. It's the responsibility of the governor to promote service reuse and mediate between parties to accelerate the adoption of the services.

- **Brokering and Mediation:** Owners own the service and have the final say on the part they own. The governor brokers the requests between consumers and other parties for access, changes, and enhancements. In a complex enterprise ecosystem, conflicts are inevitable. The priorities of stakeholders could be very different and the governor brokers the communication between the various parties.
- **Funding Management:** The governor is generally responsible for administering the funds for building and running the services. The funding model should support the allocation of funding for services and chargeback to recover costs. The governing body could run as a cost center or investment center. In a cost center model, it's responsible only for the costs and recovers costs through chargeback to the consumers. In an investment center model, it earns revenue through capital budgeting and chargeback and reinvests into building additional enterprise services at its discretion.
- **Cost Allocation:** The governor ensures that the cost of development and the maintenance of services are allocated appropriately to the consumers of the service. To measure and motivate the performance of the firm, the costs should be measured and allocated equitably. Failure to do so would result in the misuse of valuable resources.
- **Release Management:** The governor also manages the development and release cycles of the various versions of the service with the help of the portfolio architect. As shared services are being developed and enhanced actively, it's not uncommon for multiple versions to coexist. In such cases, different service versions may be owned by different owners at any point. Figure 3 explains this concept where v1.0 is in production and being monitored and managed by the operations, while v1.1 is being implemented by the development owners (service provider). At the same time, the semantic owner may be working on fleshing out the requirements to create a newer major version of the service (v2.0). The governor coordinates the release and retirement of these versions among the owners and consumers.
- **Continued support of services:** The governor measures and understands the value of the services and should ensure that the services are supported by the appropriate owners until they are retired from use.

Now that the framework has been explained, a useful exercise would be to identify the various owners for your services. A good starting point would be to identify the respective owners for each category of services. Table 1 lists the standard categories of services. Write down the owners for these service types in your organization. Also, it is a good time to start thinking about which group or groups will perform the governor duties in your organization. If you can't determine which group will perform the governor role, then it could potentially be a current gap.

Summary

Effective SOA governance depends on the identification and assignment of ownership of various components of the service. During the life cycle of a service, it requires that the semantic, development, and operational owners go from concept to production. Coordination between various owners and other stakeholders

Service Category	Semantic Owner	Development Owner	Operational Owner
Business Service			
Shared Business Service			
Common Security Services <ul style="list-style-type: none"> • Auditing • Encryption / Decryption • Transaction Identification • Trust • User and Password Maintenance 			
Corporate Security Services			
Audit Services			
Logging Services			
Data Access Services			
Information Access Services			
Utility Services (Common Technical Services) <ul style="list-style-type: none"> • Logging • Error Handling • Notification • Publish & Subscribe • Messaging • Scheduling • Versioning • Discovery 			
Monitoring Services – Business Metrics			
Monitoring Services – Operational Metrics			
Monitoring Services – Usage Metrics			
Presentation Services			
Connectivity Services			
Error Handling – Business Exception			
Error Handling – Technical Exception			

Table: 1

would require the involvement of the governor. The GO (Governor-Owner) Framework captures these elements of service ownership.

References

1. Pearlson, Keri E., Saunders, Carol S. Managing and Using Information Systems: A Strategic Approach. Third Edition. John Wiley & Sons. ISBN: 0-471-71538-7.
2. O'Driscoll, Gerald P. & Hoskins, Lee. (2003). "Property Rights: The Key to Economic Development." Policy Analysis. No. 482.
3. von Mises, Ludwig. (1966). Human Action: A Treatise on Economics, 3rd rev. ed. Chicago: Henry Regnery Company. pp. 311–12. ■

About the Author

Anbarasu Krishnaswamy has over 15 years of IT industry experience, nine of which were with BEA. In his current role as the Enterprise Architect Lead, he leads the enterprise architecture and SOA practices for the central region professional services at BEA. As a SOA practitioner, he has helped several customers with SOA transformation and implementation. His experience also includes design and development of Java/J2EE applications, client/server computing, Web development, and enterprise application integration (EAI). Anbarasu holds a MBA from NIU and an MS in computer science and engineering.

anbarasu@bea.com



The Unreliable Internet

When you need reliable messaging and how to use it

BY MICHAEL GALPIN



The Internet's a dangerous place for a message. Component failures, network connection issues, and other problems can prevent a message from being delivered. Fortunately, there's WS-ReliableMessaging, which makes sure messages get through. This article explains how to use reliable messaging, why you should use it, and how to use it with WSO2's Web Services Application Server (WSO2

WSAS) 2.1.

What Is Reliable Messaging?

Reliable messaging may seem like a dull thing. We're used to being able to rely on the Internet, so why worry so much about increasing

the reliability of something that's already reliable. But is the Internet really that reliable? How many times have you clicked on a hyper-link that didn't work at first and you had to click it again or hit the refresh button? It's something that happens so often that we don't think twice about it. It's easy to ignore when you have a high-speed connection.

You can't ignore the unreliable nature of the Internet when building systems that rely on the exchange of data and messages with other systems. You have to deal with this unreliability. That's where reliable messaging comes in.

When Do You Need It?

So how do you overcome the unreliability of the Internet? Well you could start by paying attention to the HTTP status code you

get when you send a message. A status of 200 means the message was delivered. But can you be sure that the message was processed, not just delivered? You need more than just a status code; you need an acknowledgement to be sure your message went through properly.

And what happens if you don't get the acknowledgement? Clearly you have to pick some amount of time to wait and then retry if you don't get the acknowledgement. Now what happens if the other server was just slow, so your message actually went through, but you sent it again? What if your message is a transaction of some sort? Sending it twice could cause serious problems. You could build logic into your application to deal with this, but wouldn't it be nice if you didn't have to?

The WS-ReliableMessaging (WSRM) Protocol

Fortunately there's a standardized protocol for handling and solving all of the simple scenarios above and even more complicated scenarios. It's called WS-ReliableMessaging or WSRM for short. It's important that this not only solves the problem, but that it's standardized. Even in the simple scenarios described above, there has to be some mutual understanding between the message sender and message receiver for any kind of reliability protocol to work.

Both sides have to understand and implement complementary components of a unified strategy. If you control both sides of a conversation, you can implement any unified strategy that you want. However, there's probably a lot of situations where you only control one side of a conversation. That's when a standardized solution is needed. It's the only way to ensure interoperability. It's the interoperability from standardized solutions that have made Web Service protocols such as SOAP so successful. WSRM is in the same vein.

Numerous infrastructure vendors such as IBM, Microsoft, and Oracle have standardized WSRM. This lets you use it for either side of a conversation and have confidence that the other side will be able to understand and communicate with you effectively and efficiently. It also means that a lot of people have reviewed the approach to make sure it works properly.

The Mechanics of WSRM

Let's take a look at a typical scenario that WSRM handles in Figure 1.

In this sequence diagram, we see the Sender needing to send two messages to the Receiver. WSRM lets the Sender specify which is the "last message" so the Receiver knows how many messages there are in the sequence. It just so happens that the first one doesn't make it but the second one does. The Receiver can see that it only got the second of the two messages that were supposed to be sent. It sends an acknowledgement that it only got Message #2. So the Sender knows to resend Message #1. Once this comes, the Receiver knows that everything has been sent. It sends a final acknowledgement and the conversation ends.

The key here is that both Sender and Receiver use a common protocol to relay metadata back and forth about the conversation. This lets them work together to ensure that the conversation is reliable, i.e., nothing is lost. Let's take another look at the variant in Figure 2.

In this case there's no acknowledgement so the Sender gets impatient and resends both messages. The Receiver then sends

an acknowledgement for both message #1 and message #2. There are several things to realize here. First, the Receiver is going to ignore the repeat (message #1). The protocol handles ignoring the repeated message, so you don't have to put anything into your application logic to deal with this scenario. You don't have to worry about it because WSRM is going to handle it for you. This is known as duplicate elimination.

Next, WSRM is smart enough to send acknowledgements for both message #1 and message #2 in a single acknowledgement message. This is an important optimization. It's easy to imagine that if the Receiver had to send two separate acknowledgements then the impatient Sender might have resent message #2 in between the acknowledgements. Of course you can imagine if there were N messages in the sequence, not just two.

As you can tell, the WSRM specification addresses a lot of complexity. You can imagine trying to deal with this complexity in your application. You might spend more time doing that than developing business logic. WSRM frees you from that by pushing that problem set down to the middleware level, where technology stacks like WSO2 WSAS implement the WSRM specification.

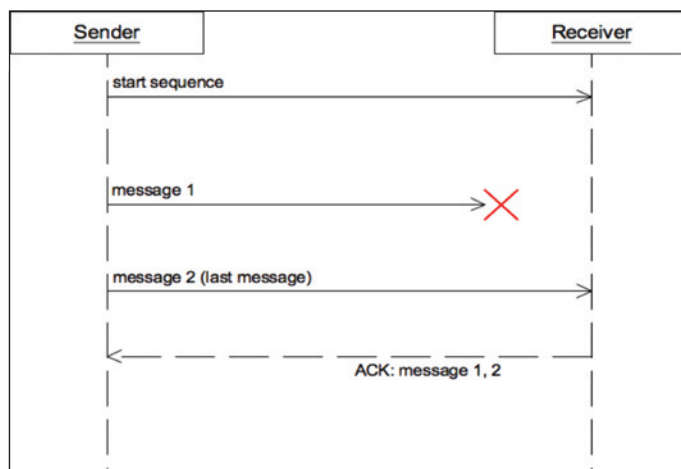


Figure 1 A sample WSRM message sequence

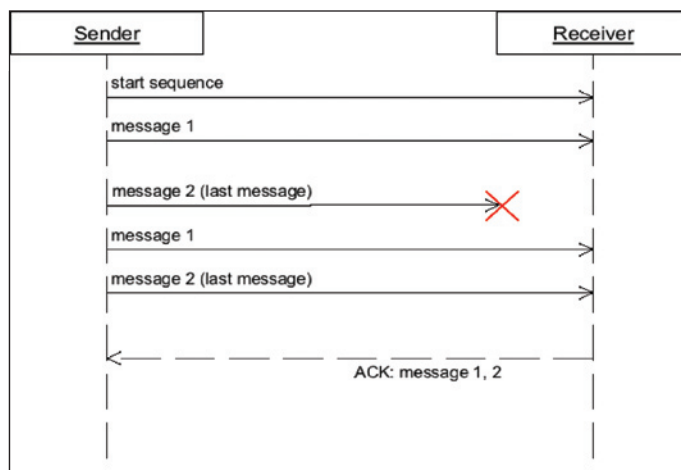


Figure 2 WSRM message sequence #2

Policy	Description
AcknowledgementInterval	This tells the receiver how long it should wait before sending an acknowledgement. A longer interval allows for more acknowledgements to be sent in a single message.
RetransmissionInterval	This specifies how long the sender will wait for an acknowledgement before resending the message.
ExponentialBackoff	This is a Boolean value used to slow down retransmissions. If it's set to true then each successive (starting with the second) will take twice as long as the previous.
MaximumRetransmissionCount	Sometimes the receiver is dead or doesn't exist, and there's nothing WSRM can do about that. This specifies how many times you want to retransmit before giving up.
InactivityTimeout	This specifies a way for the sender to realize that the sender isn't responding in the middle of a long sequence of messages. If there's been no response to previous messages in the sequence then the sender can call it quits without finishing retransmitting, etc.
InactivityTimeoutMeasure	This is simply the units of measure of the InactivityTimeout
InvokeInOrder	As we saw in the second example, messages can easily be received out of sequence. Sometimes that doesn't matter, but sometimes it does. This will tell the client whether to invoke the messages in the order sent or the order received.

Table 1 Enumeration of WSRM Policies

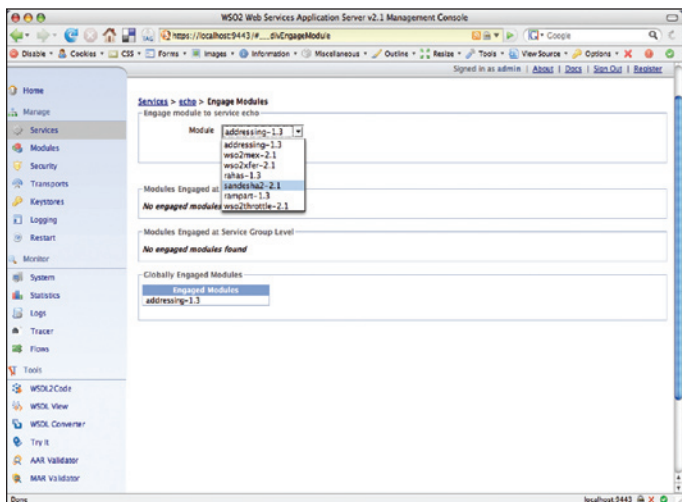


Figure 3 Enabling WSRM using WSAS Console

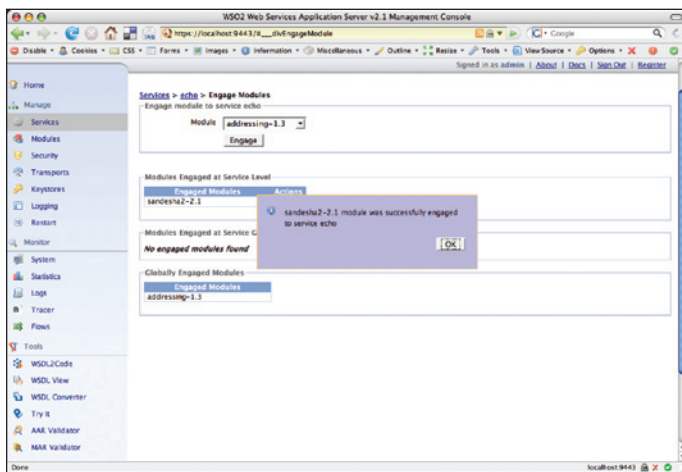


Figure 4 WSRM enabled!

WSRM: Builds on SOAP

So far we've mentioned all the great things that WSRM does. We haven't looked at the underlying WSRM implementations, but we'll get to that later. First, it's important to understand how WSRM metadata is sent back and forth between the senders and receivers in a conversation. For example, in the diagrams you see things like "message 1" and "last message." This metadata is key, but how is it sent?

The answer is SOAP headers. One of the keys of WSRM is that it builds on top of SOAP. WSRM isn't a reliable messaging solution for any message protocol; it's designed for SOAP. With that in mind let's take a look at a SOAP message that includes WSRM metadata, shown in Listing 1.

Notice all the elements with the WSRM namespace? That's the WSRM metadata. It's all contained in the header section of our SOAP message.

If you look at the header of this message, or at other SOAP messages, you'll probably notice other metadata that has nothing to do with WSRM. This metadata is being used as part of various WS-* standards. For example, Listing 1 shows an element in the WSA namespace. This element is being used to pass WS-Addressing metadata. One of the key benefits of WSRM is that it integrates with the other WS-* technologies.

WSRM: Working Together with Other WS-* Technologies

WSRM is an effective solution to the problem of creating reliable messages with SOAP. It works with the other WS-* technologies to solve numerous problems facing systems that rely on the SOAP standard to send messages and data. For example, our reliable messaging solution neglects to deal with another possible cause of messages being lost, namely network security such as firewalls, NAT, etc. You may use HTTP to get your message to a particular gateway, but that's all that HTTP can provide. WSRM isn't going to help here, but WS-Addressing will help.

Another WS-* technology that is synergistic to WSRM is WS-Policy, a generalized framework for adding extra policy information to SOAP messages so both sides of a conversation can communicate more effectively. For WSRM, there are a number of policy choices you can make, and with WS-Policy, you can make sure that everyone involved in the conversation knows about these policies. Let's take a look at some of the policies you can use with WSRM.

WSRM Policies

Table 1 shows the various policies you can set to configure WSRM for your Web Service. As mentioned above, all of these can be specified and communicated to partners using WS-Policy.

How Does WS2 WSAS Implement WSRM?

So we've seen how powerful and useful WSRM can be. However, at the end of the day, WSRM is just a specification. You need something that implements the specification. The idea behind things like SOAP, WSRM, and other WS-* technologies is to standardize message formats and protocols so that Web Service stacks/middleware can take care of the hard part for us. So what's the Web Service stack that we can use that will give us the power of WSRM?

You don't have to look too hard. The WS2 WSAS stack implements SOAP, WSRM, and numerous other WS-* technologies. It's built on best-of-breed implementations of these various technolo-



Data Services WORLD

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



13th International

2008 SOA WORLD™ CONFERENCE & EXPO

13TH INTERNATIONAL SOAWORLD CONFERENCE & EXPO 2008 EAST
June 23-24, 2008 The Roosevelt Hotel, New York, NY

14TH INTERNATIONAL SOAWORLD CONFERENCE & EXPO 2008 WEST
November 20-21, 2008 The Fairmont San Jose, CA

15TH INTERNATIONAL SOAWORLD CONFERENCE & EXPO 2009 EUROPE
January 26-27, 2009 London, England

Join Us in New York in June 2008 and Gain the Insight & Knowledge to Declare Your Company's SOA Journey a Success!

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!

Who Should Attend?

- CEOs and CTOs
- Senior Architects
- Project Managers
- Web Programmers
- Web Designers
- Technology Evangelists
- User Interface Architects
- Companies and Organizations looking to stay in front of the latest Web technologies

REGISTER TODAY AND SAVE

www.soaworld2008.com

gies to provide a powerful, easy-to-use solution for Web Service developers and consumers alike. Let's take a look at the WSRM implementation provided by WSO2 WSAS.

Sandesha + Axis2

If you're familiar with WSO2 WSAS, then you might know about its SOAP implementation. It uses Apache Axis2 to implement the SOAP specification. This leads to a natural choice for its WSRM implementation: Apache Sandesha2. Sandesha was built to work with Axis2. It's implemented as an Axis2 module.

Axis2 provides a great pluggable architecture that lets you add modules that implement extra behavior on top of SOAP. In the case of Sandesha, extra metadata and processing is done at the exit of outbound messages and at the entrance of inbound messages. This lets Sandesha implement the WSRM specification transparently to the rest of the Axis2 stack and thus transparently to the clients and servers that rely on the Axis2 stack to send or receive SOAP messages.

Besides being a full implementation of WSRM and specifically designed for Axis2, Sandesha has an impressive feature list. It has a pluggable persistence mechanism (see the section on Persistence). It provides a listener mechanism to allow customized logic to handle events like when a sequence times out. It has a powerful reporting feature that gives you instant information on the Sandesha subsystem and any ongoing deliver sequences. Its configuration from both the server and client perspective is a snap. So let's take a look at how you'd configure both Web Services and Web Service clients to use WSRM with WSO2 WSAS.

Configuring a Service for WSRM

Let's take a Web Service and enable WSRM for it. This will let clients send it messages that take advantage of WSRM. Now one of the key aspects of WSRM is that this has to be transparent to your service. You don't have to change any of the code for your service to enable WSRM. As it turns out, there are a couple of ways to do this.

The first way to enable WSRM uses Axis2's Web Service descriptor. This is the `services.xml` for your Web Service. An example of a WSRM-enabled Web Service descriptor is shown in Listing 2.

Most of this is just normal Axis2 elements and configuration options. The only thing we did to enable WSRM was to add the `sandesha2` module. We didn't have to do anything else.

That's pretty easy, but WSO2 WSAS makes it even easier. WSAS's Management Console is a powerful graphical interface for managing Web Services. You can use it to enable WSRM for any Web Service. Just log in and go to Services -> <Your Service> -> Manage Module Engagements. You can then select `sandesha2` as shown in Figure 3.

After you select `sandesha2` and click the Engage button, the UI will be refreshed to show that `sandesha2` has been engaged and so WSRM has been enabled, as shown in Figure 4.

With just a few clicks we were able to enable WSRM on any Web Service deployed on WSO2 WSAS. Now let's take a look at creating a Web Service client that will use WSRM to send reliable messages to a WSRM-enabled Web Service.

Configuring a Client for WSRM

Enabling WSRM for a Web Service running on WSO2 WSAS was very easy. It's not much harder to enable it on the client as well. Let's take a look at the code for doing this in Listing 3.

This code is just like any other Axis2 client code you would write, but with a couple of notable exceptions. The class `org.apache.axis2.client.Options` let's us set Axis2 options on our `org.apache.axis2.client.ServiceClient` instance that we use to send SOAP messages to the WSRM-enabled Web Service. We use this class to enable WSRM by using the `engageModule` API. That's all it takes on the client to enable WSRM.

There's one other thing you might notice in Listing 3. In the code, two messages are sent. After the first one is sent, we add another option. This is the "last message" option that tells Sandesha to add this metadata to the SOAP header on our message, as seen in Listing 1.

WSAS Extensions to WSRM: Persistence

The WSRM specification is what is known as a wire-only specification. This means that it only specifies the behavior of messages being sent. It doesn't specify how clients and servers should deal with messages before they're sent or after they're received. The network itself is a major cause of unreliable messaging, but the clients and servers can also be the cause.

WSAS contains an extension to WSRM adding persistence to messages. There are two kinds of persistence or storage. The default is in-memory storage. This is just what it sounds like. A WSAS service will keep track of what messages have been received, along with acknowledgements, etc., in memory. This is very fast, but obviously there could be problems in the case of a crash. An acknowledgement might not be sent out, for instance. An alternative is permanent storage, which is typically storage in a database. This allows for recovery of state after a crash.

How do you configure storage? Luckily the WS-Policy standard is made for extensibility, so it can be leveraged to add things like storage. Listing 4 shows a policy specifying the in-memory and permanent `StorageManagers`.

Of course you can use different `StorageManager` implementations than the default ones shown here. To enable permanent storage on your Web Service, simply add a line of XML to your service.xml descriptor, as shown here:

```
<parameter name="Sandesha2StorageManager">persistent</parameter>
```

That's all you need to do to enable persistence on top of reliable messaging with WSAS.

Summary

In this article we've taken a look at the unreliability of the Internet. We've seen how WSRM addresses this problem with a standardized and powerful solution. We've looked at how WSRM works with SOAP. Finally, we took a look at WSO2 WSAS's implementation of WSRM using Axis2 and Sandesha. We saw how easy it is to enable WSRM on both clients and services using WSAS. WSO2 WSAS provides an easy turnkey solution to enabling a modern SOAP stack with various WS-* technologies, including WSRM.

Resources

- "WSRM and WS-R: Can SOAP be reliably delivered from confusion?" <http://www.ibm.com/developerworks/library/ws-rmpaper/>
- WSRM specification from the developerWorks library: <http://www.ibm.com/developerworks/webservices/library/ws-rm/>

- How WSRM and WS-Polling can be used together: <http://www.ibm.com/developerworks/webservices/library/ws-soa-wsrm-wsp.html>
- “Use Apache Sandesha to support Web services implementation”: <http://www.ibm.com/developerworks/webservices/library/ws-apache-sand/>
- Download WSO2 WSAS: <http://wso2.com/products/wsas/>
- The latest features in WSO2 WSAS 2.1: <http://ruchith.blogspot.com/2007/07/wso2-wsas-20-released.html>
- The WSO2 community on the WSAS Wiki: <http://wso2.org/wiki/display/wsasjava/Home>

- Web services and Axis2: <http://ws.apache.org/axis2/>
- “SOA with Axis2”: <http://www.ibm.com/developerworks/webservices/library/ws-soa-axis2-1/>
- TSS Interoperability blog: <http://tssblog.techtarget.com/index.php/mini-guide-apache-web-services/>. ■

About the Author

Michael Galpin is an architect at eBay in San Jose, CA. He's been developing software since 1998, and holds a degree in mathematics from the California Institute of Technology.

Listing 1 SOAP message with WSRM

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope>
  <soapenv:Header>
    <wsa:MessageID>e3016aa5f1b043f684cb58cae97fe317</wsa:MessageID>
    <!-- Other WSA headers go here -->
    <wsrm:Sequence soapenv:mustUnderstand="1">
      <wsu:Identifier>
        http://id.developerworks.org/e3016aa5f1b043f684cb58cae97fe317
      </wsu:Identifier>
      <wsrm:MessageNumber>2</wsrm:MessageNumber>
      <wsrm:LastMessage />
    </wsrm:Sequence>
  </soapenv:Header>
  <soapenv:Body>
    <!-- Body of message goes here -->
  </soapenv:Body>
</soapenv:Envelope>
```

Listing 2 Enabling WSRM using services.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<service name="SuperReliableService">

  <parameter name="ServiceClass" locked="xsd:false">
    org.developerworks.services.MyReliableService
  </parameter>

  <description>A sample web service.</description>

  <module ref="sandeshaz" />
  <module ref="addressing" />

  <operation name="someAction"
    mep="http://www.w3.org/2004/08/wsdl/in-only">
    <messageReceiver
      class="org.apache.axis2.receivers.RawXMLINOnlyMes-
sageReceiver" />
    </operation>
    <operation name="anotherAction">
      <messageReceiver
        class="org.apache.axis2.receivers.RawXMLINOutMes-
sageReceiver" />
      </operation>
    </operation>
  </service>
```

Listing 3 WSRM-enabled client code

```
public void sendMessage(){
    String serviceUrl = "http://some.server.com/SomeReliableWeb-
Service";
    ConfigurationContext configContext =
        ConfigurationContextFactory.
        createConfigurationContextFromFileSystem("path_to_
repository",
                                                "path_to_client_axis2.xml");
    Options clientOptions = new Options ();
    clientOptions.setTo(new EndpointReference (serviceUrl));

    ServiceClient serviceClient = new ServiceClient
(configContext,null);
    clientOptions.setAction("urn:srwr:SomeAction");
    serviceClient.setOptions(clientOptions);

    serviceClient.engageModule("sandeshaz");
    serviceClient.engageModule("addressing");

    // create an OMElement for your message body
    OMElement messageBody = null;

    serviceClient.fireAndForget(messageBody);

    clientOptions.setProperty(SandeshaClientConstants.LAST_MES-
SAGE, "true");
    serviceClient.fireAndForget(messageBody);

    serviceClient.cleanup();
}
```

Listing 4 Storage policy for WSAS

```
<wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" >
  <wsrm:StorageManagers>
    <wsrm:InMemoryStorageManager>
      org.apache.sandesha2.storage.inmemory.InMemoryS-
torageManager
    </wsrm:InMemoryStorageManager>
    <wsrm:PermanentStorageManager>
      com.wso2.sandesha2.storage.persistent.hibernate.
PersistentStorageManager
    </wsrm:PermanentStorageManager>
  </wsrm:StorageManagers>
</wsp:Policy>
```

SOA Issues Are People Issues...Not Technology

Finding a middle ground

BY DAVID LINTHICUM



According to the Burton Group, the issues around SOA are not so much about technology and complexity as they are about the people and the processes within an enterprise. Indeed, in a recent article by Jon Brodtkin, some of these issues are highlighted.

"The state of the union of SOA right now is there's some fatigue set in," Howard [Burton Group's Chris Howard] said, noting that when he recently asked an audience of 300 people whether their SOA efforts were going well, only a half dozen responded positively.

"The problem's not technology," Howard said. "People and processes are at the heart of what's wrong with SOA as it currently exists in enterprises."

I know I've been a broken record about this issue for a few years now, and while it's nice to get this validation, it's not nice to hear that SOA progress is hindered by office politics, turf battles, and good old-fashioned laziness. That's just the truth of the matter. The core issue is that IT thinks tactically, and SOA is strategic. They are not finding a middle ground.

Issues with SOA continue to be that SOA is a core and systemic change to the way we do IT. Change is something everyone seems to embrace conceptually, but when it comes down to actually changing systems that are a part of someone's job security, that's when things get ugly, fast.

Moreover, those who are tasked with driving SOA within their enterprise are not given the money and/or the power to drive change. Instead they are asked to "convince" and "influence." That never works; you have to control their budgets and be able to fire them in order to drive change at the speed it needs to be driven.

The counter to that argument is that those tasked with building SOA are doing a poor job in defining the value to the C-levels. Frankly, CEOs, CFOs, and even CIOs have heard it all before...reuse...agility...valuable technology change...and they never received the promised results. Thus they are skeptical with SOA and want some better data points and business cases. IT can't seem to get those business cases completed, and that's hindering progress as well.

The fix is easy. Just do the following:

1. Define the business cases clearly. If you can't, don't do SOA.
2. Empower those who need to drive the systemic change that SOA requires, typically, with the money and the authority to do something. Else, don't bother. You need to control the money and be able to fire people if this is to work in a reasonable amount of time. Otherwise, you're in endless meetings with people who

have agendas that don't include rebuilding the architecture for agility and reuse.

3. Think long term and strategic, not short term and tactical. It's okay; things won't collapse as you move from a reactive to a proactive mode. Indeed, that's how companies win their markets.
4. Start small, but keep the momentum going. Small battles win the war, and little by little the architecture will get better if you just keep moving the ball forward.

This is perhaps the motivation behind the new Web-oriented architecture movement, or WOA. In essence, developers and architects are so frustrated with the people and process issues within the enterprise that they are circumventing the politics and turf issues by outsourcing bits and pieces of architecture to Web-based development and hosting resources. I can't say that I blame them.

Reference

1. Brodtkin, Jon. "SOA failures traced to people, process issues." Network World, April 30, '08.

About the Author

Dave Linthicum is the CEO of StrikeIron (www.strikeiron.com), which offers Web services on-demand. In addition Dave is the author or co-author of 10 books, a thought leader in the Web 2.0 and SOA space, frequent keynote presenter, and has served as the CTO for 3 technology companies.

david.linthicum@strikeiron.com

Advertiser Index

ADVERTISER	URL	PHONE	PG
APC	www.apc.com		15
Certeon	www.certeon.com		21
DataDirect Technologies	www.datadirect.com		36
DataServices World	www.dataservicesworld.sys-con.com		23
Fujitsu	www.fujitsu.com/interstage		13
Intel	www.intel.com		2
Managed Methods	www.managedmethods.com		7
PlateSpin	www.platespin.com		11
SOA World Conf & Expo 2008	www.soaworld2008.com	201-802-3020	31
Software AG	www.softwareag.com		5
Web Age Solutions	www.webagesolutions.com	877-517-6540	35

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.



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





EXPLORE THE FRONTIERS OF DATA INTEGRATION

Don't go it alone.
Come with the software industry leader
in data access technologies.



DataDirect Technologies, the leader in data access and
mainframe integration technologies for over 20 years, is the
2008 Data Services World Diamond Sponsor.
www.datadirect.com