

SOA Web Services JOURNAL

DECEMBER 2006 / VOLUME: 6 ISSUE 12

Where's i-Technology Headed in 2007?

THE ANNUAL POLL — OF — INDUSTRY PROGNOSTICATORS

Case Study

**BPEL & B2B Synergies
Reduce Supplier
Enablement Costs**

Product Review

**ActiveBPEL 3.0 from
Active Endpoints**

Business

**How Much Will
Your SOA
Cost?**

**AJAXWORLD
UNIVERSITY
BOOTCAMP**

Jan. 22, 2007

Coming to New York City

Hands-On AJAX Training!

Visit events.sys-con.com



Gear up for XML excellence

Take off with the Altova® XML Suite, and
save ½ off the top tools for XML development.



Included with the Altova XML Suite 2007:

- Altova XMLSpy®, MapForce®, and StyleVision® Enterprise or Professional Editions
- Plus Altova SchemaAgent™, SemanticWorks™, and DiffDog® with Enterprise Edition
- Also get a FREE copy of Altova DatabaseSpy™ 2007 for a limited time*

The Altova XML Suite 2007 delivers the latest releases of world's leading XML development tools all in an unrivaled deal. It contains Altova XMLSpy, the industry standard XML development environment; MapForce, the premier data integration and Web services implementation tool; and StyleVision, the ultimate visual stylesheet designer. What's more, the Enterprise Edition also includes XML Schema management, Semantic Web, and XML-aware differencing tools. Save a bundle!

Download the Altova XML Suite today: www.altova.com

***Special offer:** Now until Dec. 24, 2006, purchase or upgrade to the Altova XML Suite and get the NEW Altova DatabaseSpy database query and design tool for FREE!

Conditions apply, see Web site for details.

Visit us online at WebServices.SYS-CON.com

Inside This Issue

CASE STUDY

10

David Webber
and Nishit Rao



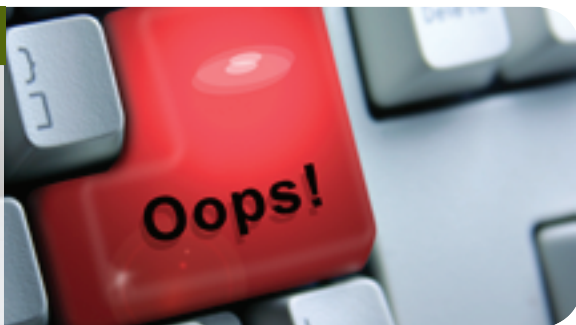
BPEL & B2B Reduce Costs

A Company Deploys New Technology & Techniques

ANTI-PATTERNS

30

Anthony Carrato,
Dr. Harini Srinivasan, and
Dr. Chris Harding



Learning from Mistakes

A guide to SOA anti-pattern solutions

PREDICTIONS

42

Jeremy Geelan



Where's i-Technology Headed in '07

Annual Poll of Industry Prognosticators

EDITORIAL

Would You Buy an SOA from This Man?

By Sean Rhody

5

BEST PRACTICES

Avoid SOA Pitfalls!

By Kevin Smith and Lou Blick

6

ENTERPRISE ARCHITECTURE

Part 2: Considering the SOA Reference Model

By Michael Poulin

16

PRODUCT REVIEW

ActiveBPEL 3.0 from Active Endpoints, Inc.

By Paul T. Maurer

22

METHODOLOGY

En Masse SOA Enablement Methodology Distilled

By Paul O'Connor

24

TRANSACTIONS

Long-Running Transactions in SOA

By Anshuk Pal Chaudhuri, Bijoy Majumdar, and Sunny Saxena

38

NEWS

SOA Web Services and more!

48

VIEWPOINT

SOA in the Small

Ajit Sagar

49

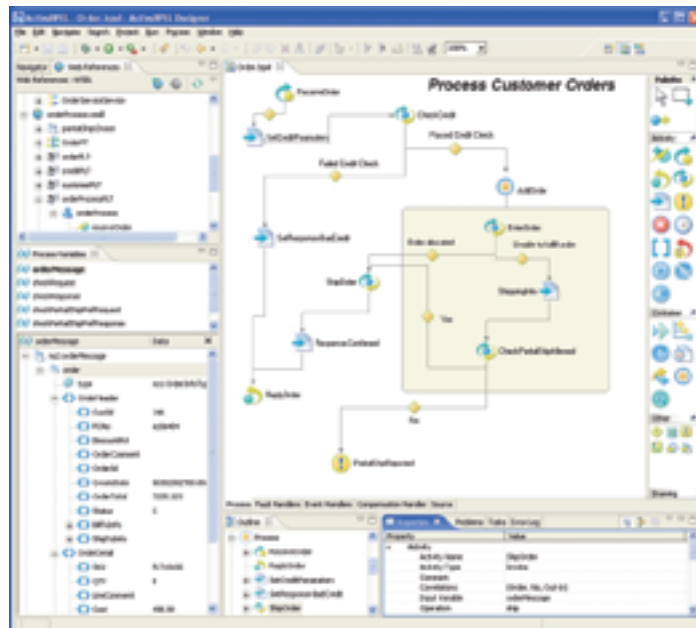
BUSINESS

How Much Will Your SOA Cost?

David S. Linthicum

50

Get Started with BPEL 2.0



**Build next-generation SOA applications
with the leader in BPEL technologies**

Download BPEL tooling & server software today

active-endpoints.com/soa

BPEL consulting and training.

**BPEL design tools, servers and source code for Eclipse, Apache Tomcat, JBoss,
WebSphere, WebLogic, BizTalk and Microsoft .NET.**

activeBPEL

**active
endpoints**

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

PRODUCTION**ART DIRECTOR**

Alex Botero alex@sys-con.com

ASSOCIATE ART DIRECTORS

Abraham Addo abraham@sys-con.com
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
WEB SERVICES JOURNAL (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:
WEB SERVICES JOURNAL, SYS-CON Publications, Inc.
577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677

©COPYRIGHT

Copyright © 2006 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.

Would You Buy an SOA from This Man?



WRITTEN BY SEAN RHODY

This month I thought I'd put on my sales hat for a moment and talk about what it takes to actually sell someone on the concept of using service-oriented architecture as the underlying paradigm for an organization's information technology implementation and direction. In part this is because there's still a good deal of resistance to SOA as that basis.

In the early days of trying to persuade corporations that SOA is a good thing, there was a great deal of commotion around the issue of security. The perception was that services were not really secure – and in fact, the very basic Web services standards of HTTP, SOAP, WSDL, UDDI and XML are not secure, other than perhaps with an SSL layer. But we've managed to put a stake in the heart of that specter, and worries about security now revolve more around proper implementation than around the actual availability of capabilities of an SOA to provide security.

Yet, while this had seemed at one point to be the sole stumbling block, why haven't we seen the floodgates open and a million SOA implementation take place? This is the key question.

SOA for many is like source code management. Unquestionably, source management is a good thing when we talk about software development, but the question that always comes to mind is – how do you quantify the goodness? Is there a return on the investment, which includes time, dollars, and software? Conventional wisdom says there is (and I personally believe it strongly), but at the same time, when was the last time someone actually presented actual cost savings for source code management.

I bring this up to point out the similarity of the situation with respect to service-oriented architecture. SOA provides key advantages that are fairly easy to see – loose coupling, application composition rather than development, the ability to manage services on a granular basis, and finer control of the actual work done by information technology. Integration, which becomes crucial the minute a business has more than one software system, is the key facet of SOA that makes every technologist realize that SOA is a better way to do software and implement processes.

The key difficulty with selling SOA as a concept for how to do technology is quantifying the actual benefit that results from using services. In business terms, determining the ROI of a change to SOA is what's important, but, perversely, it's also very hard to capture the value of SOA in absence of some business change to accompany it.

Look at it this way – it's easy to say, and also easy to understand and agree on the fact that SOA makes software integration much easier. But what is the dollar value of that ease, and how much of an effect will it have over time? That's the difficult question that is holding back funding of SOA in the current world. Sometimes you can quantify parts of it, such as saying you no longer need a particular EAI package, so your license, maintenance and operating costs could be viewed as the cost savings against which the cost of the implementation could be measured. Those cases are the simple ones, the ones where it's easy to make the case to go ahead with SOA. Unfortunately, in most shops, the case for a pure SOA conversion is much less clear. The ability to do something "better" needs to be correlated to a reduction in cost, or an increase in productivity, or both.

Most SOA approaches are couched in the context of some business improvement or other. It might be the deployment of a new ERP system, where the benefit of decoupled services can be quantified more easily due to the nature of the services and the cost of maintenance versus the same task with monolithic applications. Many IT shops are making SOA concepts – services, service buses, asynchronous invocations, and the like – a part of future designs and hoping at some point, when the shop looks more service oriented than not, that funding to go the rest of the way will be easier to acquire. That may be the way SOA comes into full fruition, not with a bang, but with a series of small improvements. Happy holidays and have a great new year. ■

About the Author

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

Avoid SOA Pitfalls!

Don't find yourself S.O.L with your S.O.A!

WRITTEN BY KEVIN SMITH AND LOU BLICK

➤ Software architects, developers, and project managers who've worked "in the trenches" of SOA design and implementation over the last few years have learned some hard and valuable lessons. Some of these lessons can help you avoid the serious traps and pitfalls associated with SOA design and implementation. Most of the gotchas that can run your SOA project into the ground are issues that can be addressed early in your project and revolve around proper project management and planning, architecture, and design.

This article offers seven guidelines for keeping your SOA projects on track, based on lessons learned in past SOA projects. Ignore them and you may find yourself S.O.L (SOA Outta Luck)!

1. Understand the Requirements and Manage Expectations

Proper requirements analysis techniques aren't unique to SOA, but they should be mentioned, because this is where you run the greatest risk of failure. For any project, requirements analysis, communication, and project planning are vital. To build and deploy a SOA properly, you must understand the enterprise requirements, you must know what has to be done technically, and you must understand the business processes, current policies, and current infrastructure of the organization(s) involved to meet their needs. The bottom line is – you can't solve somebody's problems if you don't know what they are.

Throughout your project, you should be in constant dialogue with your customer. Manage expectations by letting the customer know what benefits a Service-Oriented Enterprise (SOE) will bring, but also let him know that SOA won't magically bring world peace or solve all their problems! If there are problems with the organization's business processes, these have to be addressed first. Otherwise,

they'll blame you and the SOA.

If you take the time to listen to your customer and ask the right questions, you'll be in a better position to deliver a solid solution. Based on the requirements you collect in this phase, you'll know what to plan for in your project. Enterprise standards, SOA management and governance techniques, business process management, security methodology, performance requirements, and scalability requirements all evolve from the initial requirements phase.

2. Restructure Your Development Organization by Service Area and Application

Traditional development organizations are typically stove-piped application teams that develop the end-to-end functionality for an entire end-user application. This leads to duplicate functionality so separate your development organization into services teams and application teams.

Services teams develop the services that expose related functionality for a specific area of the business. The services they design and develop may be fine-grained, such as an inventory update service, or business process services that aggregate many of these fine-grained services, such as a service that creates a purchase order.

Application teams develop the user interfaces and business processes for the



end-user solutions needed in various areas of the business. These interfaces and business processes consume the services developed by the services teams.

Structuring your organization this way provides several benefits. First, you have one services team creating services that can be used by every application that needs this functionality, promoting service reuse. Second, each services team becomes a mini "center of excellence" in the area it's focused on. If you need to update product inventory in your specific application, for example, you have one team to talk to. Third, each functional area is centralized. If supply chain business processes have to be changed, the services team responsible for this area of the business can make the changes in its set of services, and these service changes get automatically propagated to each application using them. Restructuring your teams this way realizes the benefits of service reuse, forcing loose coupling between the user interfaces and business logic in your enterprise applications.

3. Create a Standards-Based Infrastructure

A common pitfall related to building Service Oriented Architectures occurs, when developers focus exclusively on standing up Web Services for each application, missing the "big picture." This focus leads to creating a large number of services that

speak a different language (i.e., a SOAP message schema), providing no infrastructure to manage the provisioning and lifecycle of these Web Services, providing no auditing capabilities, and failing to use the same service standards versions. This situation is the equivalent of “spaghetti code” in traditional applications. Here we’ll call them “spaghetti services.”

A solution to spaghetti services is the Enterprise Service Bus (ESB). There’s been a lot of industry debate about what an ESB is and isn’t, and whether or not an ESB is required in an enterprise SOA. For purposes of this article, an ESB is a distributed infrastructure that’s based on existing service standards and provides the means to:

- a) create a single implementation of your organization’s “plumbing” services, such as message routing, message security (i.e., single sign-on, non-repudiation, etc.), multi-service transactions, and guaranteed message delivery;
- b) provide a canonical message schema that lets any application easily “plug in” to the ESB and talk to other applications that are already plugged in;
- c) provide the message transformations necessary for application-specific data to be transformed into the canonical message schema;
- d) monitor and audit all message traffic to provide network administrators with accurate metrics, complete visibility into how the services are being used, and how performance can be tuned.

Using a distributed ESB also allows application teams to plug in to the enterprise SOA with the best tools for their specific applications. So, an application team can choose .NET, J2EE, Ruby, C++, or any other tool as long as it supports the defined service standards.

4. Create a Multi-Layered Services Architecture

There’s one thing as sure as death and taxes: requirements change (...and change ...and change again). A multi-layered approach to your Service Oriented Architecture can help you isolate yourself from these constant changes. Define fine-grained services that provide exactly one function (say, calculate shipping costs) and higher-level services that aggregate these fine-grained services into coarse-grained business processes using BPML/BPEL (for example, create purchase orders). This will

allow your business processes to change more easily when your business changes by allowing the coarse-grained services to aggregate different fine-grained services into the process and/or change the order in which these fine-grained services are called.

When you separate services into multiple layers, you can make the most of object-oriented design patterns. The Session Façade J2EE design pattern and the “Gang of Four” Mediator design pattern are two design solutions that map very well to Web Services. These two design patterns can be used to allow higher-level business services to centralize, control, and coordinate complex interactions with lower-level, fine-grained services. Separating your services into multiple layers will make your architecture more flexible and manageable.

5. Design Interfaces for Flexibility

Architectural flexibility revolves around the design of your interfaces. Here’s where well-intentioned developers go wrong:

- **Putting SOAP on Top of a Stove Pipe:**

Many poor excuses for “SOA implementations” just put SOAP abstractions on top of existing poorly designed stove-piped applications, mapping legacy interfaces to service interfaces one-to-one. Putting a Web Service wrapper around a brittle application with tight interdependencies is like putting lipstick on a pig, and when you do this, you miss the point of SOA. If you have to use an existing system, use only components of that system and take the time to rewrite the interfaces.

- **Implementation Details Tightly Coupled to Implementation:** It’s important never to expose the storage mechanisms or implementation details of your services in the interfaces, and it’s important to focus on standard schemas when defining your interfaces. If you have a Web Service method called “insertTieValueIntoTargetTableThree,” for example, it will be impossible for someone who doesn’t know the internal database schema to use your Web Service correctly. For that matter, if you ever migrated to a different database or changed your schema, you’d change your SOAP interfaces as well. Maintainability and version management could be nightmare in your SOA.
- **Lazy Web Service Generation:** Using point-and-click developer tools that automatically turn your current appli-

CORPORATE

President and CEO

Fuat Kircaali fuat@sys-con.com

Group Publisher

Jeremy Geelan jeremy@sys-con.com

ADVERTISING

Senior VP, Sales & Marketing

Carmen Gonzalez carmen@sys-con.com

VP, Sales & Marketing

Miles Silverman miles@sys-con.com

Robyn Forma robyn@sys-con.com

Advertising Sales Director

Megan Mussa megan@sys-con.com

Associate Sales Managers

Kerry Mealia kerry@sys-con.com

Lauren Orsi lauren@sys-con.com

Andrew Peralta andrew@sys-con.com

SYS-CON EVENTS

Associate Event Manager

Lauren Orsi lauren@sys-con.com

CUSTOMER RELATIONS

Circulation Service Coordinator

Edna Earle Russell edna@sys-con.com

SYS-CON.COM

VP 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: \$69.99/yr (12 issues)

Canada/Mexico: \$89.99/yr

All other countries: \$99.99/yr

(U.S. Banks or Money Orders)

Worldwide Newsstand Distribution:

Curtis Circulation Company, New Milford, NJ

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.

Don't wait until the end of the project to try to "fix" scalability and performance issues – trying to retrofit scalability into your SOA can cost more time and money than the initial implementation!

cation's objects into Web Services may be convenient and easy-to-use, but if the methods in your objects are poorly designed, you may be deploying non-intuitive WSDL. That's why you should use standard enterprise schemas in doc/literal SOAP messages.

It's important to plan to be flexible, and designing your interfaces properly will lead to architectural flexibility. Design your interfaces knowing that the implementations could change. Services should be loosely coupled and standards-compliant.

6. Design Your Services with Purpose, Expecting the Unexpected

Make the point of your Web Services simple, intuitive, and clear, so the results and consequences of the service calls are well known. Don't make assumptions on how they will be used – instead, design them so that the effects of calling them are clearly defined. Relating to Rule 4 (Create a Multi-Layered Services Architecture), both higher-level services and lower-level services should be clearly defined and described as such in a service registry.

Since Web Services are "black boxes" with published interfaces, they'll be called and used together in ways that we may not anticipate. In an earlier article that I wrote this year ("Six Basic Rules for Securing SOA-Based Projects" in the October issue of the SOA/Web Services Journal), I mentioned a previous SOA project I was involved in where the SOAP faults didn't contain data about the Web Services in which errors occurred. When end-user applications call Web Services that call other Web Services that combine the results of other Web Services, it will be incredibly hard to troubleshoot your enterprise applications if you don't know where the SOAP faults are coming from. This is one example, but it demonstrates what can happen

in a dynamic Web Services environment. Expect orchestration, expect Web Service chaining, and embrace the fact that your services will be used in ways that you may not have thought of. If you've planned for this, you'll avoid a lot of the traps that can cause chaos.

7. Plan for Scalability and Performance

At the end of the day, the users of your enterprise applications probably won't care that a SOA is in place. Users want functionality, and they don't want their applications to run at a snail's pace. As a result, your deployed systems will have to scale to meet the load required by the intended number of applications and end users. Enterprise architects will need to analyze the scalability requirements and start planning at design time, because the risks of not planning for scalability are huge – exposing a SOA to an organization that can't handle the load promises to ruin your entire project. Some of the issues here involve estimating service usage patterns, scalability planning for data resources, managing session state, and planning for security requirements that may involve CPU-intensive cryptography.

Certainly, there are off-the-shelf products that aid in scaling simple stateless services, and some ESBs come pre-packaged with scalability options. If you need strong cryptography, you may want to consider using a hardware appliance to offload cryptographic and XML processing. Depending on your requirements, you may also want to look at using a grid computing platform that's inherently scalable and robust. As you can see, there are many choices as you plan for scalability. All of these choices, however, are dependent on understanding the true requirements (see Rule 1), and involve planning up front. Don't wait until the end of the project to try to "fix" scalability and performance is-

sues – trying to retrofit scalability into your SOA can cost more time and money than the initial implementation!

Conclusion

This article has focused on best practices in SOA project planning, design, and implementation, and we have provided seven guidelines to help keep your SOA projects on track. Some of these guidelines (Rule 1 and Rule 7) focus on understanding the requirements, customer expectations, and enterprise application use, so that your SOA is purpose-driven and meets the needs of the users and applications in your enterprise. Other guidelines revolve around forming a successful development team's structure (Rule 2), using a standard SOA infrastructure (Rule 3), and important design guidelines for building services (Rules 4, 5, and 6). If you plan and architect your projects with these guidelines in mind, you'll find yourself in SOA's sweet spot, realizing the benefits of what SOA can truly provide. ■

About the Authors

Kevin T. Smith is a technical director at McDonald Bradley, where he leads the SOA & Semantics Security Team (S3T) focusing on information assurance initiatives for multiple SOA-based projects. The author of several technology books on XML, Web Services, Java development, and the Semantic Web, he has been invited to speak at conferences such as JavaOne, OMG Web Services, AFEI, ApacheCon, the RSA Security Conference, and Net-Centric Warfare.

kevintsmith@comcast.net

Lou Blick is a SOA architect at a large multinational consulting firm, where he focuses on the design and development of Service Oriented Architecture for a number of civilian, intelligence, and military organizations in the federal government. He was a member of the team at Celera Genomics that sequenced the Human Genome, helping design and build scientific applications for visualizing and researching the information contained in the genomic sequence.

loublick@yahoo.com

Facing a few barriers on the road to SOA?



JackBe's Rich Enterprise Application (REA) platform clears the road to SOA business benefits.

There's an abundance of products and vendors to help you *create* your SOA. Now, *consume* those SOA services with JackBe REAs to achieve the business productivity and results that led you to SOA in the first place. Our new REA platform combines the power of Ajax and SOA with reliable, secure communications to extend your SOA directly into powerful business applications.

A fully visual IDE reduces manual coding and accelerates the development process. And our lightweight, vendor-neutral platform easily integrates with existing middleware and services while maintaining server-side governance and control--unlike products that leave governance to the browser or create middleware platform dependencies.

Join over 35 industry leaders like Citigroup, the U.S. Defense Intelligence Agency, Sears, Tupperware, and Forbes who are already optimizing their business activity with JackBe solutions. Call or visit our website—let us help you remove the barriers on the road to achieving real business value from your SOA investment.



Web: www.jackbe.com

Phone: (240) 744-7620

Email: info@jackbe.com

BPEL & B2B Synergies Reduce Supplier Enablement Costs



WRITTEN BY DAVID WEBBER AND NISHIT RAO

➤ Although organizations use multiple technologies to solve myriad business problems, integrating two or more of these technologies to derive new business benefits presents additional challenges. This is especially true when the collaboration extends beyond an organization's own systems to include those of its business partners.

This article describes one such customer scenario in which Helena Chemical Company, a leading U.S. agricultural products specialist, used BPEL (Business Process Execution Language) and B2B technologies together to automate better and more productive supplier/distributor relationships. Put together, these technologies enabled a process-centric hub that provided significant business cost savings, faster supplier ramp-up, more responsive customer relations, and better process visibility both inside and outside the enterprise.

Helena Chemical Company Deploys New Technology & Techniques

Traditionally, the alignment of information and processes has proven challenging for Helena because of the vast differences in its partners' systems and the content they produce and consume. The challenge for Helena (see Figure 1) was to meet its XML and electronic data interchange (EDI) needs by balancing the right enterprise components across diverse supplier systems and to remove the administrative overhead of manual approvals, support, and data entry to streamline its seasonal order process, which processes tens of thousands of supplier interactions per partner in a four-month period. Helena also needed fault-tolerant exception handling and controlled manual intervention to resolve business decisions and needs.

Faced with the need to manage supply partners and handle ordering and production details ahead of farmers' seasonal crop-planting cycles, Helena chose a process-based approach in which its internal order processing seamlessly integrated with partner B2B processes. The end result was an integrated solution that smoothly bridged the disparities between the diverse supplier systems by using standards-based information exchanges in tandem with process alignment techniques.

Solving the Business Needs

Although Helena's existing EDI systems provided some measure of automation, it had an extended support infrastructure of paper, fax, and telephone-based coordination that needed to be replaced with the modern business process capabilities provided by BPEL and B2B solutions. These modern tools provide exceptionally greater levels of information agility, process control, and exception handling than traditional EDI-based systems.

When companies integrate with partners and suppliers, the initial focus is on automating the exchange of XML documents. This exchange can be handled by modern B2B protocols such as ebXML, RosettaNet, and EDI over the Internet (AS/2). But once the XML business documents are exchanged, they are often processed internally using traditional techniques such as batch-oriented transfer, human data entry into multiple systems, and traditional manual approval processes — leaving the information fragmented across multiple systems. This mismatch between slow manual internal processes and automated external processing can wipe out the gains achieved from supplier partner automation. Staff resources and time continue to be needed to manage customer relationships and resolve delivery and order tracking and coordination issues. The reverse scenario can also occur: internal processes are streamlined, but are saddled with archaic external B2B transactions that use FTP, fax, e-mail, and human interaction.

The ebXML standard solution stack provides a B2B toolkit that allows implementers to resolve these partner integration issues. This toolkit includes formal XML-enabled mechanisms that capture collaboration protocol agreements (CPAs) between participants — which define their actions, roles, and the transactions that will

be carried out between them — and couple them into discrete send/receive binary collaborations. These XML mechanisms also provide end-point addressing to server messaging systems, as well as secure and reliable message delivery. Each partner then uses its own ebXML message envelopes to convey its partner ID, business actions, and the transaction data needed for each step of the business process. A rule- and event-driven business process engine — such as BPEL provides — can then key off these XML exchanges and control the state transitions and exceptions, along with the integration into back-end enterprise resource planning (ERP) systems.

The Helena case study shows how traditional B2B interchanges can be rapidly upgraded and fully automated with modern business process management (BPM) technology using open standards-based software with graphical and visual configuration tools that set up each partner and control the exchanges needed. These tools allow fine-grained control using standards-based XML formats for everything from partner profiles to collaborative message mappings to process flow details and rules, thus allowing delivery of a complete event-driven solution (Figure 2) that doesn't require custom low-level coding.

Developing Process-Centric B2B Hubs

To move from traditional B2B hubs that integrate data with internal systems, to process-centric hubs, organizations are integrating B2B with internal processes using XML-aware BPEL scripting technology. These internal processes are not simple data-handling receive/respond integration processes, but the actual end-to-end, event-driven business processes that run the organization.

For example, in the past, a purchase order (PO) received over RosettaNet would typically be input into an ERP or customer relationship management (CRM) system and any further processing would reside in the ERP or CRM system. With BPEL, processes — such as “Order to Cash” and “Procure to Pay” — are executed in the business process engine, with subsequent orchestrations into ERP, CRM, and other internal applications. Thus, B2B business interactions can be integrated into the larger BPEL processes and become reusable and extensible as BPEL process components rather than static low-level coding done at various downstream points in traditional legacy systems. This approach — which leverages XML transactions, Xpath, and BPEL rules — creates a more agile solution that can deal with a wider range of disparity in the XML transaction formatting and data point details received from the external partners' systems. It also brings together data, rules, and process decisions into one coherent solution rather than dispersing them across systems.

This approach presents significant benefits (see Figure 3). First, the direct integration leads to fewer conflicts of rules and triggers across software components. Second, monitoring and management at the process level provide a view of not only the status of the process but also the status of the B2B engine, allowing for an integrated view of the overall process. Third, BPM technologies such as BPEL allow for the inclusion of human-directed workflow components, which are the critical elements of any true B2B integration, thus allowing for complete process integration.

Figure 3 shows a summary of the handoff and process flow between the B2B and BPEL components in the solution architecture. Each component performs a discrete role, such as steps five, six, and seven, which are responsible for correctly routing each message to the partner based on the collaboration protocol agreement

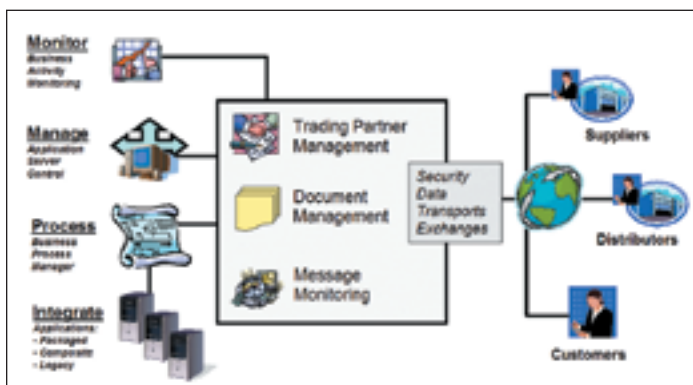


Figure 1: Requirements for internal and external integration

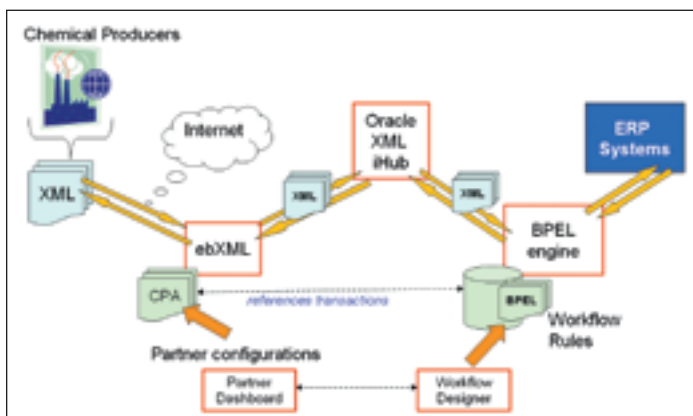


Figure 2: Event-driven XML scripted solution combining Oracle ebXML with BPEL

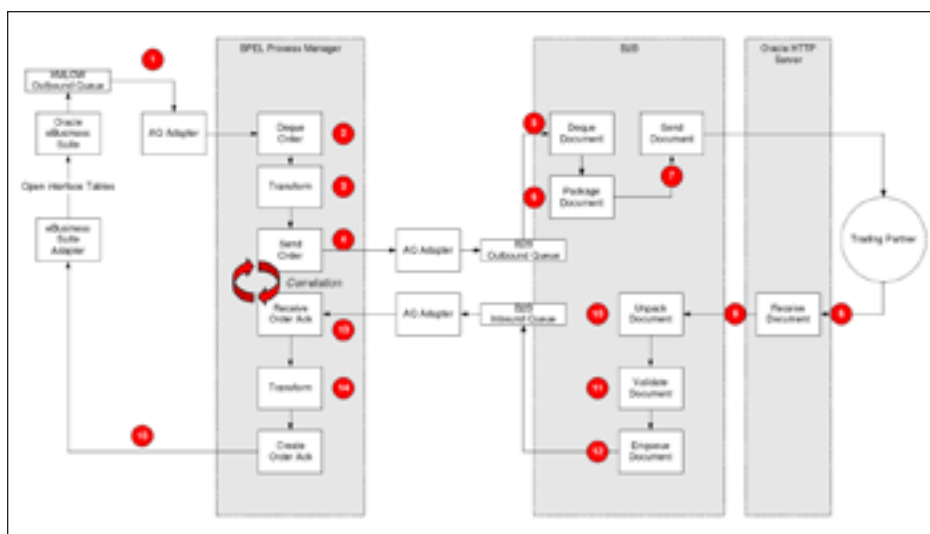


Figure 3: Example of workflow orchestration between B2B gateway and BPEL

(CPA) and the transaction type, and securely and reliably delivering the message. Similarly, in the BPEL processing section, steps three and 14 ensure that the message content is correctly transformed based on business rules, partner, and transaction type. Most importantly, BPEL provides the correlation and state management between inbound and outbound transactions (steps four and 13) based on the unique partner IDs and transaction IDs contained in the message envelopes. Steps one and 15 provide the integration to the back-end applications through standard APIs.

To deliver the full production environment at Helena, the IT consulting team implemented the new technology using an incremental approach. Building on the existing database foundation at Helena, the team added the Oracle XML gateway, along with ebXML messaging support for the partners' chemical industry transaction exchanges. Then the team prototyped integration into the back-end processing systems via BPEL and schema transformations and refined the workflow.

This incremental approach let Helena step from a simple EDI-based legacy environment to a sophisticated workflow process with B2B XML exchanges, ERP integration, and alerting. Particularly impressive was Helena's own design approach, which uses BPEL/human workflow processes in its B2B interactions and BPEL for IT task automation. By using BPEL for error analysis during the IT task mode, and then quickly switching to BPEL/human workflow orchestration to pass the error details to the appropriate e-commerce support specialist for resolution, Helena was able to rapidly evolve and adapt its business solution.

The Solution — The Technology Details

The seamless integration of the ebXML message service (ebMS), message transformation and BPEL tools inside the Oracle XML gateway was the key to rapidly configuring the business solution that Helena Chemical Company required (see Figure 3). Helena's own staff was involved in every step of the development process and was able to immediately assimilate the new technologies due to the transparency of the tools and the XML-enabled mechanisms. Stress testing was also done before going live to simulate seasonal peak production volumes and ensure that the supporting hardware configuration was sufficient.

Helena chose not to add complex lookup and transformation capabilities using the optional Oracle XML Publisher data trans-

formation toolset and instead achieved better control in BPEL itself by combining mapping as an external e-commerce function and customizing it by extending BPEL's own XML-handling capabilities. This approach will allow Helena to support future enhancements such as business activity monitoring (BAM) by creating triggers and rules based around the XML content and transaction flows, and Helena is actively investigating such future options.

Helena chose to use the Oracle solution set after carefully evaluating its existing tools and available options. (Although Helena's choice of solution gave it significant advantages, it's important to realize that comparable standards-based solution stacks can deliver equivalent levels of integration between the B2B ebXML standards approach and the BPEL approach.) The ebXML editor used to create each partner's collaboration protocol profile (CPP) is integrated with the Oracle messaging and partner registry and the BPEL workflow manager. This visual editing tool allows partner templates to be quickly created and then used to add new partners. The template ensures that the CPA generated for each partner can be validated against the transactions received, and also against the actions in the BPEL business process definitions. Similarly, the ebXML message envelopes link to the partner CPAs via the unique partner IDs and the business actions assigned to the XML message handling and the BPEL processing scripts. Each partner therefore has its own private, secure CPA definition between it and Helena that controls the B2B actions its system can perform and how Helena handles those exchanges internally.

Next, we consider the messaging transaction standards and the handling logic. While Helena's process was inherently more complex and involved a number of collaborating processes and human worklist interactions, let us look at the components and interactions of a generic process flow. The main control logic is typically implemented using the BPEL process definition editor as a series of BPEL actions, conditions, rules, and workflow steps. The B2B/XML interactions used by the agricultural chemical industry are a result of collaboration between the Chemical Industry Data Exchange (CIDX), Petroleum Industry Data Exchange (PIDX), and a nonprofit trade association representing the agricultural industry.

CIDX previously built the XML-based Chem standards that defined 52 business messages required by chemical companies to



tamino
XML Server

Dear IBM,
Congratulations on
your first native
XML database.

We know this business
inside out*, so:
Welcome to the club!

Kind regards,
Software AG

*For seven years we have fulfilled customer needs with
our powerful high-performance Tamino XML Server.
Welcome to the club, IBM.

>> www.softwareag.com/Tamino

 **SOFTWARE AG**

The Helena case study shows that by combining open standards and leveraging the flexibility inherent in XML organizations can produce an innovative solution that elegantly solves a complex set of business needs

carry out highly secure transactions with suppliers and customers over the Internet. This work was enhanced and extended in collaboration with PIDX, and then the ebXML messaging standards were selected as the preferred secure transport mechanisms via the Internet.

Between the B2B interactions, the BPEL processes, and the back-end ERP system, the XML content acts as the integration glue. To achieve this result, the BPEL — along with the Oracle XML handling tools — retains the values of the content as variables in memory, along with the state of the particular process handling. This means that content does not need to be unmarshaled from the XML into traditional tables and columns in the SQL database. Delaying that handling to a point as late in the process as possible adds agility and greatly reduces the overhead of continually repackaging information and maintaining the associated programming of code logic.

Once the information is committed into the back-end ERP system, the Oracle AQ adapter in conjunction with XML Gateway, an eBusiness Suite component, handles the manipulation of the XML content into the appropriate ERP formats and data. Again, this reduces the amount of custom coding required.

Open Standards, Open Solutions

The Helena case study shows that combining open standards and leveraging the flexibility inherent in XML can allow organizations to produce an innovative solution that elegantly solves a complex set of business needs. The ability of BPEL (in combination with B2B) to support this blend of technologies in an open way—quickly and easily—is crucial. For Helena, having these robust capabilities supported by graphical and visual development tools dramatically reduced the learning curve and setup times for its staff and delivered speed to market. These were critical business success criteria for Helena, because its cyclical business required quick delivery of the new system prior to its peak business period.

The technology combination used by Helena shows great flexibility to meet a wide range of operational needs, and the successful working model implemented at the company can clearly be duplicated by other companies with similar industry B2B and enterprise application integration (EAI) needs. By providing direct support for event management, the BPEL component also makes it possible to support additional enterprise-level process management solutions — such as BAM and ERP management reporting — that are designed to give greater visibility into processes, milestones, and intervals in delivery of services and products to the customers.

Summary of the Benefits and Lessons Learned

In bringing together the power of traditional B2B processing and the RAPID messaging standards defined for the chemical industry, along with the business process management tools developed in the BPEL specifications, the Helena Chemical project demonstrates the future pathway for Web Services and the business solutions built around them.

The B2B approach leverages years of solid formal business experience, while the Web Service approach and BPEL provide agile adoption strategies and technologies. Clearly these tools can be combined into a solution that offers customers effective business tools that don't require extended programming and development efforts to implement, and can therefore be implemented and deployed in weeks rather than months.

This scenario illustrates that the future of B2B is based on the formal business process and transaction models that have always been its strengths, and their adoption across a whole industry. Rather than diminishing the importance of B2B, the continued and accelerating adoption of Web Service technologies is likely to create more need for formal B2B-based business processes to reduce the cost of adoption and implementation across industry groups. ■

Resources

- A live audio transcript of the customer project review handoff conference call hosted by Helena Chemical Company.
- Information on ebXML messaging standards: www.ebxml.org
- Current work on OASIS BPEL: www.oasis-open.org/committees/wsbpel
- Tutorials on Oracle BPEL Process Manager: oracle.com/technology/products/ias/bpel
- Configuring Oracle with ebXML: download-east.oracle.com/docs/cd/B14099_19/integrate.1012/b19370/tutorial_ebxml.htm
- Oracle SOA Suite article: java.sys-con.com/read/204725.htm
- RAPID standards site www.rapidnet.org/Standards_Tools/XMLStandards.asp

About the Authors

Nishit Rao is a group product manager focusing on strategic SOA customers and their use of Oracle Fusion Middleware. He has more than 12 years experience in engineering and product management for messaging, CORBA, J2EE, integration, and SOA products, including experience developing and deploying middleware solutions on a global scale as an architect for a large logistics company. He holds an MBA from the University of California, Berkeley, and a BS in electrical engineering. nishit.rao@oracle.com

David Webber is an industry consultant for applications of XML and ebXML. Previously vice-president of business development for XML Global Technologies, Inc., and co-founder of the XML/edi Group, he is now working with OASIS XML standards committees. David's current project team just got a special recognition award for its SOA work from the SOA/CoP Second Conference in Washington, D.C. He holds two U.S. patents on advanced information transformation with EDI and has a degree in physics with computing from Kent University, England. drwebber@acm.org



The Art of Web Services Testing



Build Your Applications on 4 Pillars of SOA Testing

- I. Automated Functional Regression
- II. Performance Profiles
- III. Interoperability Compliance
- IV. Vulnerability Risk Posture

Download SOAPSonar Enterprise Edition
<http://www.crosschecknet.com>

Considering the SOA Reference Model

Part 2 – Design Pillars

WRITTEN BY MICHAEL POULIN

➤ The main drivers for SOA-based architectures are to facilitate the manageable growth of large-scale enterprise systems, to facilitate Internet-scale provisioning and the use of services, and to reduce the cost of organization-to-organization cooperation – SOA RM

When approaching a SOA implementation, I would like to consider two fundamental questions that many developers ask:

- 1) What's the difference between service-oriented and service-based architectures?
- 2) What special architecture elements are defined by the SOA RM?

In my opinion, the answer to the first is in the difference between the words oriented and based. I believe that smart IT organizations offer a lot of services already because the technical benefits of services have been well known for a while. However, the applications based on these services are still monolithic and don't provide convenient ways to implement business change. SOA, by contrast, elevates services to the level of first-class citizens by associating them (well, some of them) with business responsibilities. That is, business tasks can finally surface in the form of aggregated services that are open to business changes by design. SOA RM stresses a new role and meaning for services and outlines that:

- 1) "The central focus of Service Oriented Architecture is the task or business function" versus both IT task/function and business operation automation task. This means that SOA primarily targets actual business services and processes instead of just improving existing (obsolete or legacy) applications.
- 2) SOA service may be MUCH richer than a technical service because SOA service is supposed to run in the "execution context." The latter is defined as the set of technical and business elements that form a path between those with needs and those with capabilities. The "business

elements" open absolutely new sphere of conditions, particularly business conditions that a regular technical service doesn't deal with.

- 3) SOA "reflects business-motivating considerations" expressed "in service descriptions and service interfaces", where the service description can be expressed in the form of a contract. The service contract includes policies that "MAY also express business-oriented policies – such as hours of business, return policies, and so on".

If you elaborate on the business aspect of SOA, you'll get a much different picture of the service realm with different purposes, use, governance, maintenance, and, finally, architectural concept than we have in a "regular" service-based architecture. Besides these, I don't see any real differences. The service serves; the questions are how, where, why, when, to whom, and if it's needed after all.

The second question is trickier. The SOA RM standard doesn't articulate much on architecture elements. Instead, it defines SOA services and their relationship in new way that allows for addressing actual business services and processes by assigning business responsibilities to the services directly. That is, the standard defines what an IT service has to be oriented to in order to provide for IT agility to the business. I understand that the intention of SOA RM is giving us the definitions of the service and the "associates" that require a business service-oriented infrastructure. This infrastructure is the very architecture meant by the reference model.

So, since we're going to discuss SOA design pillars, let's list some commonly used



elements of SOA infrastructure that include:

- a SOA Service Communication Framework
- a SOA Service Composition Framework
- a SOA Service Transaction Framework
- a SOA Service Security Framework
- User Interfaces (including human UI) and
- a SOA Service Component Adapter Framework

Elements of SOA infrastructure are reviewed in many technical publications; we will use them for references without deeper discussion.

Going forward, let's notice that some pillars relate to the SOA service design while others relate to the service interactions with other services and its consumers. Pillars 1 to 5 have ordered dependencies, the rest are applicable to the entire design process. Plus we'll use terms that have been defined and discussed in part one of this article, so you might look through it.

Service Design Pillars

Pillar 1

Considering Business Services selected for the implementation in SOA *identify technical "vertical" and "horizontal" services*. A vertical service implements one or a few business tasks while a horizontal service implements a supporting SOA infrastructure task, e.g. a Service Repository or Service contract management function. Traditional technical services like security, login, and the like also fit into the horizontal services category.

While the principle of separation of concerns is fully applicable here, the tasks implemented by the same vertical service have to provide "cohesive collections of be-

SOA RM:

SOA does not provide any domain elements of a solution that do not exist without SOA

SOA RM:

A service is opaque in that its implementation is typically hidden from the *service consumer* except for...

- (1) the information and behaviour models exposed through the service interface and
- (2) the information required by service consumers to determine whether a given service is appropriate for their needs.

SOA RM:

- Service visibility is promoted through the *service description* that contains the information necessary to interact with the service and describes this in such terms as service inputs, outputs, and associated semantics.

The service description also conveys what is accomplished when the service is invoked and the conditions for using the service.

haviors" from the business perspective. Plus technical services have to be reasonably coarse-grain in their interfaces (to minimize the network round-trips and future modernizations in production caused by new technologies) but still provide flexibility. Nevertheless, one has to expect a certain amount of exploration and refactoring in this area of the service lifecycle.

When identifying horizontal services, first consider scalability, reliability, performance, expendability, and manageability – all together; consider programming convenience second. Services that are too generic usually don't provide the best performance just as too detailed ones cause problems in aggregation, management, and maintenance in light of frequent changes in the execution context.

Pillar 2

Considering business units of work, *identify business data and its flows*. Take into account two orthogonal sets of requirements – the presentation layer and the persistence layer requirements. Some units of work can require compositions of them, so watch for a clear information transition with layered data format mappings. Also consider combinations of the business and just the technical data in the flows. That is, define metadata for the business and technical data upfront and operate at the metadata level (UML and WS-CDL can help).

Metadata is an important SOA element. Metadata provides the necessary data abstraction and allows for a data-quality controls. An XML Schema is one possible metadata definition and control instrument. This doesn't mean that only XML format is recommended. In some cases, especially for synchronous short-running transactions, programmatic Data Objects may be used for performance improvement. However,

the Data Objects better be derived or generated from the XML Schema, which would provide metadata enforcement (related utilities are available in Java and C# and are known as XML binding).

Pillar 3

Define and lock down service interfaces and interfaces to the service resources such as persistence storage. Design the interface as business-oriented coarse-grained so your consumers can avoid frequent refactoring.

Best practices recommend avoiding RPC-like style in service interfaces. Web Services, RPC encoding, and SOAP-encoded arrays, in particular, seriously restrict service interoperability; consider a document/literal style instead. For non-XML cases, use a container style of interface to pass data structures and "exchanged" operations (aka commands in the Command Pattern).

It's important to design service interfaces with an eye to security, transaction, manageability, error handling, versioning, and interoperability from the beginning. Adding any of these things later is either extremely expensive (in effort, time, and money) or practically impossible.

For example, security is very important these days because of multiple technology threats and industrial/corporate security policies. Transaction support is dictated by its potential participation in service compositions. Manageability and error handling are core to any business service and process. Interoperability is one of the most critical factors in the SOA service lifecycle. That is, use of specific programming platform constructs can significantly reduce the utility of a service and lead to its refusal.

In spite of any ideas about keeping the service interface immutable, in real life variations in service invocation always happen over time. Anyway, a service interface

isn't the only one to constitute a SOA service; there are also possibilities of SOA service behavior changes that affect service consumers. That is, the service has to be versioned. The versioning may be expressed via the interface and/or provided by a service registry. Only one condition has to be preserved in this regard – no service parts including the interface may be versioned separately from the service itself for service consumers.

Pillar 4

A service access protocol may be considered a part of the service interface definition or a separate issue. The spectrum of communication protocols used in SOA is wide including TCP/IP, HTTP/HTTPS, SMTP, and IIOP.

The designer has to *choose communication protocol(s) for every SOA service*. When considering a protocol, one has to take into account: 1) invocation model – synchronous or asynchronous; 2) the amount of transferred data; 3) performance; 4) extendibility; 5) scalability; 6) reliability; 7) transaction; and 8) manageability.

Protocols are also important for communicating with service resources. It's better to use standardized protocols where possible. For example, current best practice refers to the Web Services Invocation Framework (WSIF) as a binding bridge between Web Service's WSDL and a particular service resource protocol while Java components rely on JBI.

The right protocol is a balanced choice. For example, HTTP/SOAP-based protocol is slower than IIOP but it's more extendible; TCP/IP is more reliable than UDP but it's slower. Don't forget that one service may have more than one communication channel. For example, the same session EJB can be reached through RMI-over-IIOP and SOAP-over-HTTP/S.

The pioneers of SOA implementations have recommend doing a “protocol performance POC” for particular kinds of tasks. That is, the earlier in the design process a performance test gives the results the better for the entire project because refactoring in this area means costly redesign from the scratch.

Pillar 5

Define the SOA Service contract and Execution Context. One of the most important parts of the SOA service design is the definition of the SOA service contract template. While SOA RM simply says that “a contract...represents an agreement by two or more parties. Like policies, agreements are also about the conditions of using a service; they may constrain the expected real-world effects of using a service,” the SOA service contract may be relatively complex. The contract, preferably written in XML, defines among other things:

- runtime policies/constraints
- runtime functional service parameters exposed to the service consumers; these parameters can be monitored and controlled
- non-functional service parameters like scheduled service availability
- business functional characteristics promised to service consumers like gathering audit information on service invocation events (the characteristics should be evidential and measurable)
- all service interfaces and communication protocols exposed to a particular consumer
- definitions or references to the Change Control procedures related to the service
- service version information
- references to the external (to your organization) SOA services invoked by the service. It's important for the business to know what other SOA services are involved because some political and business considerations may cause certain constraints on such involvement
- some custom or business domain specific parameters if needed

The content of a SOA service contract isn't standardized. However it's the only document that's used by a cautious consumer to decide whether your service is the one that meets his needs. It is also the document that defines your liability as a service provider to your consumers.

Runtime policies better be derived from the SOA service contract. The policies have to follow standards wherever possible. As

SOA RM defines it, “A service contract is a measurable assertion that governs the requirements and expectations of two or more parties. Unlike policy enforcement, which is usually the responsibility of the policy owner, contract enforcement may involve resolving disputes between the parties to the contract. The resolution of such disputes may involve appeals to higher authorities.”

The SOA service implementation process treats the service contract as an objective set of development tasks. Not all implementation tasks ought to be expressed in the service contract, but only those that get visibility to the service consumers. Other tasks are the subject of regular application requirements.

Another fundamental characteristic of a SOA service is the execution context. According to SOA RM, the “Execution context is central to many aspects of a service interaction.” The context includes concepts such as the SOA service contract, and behavioural and information models. The same service can be viewed and act differently in different execution contexts. For example, a logging service may store data “in the clear” when serving load-balancing calculations but it has to encrypt data for financial risk calculations (i.e., financial regulations constitute the execution context).

Pillar 6

Develop for security. SOA security isn't included in the SOA RM. Nonetheless it's obvious that the model's “real-world effects” can't survive without a business trust, i.e., without service security. We don't recommend developing a special implementation for each service. Instead use corporate security services at the level of service communication and service components, e.g., security policy syndication services.

Since a lot of security threats come from inside an organization, perimeter protection (external firewalls) isn't enough. We can outline the minimal SOA security requirements as 1) end-user bi-directional authentication with the service, and 2) inter-service bi-directional authentication. End-user and inter-service authorizations are much desired but may be optional in particular cases (e.g., a consumer may access everything after he's confirmed as a legitimate user). Other security measures are optional as well and depend on the nature of business data and processes.

Traditional security means like PKI, Kerberos, SSL, two-way SSL, user name and password tokens serve well in SOA. The

service-specific means relate to the XML security standards. They include XML Digital Signature, XML standards for PKI (XKMS), SAML (for propagating the subject and its credentials), XACML (for transferring security policies and policy validation controls), and other encryption techniques applied to the exchange messages and communication channels. The choice of security methods to be applied depends on corporate and industry policies and regulations.

The takeaway here is that whatever security means ought to be implemented, they have to be considered in the design from the beginning. It's almost impossible or extremely expensive to add security in later phases of development.

Pillar 7

Develop to support transactions. SOA RM recognizes the importance of service transaction as “higher-order attributes of services' process models” but doesn't detail the topic. In practice, SOA uses commonly accepted standards for transaction management on different platforms and adds cross-platform transaction management standards. Transactions can be used as in an aggregated service (choreographed aggregation) as in an orchestration of services reflecting fragments of Business Process Management (BPM). The orchestration is typically based on the BPEL standard that introduces short-running and long-running transactions.

A short-running transaction is usually a synchronous RPC-type invocation running according to the known rules of transaction isolation. Transaction standards for Web Services and messaging include WS-Transaction and WS Reliable Messaging (WS-RM). A short-running transaction is good for performance and commonly used for co-located services or in cases where failure is unlikely and error handling is simple. A long-running transaction is an asynchronous transaction where the transactional state may be persisted for long periods of time (days, months, years) and are restored after a certain event. This kind of transaction is quite useful in manual operations or human decision making in the automated business process. BPEL also introduces a transaction capable of “undoing” the mistaken results of the previous transaction by compensating them (like an overcharge refund).

Transactional behaviour has to be embedded in the SOA service from the design phase. This will help provide future service aggregation- and orchestration-based SOA

Fiorano SOA™ 2006

The Quickest Path to an SOA

Fiorano provides the industry's first framework enabling developers to rapidly create SOA applications in well-formed J2EE code.

Key Features:

- JCA-compliant Service-component Framework
- JMS based messaging
- BPEL support for composite components and applications
- Synchronous and Asynchronous Services in a single framework
- Support for multiple languages: Java, C, C++, C# and others

With Fiorano you can:

- Directly deploy Service Components and BPEL processes to any standard J2EE container
- Rapidly deliver ACID transactions across front-end (JSP) and backend processes (Databases, ERPs, etc.)
- SOA enable your existing J2EE application (EJB, JCA, JMS) without reengineering or code-generation
- Easily debug complex flows spanning multiple applications and middleware
- Perform runtime service deployment, service updates and application changes



Meet us at:



May 16-19, 2006

Booth # 538, Moscone Center - San Francisco

Download your copy of Fiorano today!

<http://www.fiorano.com/downloadsoa>

Fiorano®

Enabling Change at the speed of thought

The reason that orchestration (and choreography) are not part of the SOA RM is that the focus of the RM is on modeling what a service is and what key relationships are involved in a modeling service.

solutions that might not be clear at the moment the service is initially designed.

Pillar 8

Design for interoperability. As mentioned, interoperability is a crucial and critical characteristic of a SOA. The SOA RM says, “The value of SOA is that it provides a simple scalable paradigm for organizing large networks of systems that require interoperability to realize the value inherent in the individual components.”

The existing WS-I Basic Profile addresses most interoperability issues in Web Services. CORBA IDL mapping does the same for the services communicating via IIOP. Communication via HTTP is strictly defined and doesn't leave room for interoperability issues (except for JavaScript flavors) while SOAP-over-HTTP/S still has some mismatch problems in mapping to different programming languages.

The recommendation here is to put a lot of attention on the data types and naming convention policies (besides the protocols) on the both sides of the interface – for the requester and the provider components. The policies must enforce compatibility and interoperability. One possible mechanism for data type and naming space interoperability control is XML Schema or any other schemas (metadata) accepted by the requester and the provider.

Another best practice recommendation is the creation of a proxy (isolation mapping layer) around the interfaces. That is, the data formats passed through the service interface shouldn't be driven by internal data formats (for either requester or provider). If the evolution of either side causes data format changes, it's better to create object mapping in the proxy than to try to modify the interface.

Pillar 9

Use Rules for flexibility and easy-to-change implementation. Rule engines are just tools used in SOA infrastructure implementation and certainly not a part of the SOA RM. However, being combined with policies on one side and decision-making

points on the other side, rule engines become quite powerful instruments for providing a high level of flexibility and quick adaptation to business changes. Leading SOA infrastructure/framework vendors include different rule engines into their business process design tools and Enterprise Service Bus solutions.

While rule engines are really convenient instruments, there's no compatibility between engines from different vendors; a rule engine locks you into a vendor. Anyway, not everything should be driven by rules – rule maintenance is expensive and requires special skills. That is, the designer and developer have to be reasonable in using rule engines.

For example, if the business logic introduces a point for choosing from a finite number of cases (while cases change seldom), using rules may be overkill. Another known risk in using rules is a potential rule contradiction. That is, a rule engine generally doesn't automatically validate if a new rule contradicts any existing rules and any permitted combinations of those rules. Without such validation, the new rule can bypass restrictive rules and expose prohibited information (e.g., a junior clerk gets access to a high-volume account). So when using a rule engine in SOA, proceed with caution.

Pillar 10

Develop for extensibility, scalability, and reliability. These are not service-specific characteristics but they are important for a SOA because they're typical business objectives.

In particular in a business nobody would deal with you if your service wasn't reliable. You're also at the mercy of a competitive service if you can't scale well enough to provide service for a few years into the future. The existence of another service providing very similar functions is one of the realities of the business world, and, so the reality of SOA. It's not a resource or application redundancy that enterprises try to get rid of using services, but it is service redundancy, which is a natural feature of a Business Pro-

cess that may be modeled in BPEL for SOA.

Extensibility deserves special note. As we know, one of the goals of SOA is to provide an architecture that can easily accumulate and adapt to business changes. This may be done in different ways including different compositions of reusable service components and/or by design for extensions. SOA allows for design for extensions because it can manage the scope of potential extensions, i.e., SOA architects and designers can know upfront what can be extended. In particular, only those extensions are possible (and should be observed in SOA) that can be derived from the enterprise business model. If a required or proposed extension doesn't fit into the business model, this signals that either something is wrong with the proposal or it's time to change the model. The latter can't be done by IT alone because it's a lifetime event in the organization. For example, if an enterprise is in the skating ring business, its business model can be extended to sprint skiing, sprint running, boxing or a swimming pool (inside the ring), but a mountain slalom is certainly outside the scope of the business model.

Pillar 11

Service components development and bottoms-up recycling. This is a service-specific pillar. It relates to the methodology and standards applied at the level of the design of the software components comprising SOA services.

Dealing with enterprise IT assets, the SOA design decomposes existing applications and operational processes to integrate some of them in new way. SOA is supposed to destroy application silos and reuse or recycle only the application features that relate to the business services/processes instead of mummifying existing operations under new interfaces. The emerging Service Component Architecture (SCA) specification (proposed to the standard) addresses target design of the service components. SCA defines coarse-grained business interfaces for the components, i.e., interfaces “with relatively few methods and where parameters and return values are typically

document-oriented.” Such interfaces may be exposed in any programmatic model, e.g., WSDL or Java, without changing its business orientation.

Service component development can be done in an iterative-incremental manner after the business service/process is defined. Well-known best practices in distributed computing development are the only thing recommended here. For particular service development, the agile methodology is the first candidate. If an agile methodology were combined with a design for potential extensibility, this would be the right development approach to SOA.

Special attention should be paid to the bottoms-up development that I call recycling. Obviously when SOA development starts we can expect that the enterprise already has its applications, legacy systems, even services. These can be effectively reused in the new architecture as they are or partially if they can be converted into the SOA services, i.e., related SCA interfaces and associate SOA service contracts may be defined. While SOA RM says, “A service contract is a measurable assertion that governs the requirements and expectations of two or more parties,” we need to notice that the Service contract is consumer-centric. If a legacy asset can operate only in an application-centric manner forcing consumers to become dependent on the application’s internal processes and procedures, such an asset might not be the right candidate for transformation into a SOA service.

Pillar 12

Watch for service testing and governance. These are fundamental tasks for SOA design, not only for development, deployment, and runtime. Both of them are rich topics deserving, at least, a few articles each. Being the elements of the SOA implementation technique, they aren’t a part of SOA RM but no realistic SOA solution can afford to be put into service without them.

Testing in SOA is similar to testing in a component-oriented architecture. However SOA stresses testing in different execution contexts. Service testing requires the intensive design of test cases and a related test environment (that may be initially unavailable). This includes functional tests for the service itself, tests for its behavior in different aggregation/process flow scenarios, and tests for non-functional and/or service contract obligations.

Acceptance testing of a SOA service is

individual to each organization. Its scope is actually limited by the SOA service contract. Even if no new code is released but the service is to be used in a different business context (e.g., the service contract is changed), new test sessions are generally required.

SOA governance is one of the hottest topics of our day. In a nutshell, SOA needs design, development, deployment, and production governance. Governance is based on policy controls, not all of which can be automated. A service contract can help a lot in defining informal business policies but the controls are still required. The issue with SOA services is that not all elements of the service’s behaviour can be guaranteed by testing; it’s necessary to watch the service at runtime and adjust it.

For example, a very popular IT promise of service reusability is a double-edged sword for the business. On one hand, the more a service is reused the better its ROI gets, and businesses love it. On the other hand, each reuse is subject to a new governance session and new risk validation (in the new context) – both of them costly. If one skips governance because the service hasn’t changed, it gets worse for the business because modern industry policies and regulations like SOX require that certain procedures (controls) be in place. If one skips risk validation, an organization can end up in a situation where a service contract discloses the sensitive information described in another service contract and this problem isn’t recognized (and rectified) upfront.

Process Design in SOA

As we know, SOA services can participate in the SOA processes that reflect business processes. Many designers observe the business process first and try to derive business services from it. Designing SOA process is quite important task. However, the current version of the SOA RM touches on this very delicately. “The process model,” it says, “characterizes the temporal relationships and temporal properties of actions and events associated with interacting with the service. Note that although the process model is an essential part of this reference model, its extent is not completely defined.” Moreover, “The reason that orchestration (and choreography) are not part of the SOA RM is that the focus of the RM is on modeling what service is and what key relationships are involved in modeling service.”

It looks like the BPEL standard covers

most business needs so far. Before presenting it in the SOA RM, we probably ought to collect and analyze more information about automated business process modeling and its exposure to the aggregated/composed service modeling.

Conclusion

This article elaborated on parts of the SOA RM standard related to the SOA service design. It offered and discussed several SOA design pillars based on the enterprise business model described in part one of the article. The fundamental takeaway was that the SOA service definition and its relationship with other systems, given in the standard, assumes creation of a technical enterprise architecture that can implement core business services and process and ease acceptance of business changes via business-oriented technical services. SOA services start behaving more like business entities running in business execution contexts and having business responsibilities in their service contracts. The shift from IT application silos to business orientation is the proposed way of making IT architecture business agile in the enterprise. ■

References

- 1) Duane Nickull. “Why We Need the OASIS SOA Reference Model.” www.looselycoupled.com/opinion/2006/nicku-why-arch0104.html.
- 2) Satadru Roy. “SOA Infrastructure: Mediation and Orchestration.” www.soamag.com/I1/0906-2.asp
- 3) Michael Poulin. “Service Versioning for SOA.” SOA-Web Services Journal. Vol. 6 Issue 7. <http://webservices.sys-con.com/read/250503.htm>
- 4) Alcedo Coenen. “An SOA Case Study: Agility in Practice.” SOA Magazine. Issue II. November/December 2006. www.soamag.com/I2/1106-1.asp
- 5) Service Component Architecture, Version 0.9. www.oracle.com/technology/tech/webservices/standards/sca/pdf/SCA_White_Paper1_09.pdf

About the Author

Michael Poulin works as a consulting technical architect for leading financial firms. He is a Sun Certified Architect for Java Technology. For the past several years, Michael has specialized in distributed computing, SOA, and application security. m3poulin@yahoo.com

ActiveBPEL 3.0 from Active Endpoints, Inc.

Still active and out in front...

REVIEWED BY PAUL T. MAURER

➤ We last looked at the Active Endpoints Inc. product suite back in May of this year. Since then, the team at Active Endpoints has been busy working towards the 3.0 release of its ActiveBPEL Designer and engine.

The biggest and most interesting enhancement is full support for all WS-BPEL 2.0 constructs. This feature will let users gain early access and experience with the upcoming OASIS standard.

The folks at Active Endpoints carefully thought through how to support users in their move to WS-BPEL 2.0. This resulted in the ability of BPEL 1.1 and WS-BPEL 2.0 processes to co-exist during design, test, and execution in both the designer and the engine. This lets users migrate their processes at their own pace. I've been burned in the past by products that required a big bang conversion so the ability to support 1.1 and 2.0 processes easily is a huge win.

ActiveBPEL Designer

My last ActiveBPEL review focused on the designer tool so I won't spend much time on it here except to point out a few enhancements.

First of which is the ability to migrate any valid BPEL 1.1 process to a WS-BPEL 2.0 process. This is done by saving the existing 1.1 process as a WS-BPEL 2.0 process. During this "Save As" step ActiveBPEL will convert as much of the process syntax as possible and report conversion issues on any BPEL sections that couldn't be converted to the 2.0 syntax. These issues are displayed in the designer as static analysis errors. Users can then correct any errors. Once this is done, the user has access to all the new 2.0 features. Again, another nicely thought out touch for converting a process. I was able to convert one of my existing 1.1 processes and correct any issues easily.

Finally, Active Endpoints has added some of the obligatory GUI enhancements such as grids, anchoring annotation, and scope level auto-layout. The ActiveBPEL Designer continues to be a well-thought-out and constructed tool.

Engine Basics

The ActiveBPEL Engine manages BPEL processes. For those new to BPEL, a BPEL process is a collection of Web Services with various interactions specifically orchestrated to complete some portion or all of a business process. Each Web Service is treated like a black box that does some part of the process. Ordering the Web Services calls, information passed to and returned from the calls, and conditional execution are called orchestration. Its details are defined in the Business Process Execution Language specification (the BPEL Spec).

The ActiveBPEL Engine is designed to run in most of today's Application Servers. I had the WebLogic 9.2 application server handy so that acted as my test bed. The ActiveBPEL Engine also requires a database, so I downloaded and installed MySQL 5.0. The installation wizard does a nice job of walking you through the installation and it automatically configured the application server based on my responses. Keep in mind you still have to know how to configure your application server of choice. I had to ensure that the appropriate database drivers were set up and that my server was clustered, but the ActiveBPEL wizard handled the engine-specific setup nicely.



Once I had the product installed on the server, I was able to load up an existing 1.1 BPEL process. I specifically chose the process I created for the original review back in May. It loaded and ran in the server without incident. I then saved the process as a 2.0 process. There were a few minor errors that I quickly fixed then deployed the new processes to the server and again ran it without any problems. Granted my process wasn't incredibly complex, but it was still nice to see everything work as advertised.

Process Versioning

The ActiveBPEL Engine supports versioning processes. Various versions of a process can execute in the engine concurrently. Versioning includes effective dates and expiration dates of processes. This is a great feature and invaluable when you're cutting over to new processes. I've especially seen this come in handy when consulting in the past for a company in the power industry. Government agencies dictate how the business operates and when changes to their processes must be effective. Having effective and expiration dates are a must in this kind of environment.

The ActiveBPEL Engine is also smart about how it handles expiration of running processes. It gives the user a choice of terminating the process and allowing it to continue to run to completion, or migrating the process to the new version. Process migration sounds a bit dicey and the manual outlines its limitations, but I'm sure there will be users out there who won't be able to live without it.

Process Deployment

When describing my test, I glossed over the deployment process. The ActiveBPEL Designer is the recommended tool for deploying BPEL processes to the ActiveBPEL Engine. It packages the various files that make up the selected process in a business process archive (.bpr) file. The tool supports automatic deployment of the .bpr file to the engine. It can also generate an Ant script to re-deploy the file.

When you use ActiveBPEL Designer to create a .bpr file, the .bpr wizard generates a resource catalog. The data in the catalog is displayed in the engine's administration console. The catalog also provides a way for the engine to find WSDL files, schema, and other resources in the deployment archive.

Administration Console

As said above, the ActiveBPEL Engine provides the obligatory administration console, but again it's nicely done. You can inspect processes while they're running. The process page even displays the process details and the process diagram.

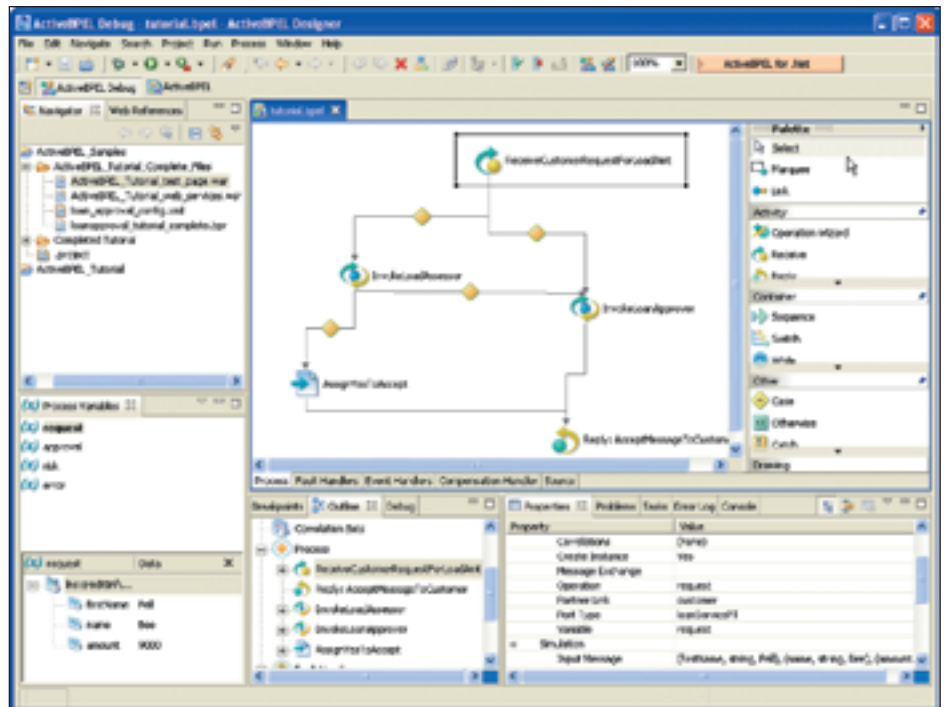
For users who run large process sets, there's a nice feature that filters processes. Active Endpoints even provides a nice expression builder to aid in creating fairly complex criteria for selecting the processes to view.

Custom Invocation

My favorite feature in the engine is its ability to implement a custom invocation handler for service endpoints. This lets a process bypass the creation and transport of SOAP messages over Axis. The process's outbound message is routed directly to a service.

Custom invocation handlers can provide a more efficient technique for communicating between processes and services on the same machine. It can also take advantage of an alternate SOAP engine or provide messaging to an application that's not exposed as a Web Service.

Custom invocation handlers are implemented as Java classes, EJB containers, or .NET assemblies that implement the engine's custom invocation interface. The BPEL process's deployment descriptor contains a custom invoke handler attribute for the partner role service. The custom handlers are loaded by ActiveBPEL and used to call out to a Partner Role service endpoint when the service is invoked in a BPEL process.



ActiveBPEL Designer

The bottom line with this feature is that you can co-locate chatty services and set up highly efficient messaging for high-volume processes. This is very cool, sort of a "pimp my ride" for the BPEL crowd.

Licensing

As stated in my previous review of the product, the ActiveBPEL Engine core is an open source product. The designer tool isn't, but it's still freely downloadable from the Active Endpoints Web site.

To get access to the high-end engine features like process versioning and clustering you'll have to license the Enterprise product. The cost of the Enterprise product varies based on the deployment platform and ranges from a low-end Tomcat-enabled version to a J2EE-enabled version that can run on a mainframe.

Support

Active Endpoints, Inc. offers a few support options. First, for open source users, forum- and mailing list-based support is provided. There's also a premium support forum for users with a valid product serial number. For enterprises with higher support demands, incident-based support can be purchased.

Conclusion

Active Endpoints, Inc. continues to stay out in front of the BPEL standard with

thoughtful features for support of WS-BPEL 2.0 migration. The BPEL design tool and execution engine continue to be freely downloadable and well documented with good community support. Teams looking to move toward the WS-BPEL 2.0 standard should consider Active Endpoints' 3.0 version. ■

About the Author

Paul T. Maurer is a principal at a leading consulting services company.

paul@paulmaurer.net

Product Info

Company

Active Endpoints, Inc.
Three Enterprise Drive
Shelton, CT 06484

Web: www.active-endpoints.com

E-mail: info@active-endpoints.com

Licensing Information

ActiveBPEL Designer: Free
ActiveBPEL Engine (Open Source): Free
ActiveBPEL Enterprise: \$7.5k - \$30k per CPU
based on deployment platform

Testing Environment

OS: Windows XP Professional (Service Pack 2)
Hardware: Intel Pentium M Processor with
2GHz - 1GB RAM

En Masse SOA Enablement Methodology Distilled

Part II: The design/build/test part of the methodology

WRITTEN BY PAUL O'CONNOR

➤ Part I of this series observed that 2006 was the year in which many large-scale SOA projects were kicked off in medium and large enterprises. Which means that an all-encompassing methodology that could evolve SOA to the enterprise was needed. Much has been written about service analysis and design for SOA but not so much about methodologies for creating an all-encompassing SOA, including the infrastructure – and, after all, it's the SOA infrastructure that supports service interactions and agility through the change management of services and metadata.

I posited in my first article that SOA derives its powerful complexity management traits from the tried and true divide-and-conquer approach to problem solving. I then set about trying to distill a “little-m” analysis method for en masse SOA enablement guided by this principle, which mines infrastructure requirements by extending existing service analysis methods, most notably IBM SOMA. Analysis results were said to be a set of system requirements and a set of infrastructure requirements, products of the same business domain analysis (and perhaps re-engineering) iterations.

System requirements in SOA specify applications and supporting business and component services while infrastructure requirements specify non-functional aspects, including all metadata management facilities. Part I focused exclusively on the creation of the infrastructure requirements set (the “enablement” of the enterprise for SOA), reflective of an overarching vision of business agility and efficiency. Also derived were three crucial data sets that will assist design greatly: A set of enterprise assets that should be reused/repurposed, a prioritization of infrastructure elements and services needed to expedite desired business functionality, and a view of the consumer base of the SOA and what it will take to rewire them to the new SOA.

We also have in hand (as part of the requirements set) a system change tax-

onomy...a categorization and specification of the metadata that must be changed in the new SOA, and its velocity, to support the business's overarching vision of agility. Inasmuch as Part I accepted service analysis methods axiomatically, this article focuses on an infrastructure creation method and only references service design techniques, though ensuring that services perform well and are policy-compliant is within the purview of any SOA infrastructure.

SOA analysis and design is iterative. Once service and infrastructure specification reaches a point where the requirements sets are specified enough (“enough” very much depends on the specific enterprise and level of complexity), then design of specific services and SOA infrastructure can join the iterations, with service design following infrastructure design to the extent that services depend on the infrastructure specification. The aptness of the divide-and-conquer paradigm for SOA enablement shows up all the more in the design phase. In fact, with respect to the infrastructure it can be said that it's the overarching design principle.

SOA standards and best practices like loose coupling and agility via metadata management mean that a meet-in-the-middle approach to design can be taken as long as strong governance is in place to ensure that all parties adhere to specifications, standards, and best practices, and that



everyone and every task is accountable. Being able to divide up complex design tasks into manageable subtasks and build out the SOA infrastructure while also building core services, and even starting an application on top of it all is the core competency of en masse SOA design.

I will now layout a series of broad work steps that should lead you to the end-point of having a future-state architecture plan and associated roadmap in hand. And I will conclude by folding in build-and-test cycles.

Create an SOA Design Center

It stands to reason that before we can send architects, developers, managers, and administrators off in different directions to design (and/or repurpose) our enterprise infrastructure components for SOA we need a way for individuals and teams to collaborate and align as they advance. Creating an SOA design center (SDC), an extension of the tried and true SOA center of excellence that adds collaboration and oversight capabilities is the way to achieve this.

In its collaboration aspect the SDC serves to allow team members to quickly find and exchange information crucial to their efforts. This will be relevant not only for infrastructure designers, but for service developers, consumers, and analysts alike. In its oversight aspect, the SDC serves as the crossroads of the project management and

enterprise architecture disciplines. In a divide/conquer approach to a large enterprise build-out, it's not possible that all design decisions route through the old enterprise architecture council for approval.

Yet, governance of the design process is no less important. Likewise, project managers can't be expected to capture such an effort in a monolithic project plan, even though the management discipline is just as important. By allowing the design process to de-federate from strictures that artificially impede it you'll both empower the designers and the governance regime. In a sense, it's analogous to the governance of services themselves – by separating services from policy enforcement we empower both service designers and policy administrators. By eliminating hurdles and gates that would otherwise impede architects as they design and specify infrastructure elements, progress is quicker, even while oversight and administration is better.

How you implement the SDC and what tools you use are entirely dependent on the scope of your effort and your enterprise's architectural governance sensibilities. Seed the SDC with the asset map and change taxonomy you created in the analysis phase, and use it as the storehouse for design artifacts including the future-state architecture plan and roadmap. Consider coupling the SDC with a service design repository and solution repository to render a complete view of how your enterprise will meet its business objectives.

One caveat: Sending designers off in different directions on the premise that fewer strictures will empower them as they design individual piece parts of the larger SOA means that there's great onus on the testing regime to ensure the sum of the parts meets the holistic enterprise requirements you have derived.

Infrastructure Chemistry & Preliminary Design Choices

When you chose to organize your enterprise around services and their assembly under a metadata-driven management regime to build business solutions you are subscribing to a well-defined infrastructure reference model. The SOA infrastructure reference model has matured over the last couple of years, and continues to mature, but it has its roots at the service protocol level where policy is enforced by intermediaries against XML data, most often in SOAP headers and payloads. Everything in the reference model is about ensuring that the right services accurately answer

the right requests with the right policies having been applied accurately and within the expected timeframe. What's "right" is communicated in the form of metadata, administered in the SOA infrastructure. Achieving such runtime governance of the service-based enterprise, and especially metadata change management facilities, is the major thrust of the SOA infrastructure design effort.

By mixing in some basic design assumptions and analysis artifacts from the infrastructure requirements set, concrete design choices are precipitated from the reference model, just like adding a precipitant to a solution in a chemistry lab experiment. For example, if you know from analysis that your business services will comprise new component data services and legacy EAI services, you might specify an SOA data services platform and an ESB for orchestration and legacy integration. On the other hand, if you're only creating business services from a portfolio of SOAP Web Services that are WS-I-compliant, you may want to pencil in a pure-intermediary governance solution with a standalone orchestration environment. Start by capturing these design decisions on a whiteboard and publish them as schematics to the SDC when they've matured from the "what if we did this" point to the "we should probably do this" stage. Feel free to publish several competing design choices...this is a best practice that focuses attention on the relevant pros and cons of different design concepts. Consider holding a preliminary design review (or several...or weekly) with the enterprise architecture team to exchange ideas verbally. The process of iterating over competing design choices (and systematically eliminating entire choices that don't measure up over time) should lead you to a small number of preliminary designs (or perhaps one).

For example, if you begin to factor in security policy enforcement requirements you might add policy enforcement points, agents, and a policy repository to your design choices. This might cause you to devalue a monolithic ESB-centered design and favor an infrastructure designed around intermediaries and policy management. Likewise, if your policy enforcement requirements are light, you might favor the ESB-centered design more. Note that you shouldn't make specific technology decisions at this point; evolving the preliminary design into a detailed design and making associated technology decisions includes several additional factors.

THREE REASONS TO

blog-n-play.com

1 Get instantly published to 2 million+ readers per month!

blog-n-play™ is the only **FREE** custom blog address you can own which comes instantly with an access to the entire i-technology community readership. Have your blog read alongside with the world's leading authorities, makers and shakers of the industry, including well-known and highly respected i-technology writers and editors.

2 Own a most prestigious blog address!

blog-n-play™ gives you the most prestigious blog address. There is no other blog community in the world who offers such a targeted address, which comes with an instant targeted readership.

3 Best blog engine in the world...

blog-n-play™ is powered by **Blog-City™**, the most feature rich and bleeding-edge blog engine in the world, designed by Alan Williamson, the legendary editor of **JDJ**. Alan kept the i-technology community bloggers' demanding needs in mind and integrated your blog page to your favorite magazine's Web site.



www.TAMI.linuxworld.com

"Many blogs to choose from"

PICK YOUR MOST PRESTIGIOUS ADDRESS

IT Solutions Guide	MX Dev. Journal
Storage+Security Journal	ColdFusion Dev. Journal
JDJ: Java	XML-Journal
Web Services Journal	Wireless Business &Tech.
.NET Dev. Journal	WebSphere Journal
LinuxWorld Magazine	WLDJ: WebLogic
LinuxBusinessWeek	PowerBuilder Dev. Journal
Eclipse Dev. Journal	

3 MINUTE SETUP

Sign up for your FREE blog Today!



there's great onus on the testing regime to ensure the sum of the parts meets the holistic enterprise requirements you have derived

Repurpose, Rehash, and Reinvigorate

You might recall that a key part of the analysis phase for en masse SOA enablement was to identify existing enterprise assets that should be included in the new SOA. This includes both services (maybe not compliant Web Services) and enterprise aspects like identity and access management systems. This list needs to be segmented into assets that can “easily” participate in the new SOA – they’re easy to include if a new version exists, or a delegate or mediator can be added so that they can quickly meet your SOA service or metadata standards – and those that aren’t readily repurposed. Examples of easily repurposed assets are databases that can be introspected and service-enabled quickly or EAI-type interfaces that can be facaded by an ESB.

Notice I said easily, not cheaply...a value judgment will have to be made about what to bring in and at what cost. This is an example of a decision that should bubble up through your SOA design center and get kicked around. Other assets that were tagged for reuse may not be so easy to get at. They’ll have to be reinvigorated by changing their core aspects or otherwise engaging in a project to repurpose them. Examples of such problematic assets include business rule repositories and policy repositories that may be scattered across the enterprise and that may need consolidation, upgrade, or outright replacement. Each preliminary design choice should clearly identify the assets being reused, even if they appear as a black box, for example, “SAML Token Provider,” to encapsulate your identity management system. And over the course of design iterations, feel free to rehash the decision of reusing assets if it’s clear it adds risk or undue cost. Such rehashing is vital feedback for the more advanced analysis iterations that are feeding design.

Create Ontologies of Change

From the analysis phase, one of the key artifacts of the infrastructure requirements set was the change taxonomy, a categorization of everything in the system that has

to change (and at what speed) to accommodate the overarching business agility vision of the enterprise. SOA agility is easily understood in the abstract, e.g., “we could increase revenue and profit if we could interact with partners more easily,” but it’s impossible to do system design from such nebulous statements. For this reason, the analysis phase is particularly interested in enumerating the metadata that has to change for the system to be agile: rules, policies, service contracts, service compositions, transforms, etc.

The design phase should further this effort by creating ontologies of change – abstract models that detail who changes what metadata in the system and what effect the change has. Even though these models will be abstract at first, it will be apparent what changes are needed in the infrastructure to meet the requirements. By way of example, if a requirement has been expressed for security administrators and business managers to coordinate on managing federation partner access control policies, an ontology should be developed that includes all three actors (security administrator, business manager, and partner) and that includes system elements with which they interact to manage partner access.

The ontology should include a functional use case that captures the system interactions and particularly the system response time. If that system is expected to support partner onboarding a day from certificate exchange and suspension within one minute of revocation, this should be detailed in the functional use case. Such agile partner management requires a more intricate infrastructure design than a much slower (and perhaps manual) response.

Once the ontology has been fleshed out, write a “day in the life” scenario for each and every participant that the system will impact, publish it to the SDC, and solicit feedback from those involved from the business and technical teams. Once the ontology is finalized, it can be folded into the preliminary design choices and the infrastructure expanded to accommodate it, if need be. Look for synergies between ontologies to lessen the impact on the infra-

structure of accommodating change, e.g., a single metadata repository to handle all service policies or a single administration portal for metadata management.

Integrate, Compose, Transform, Mediate

Once design is complete to the extent that it’s clear how services are to be assembled and governed as dictated by metadata managed at sufficient change velocity to be agile, an integration view of the design can be considered. Such a view is concerned with the edge conditions in the enterprise that break the nice whiteboard view of the architecture where integration is as easy as drawing lines between boxes. The reality is often much different. Not only might the SOA infrastructure be expected to orchestrate services of differing protocols, standards, interpretations of standards, operations, and schemas, but it might even have to bridge different message exchange patterns (synchronous/asynchronous). Metadata formats may also need to be bridged, e.g., security token formats, access control policy formats, or service compositions. Designing for mediation is especially important in large enterprises where silo’d applications and service domains are most prevalent, along with entrenched enterprise service groups who provide security, management, and other orthogonal functions. And preparing to integrate new domains under a merger/acquisition scenario, or just expanding to new lines of business will serve to future-proof the infrastructure design.

As described in the analysis phase, it may also be necessary to bridge data domains by transforming and/or composing sub-domain service data into a unified data service. For example, a ‘getCustomer’ service could well comprise several sub-domain services by composition and a transform to a canonical form. As a best practice in SOA infrastructure design, abstract and encapsulate technical details for all forms of integration to the extent possible. For data integration this would mean leveraging an EII platform, where details of ‘getCustomer’ and the like can be managed

Other companies in this magazine spent a lot of time on pretty ads. As you can see, we did not. We spent our time hiring the best people and training them to deliver outstanding support for your website. We spent our time building a state of the art datacenter and staffing it with people who care about your website like it's their own. Compassion, respect, credibility, ownership, reliability, "never say no," and exceed expectations are words that describe our service philosophy. From the first time you interact with us, you'll see what a difference it really makes. And you'll also forgive us for not having a pretty ad.



WEB HOSTING • MANAGED DEDICATED SERVERS • COLOCATION • VPS • ECOMMERCE • BLOGGING • EMAIL

as a black box. Ensure the design meets integration requirements, both implicit/future as well as those enumerated in the analysis phase.

Design for Consumption

A key topic of en masse SOA analysis was the focus on service consumption and how to ease the transition of service consumers away from local policy enforcement, and how to ease service and metadata changes in the UI and federation tiers. It's an unfortunate fact of life for user interfaces that authorization code is usually kept in portal repositories and, worse yet, hard-coded in the backing code itself. And service interaction code and screens are usually no more refined themselves. When migrating the enterprise to SOA, we can't hope to achieve agility and enhance reuse without easing the plight of interface designers and developers. We can go a long way towards achieving this end by accomplishing two main objectives:

1. *Implement* centralized UI policy management
2. *Create* reusable business service consumers

Step one in achieving these objectives is to create and implement a portal strategy. The maturity of the implementation of remote portlet standards (specifically WSRP) makes it practical and advisable to de-federate portlet design, development, and production, enabling the end-point aggregation of the portlets for specific applications. WSRP is an XML language with a SOAP protocol binding whose portlet producer interfaces are expressed with WSDL. By putting an intermediary between the WSRP producer and the consumer application, content-aware authorization policies can be enforced, as well as aspects like SLA monitoring and management. Leveraging the SOA infrastructure to enforce fine-grained, content-aware access control policies not only lessens the burden on UI developers but greatly eases the burden of changing policies.

Standardizing on a portal architecture also enables portlets to be created and governed right alongside the business services they surface. For example, if a bank has a business service that returns a customer's account history, the portlet that does the actual display should be created and governed right alongside. Portlet preferences can be used to tailor the display characteristics, e.g., which ledger columns to display, and also for inter-portlet communication, e.g., which account to display. Under such a scenario user inter-

face design and management becomes an order of magnitude less complex and costly. To achieve this end, the SOA infrastructure (and associated portal infrastructure) must be designed with these facilities in mind.

Generate a Detailed Design

There will come a point when it's time to move from the preliminary design choices to a detailed infrastructure design that may codify one of the preliminary choices or may actually be a hybrid of more than once choice. In either case, it's the detailed design that communicates the future-state of the infrastructure. It's the future-state infrastructure specification that will be coupled with the service development and governance plan to form the totality of the future-state architecture specification for the new SOA enterprise.

It goes without saying that the detailed design should have as much detail as possible, but it particularly needs to have additional views of the infrastructure that communicate to every role in the enterprise what the new architecture will mean to them. Operations teams, security administrators, business admins, and solution architects all need their own views. And the technology selection process needs to start at this time. We'll see next how a process of technology validation should feedback and finish this specification. Feel free to use the tried/true bake-off due diligence process to this end. Also, a roadmap needs to be developed based on prioritization of business needs specified in the infrastructure requirements set. The roadmap details individual projects that must be completed, and in what order, to achieve the quickest and most comprehensive solution to the business case. Leverage the SDC to communicate and collaborate on these final design aspects with the broad team, most notably the business and technical owners of the project.

Build, Test, and Validate

The design of any system is nothing without the validation that comes from a well-conceived build-and-test cycle, and certainly SOA infrastructure development is no different. Build and test should begin as soon as the detailed design stage has begun to specify technology choices. Start by setting up a single test environment and begin to add new components as they are specified. Infrastructure testing is paramount in SOA since so much of the system's functionality is derived from the infrastructure. Write


as many test cases as possible, leveraging realistic test services and infrastructure components to validate technology and design choices with respect to performance, quality of service, and ease-of-use assumptions. Feed the results back to the detailed design process to assist with technology selection. Next, begin the process of building a production environment. The test cases already written should form the basis of a production monitoring and testing regime. Detail test cases in the SDC and publish results where they can provide closed-loop feedback to concerned team members.

Summary

2006 has been a year in which large-scale SOA projects have been appearing with increasing regularity. As such, a need for an all-encompassing methodology for infrastructure development was derived to complement existing service development methods. The first article in this series focused on analysis for en masse SOA enablement with the artifacts of the analysis being a system requirements set that specifies services, and infrastructure requirements that specify non-functional aspects. In this article a design method was prescribed that began by urging that an SOA development center (SDC) be created to act as a repository for design artifacts, as well as a collaboration and oversight point. Hardcore design began with the precipitation of preliminary design choices from the well-established SOA infrastructure reference model. Next, a sub-method for rehashing and repurposing existing enterprise aspects for the SOA was espoused, followed by the important design step of developing ontologies of change. Integration design was discussed, with an emphasis on mediation and data composition, followed by a treatment of user interface design best practices and how they affect SOA infrastructure. The article concluded by specifying the creation of a detailed design that leads to the end-goal of a future-state architecture specification and roadmap. The build/test phase was addressed and tied back into the iterative design process. ■

About the Author

Paul O'Connor is chief architect and SOA Practice managing principle for e-brilliance LLC (a leading NE SOA consultancy), and is currently doing major SOA architecture and implementations for Fortune 100 clients across the US. Previously he was chief architect for Damascus Road Systems, specializing in security architecture.
poconnor@e-brilliance.com



AJAX ONE-DAY HANDS-ON INTENSE TRAINING!

ALL ATTENDEES RECEIVE:
Certificate of Completion **PLUS** Course Materials on DVD!*

AJAXWorld University's "AJAX Developer Bootcamp" is an intensive, one-day hands-on training program that will teach Web developers and designers how to build high-quality AJAX applications from beginning to end. The AJAX Developer Bootcamp is intended to be the premier AJAX instructional program presently available anywhere.

AJAXWORLD UNIVERSITY BOOTCAMP

www.AJAXBOOTCAMP.sys-con.com

Roosevelt Hotel
New York, NY
January 22, 2007

Roosevelt Hotel
New York, NY
March 18, 2007

What You Will Learn...

Overview of AJAX Technologies

- HTML vs. DHTML
- Network Concerns
- Asynchronous Conversations with Web servers
- The characteristics of high-quality AJAX applications
- The Web page is the application
- What the server provides
- User interaction

Understanding AJAX through the basics of AJAX

- Asynchronous server communication
- Dynamic HTML
- Javascript Design patterns
- User interface strategies for building elegant, highly addictive Web sites and applications
- The Essential AJAX Pieces
 - Javascript
 - Cascading Style Sheet (CSS)
 - Document Object Model (DOM)
 - XMLHttpRequestObject
- The AJAX Application with Javascript
- Using CSS
- Structuring the View Using the DOM
 - Applying Styles with Javascript
 - Communicating with the Web Server in the Background
 - Designing AJAX Applications
 - Design Patterns
- Introduction to AJAX Frameworks
 - Dojo, script.aculo.us, Prototype
 - Over of framework capabilities
 - Examples of frameworks in use
 - Best Practices

Hand-On Development The Fundamentals:

Building the Framing for an Ajax Application

- Review the courseware code with the Instructor
- Begin building a working AJAX application and start applying technique and technologies as introduced in class
- Create the basic AJAX application by creating HTML, Javascript, and CSS files
- Learn Best Practices and Validation
- Learn and add script.aculo.us effects
- Learn and add the Dojo Framework

Adding Basic Ajax Capabilities to a Web Page:

Going Deep Into the AJAX User Experience

- Elements on the Rich Internet Experience
 - Interactivity
 - Robustness
 - Simplicity
 - Recognizable Metaphors
 - Preservation of the Browser Model Bookmarks/Back Button
- Background operations
- Building a AJAX Notification Framework
- Provenance and Relevance
- Rich Experience Support with Third-Party AJAX Client - Framework
- Using AJAX layouts, containers, and widgets
- Patterns for Animation and Highlighting
- User Productivity Techniques
- Tracking Outstanding Network Requests

Hands-On Development:

Expand the Application with more Advanced Ajax

- Review the courseware code with the Instructor
- Expand the Mural Application
- Add Features using Dojo
- Add specific Dojo Libraries to support Ajax widgets

Work on server side communications in the background

- Create a tabbed layout
- Create a submission form to upload to the server, all without reloading the page
- Create an Ajax submission form that will take uploads on one tab
- Created a form validation that ensures only the right information is submitted.
- More Stretch work for those who want to learn additional concepts

Advanced AJAX Concepts

- Review Ajax Concepts
- SOA and Mashups
- Current state of Ajax Frameworks
- Web 2.0 and the Global SOA
- Ajax Constraints
- Design Patterns
- Javascript Timers
- Ajax Programming Patterns
- Performance and Throttling

Hands-On Development:

Working with Advanced Ajax Capabilities

- Review the courseware code with the Instructor
- Work with the Accordion control
- Learn how to use the Tree control
- Explore Dojo's animation capabilities
- Explore how the debug output can be used in <div> elements
- Tour Dojo and RICO demos
- Experiment with new Dojo features from the Dojo demos
- source code and attempt to add them to various parts of the Mural application
- Overview of Future of AJAX and Rich Internet Applications

What Attendees Are Saying...

“The trainer was excellent.
The material too!”

“The hands-on, although long,
was useful and educational!”

“The instructor was good. He
answered questions thoroughly!”

“Well designed and organized.
Good mix of lecture vs lots of
hands-on!”

*ALL RELEVANT COURSE MATERIALS WILL BE OFFERED ON DVD AFTER THE EVENT

HURRY! REGISTER NOW FOR EARLY-BIRD DISCOUNT!



For more great events visit www.EVENTS.SYS-CON.com



Learning from Mistakes

A guide to SOA anti-patterns – how to benefit from known unworkable solutions

WRITTEN BY NTHONY CARRATO, DR. HARINI SRINIVASAN, AND DR. CHRIS HARDING

➤ Today, global businesses are increasingly turning to Service Oriented Architecture (SOA) for their information technology infrastructure and applications. SOA is becoming an increasingly practiced approach to building software solutions as it can support integration and consolidation of activities in complex enterprise systems.

This year the percentage of functioning SOAs has almost doubled, according to Evans Data Corporation's latest Web Services development survey: "Twenty-four percent of respondents (the sample data set being 400) are saying they currently implement SOA, an 85% increase from last year." The survey also shows that Web Services are now experiencing more comprehensive implementation, with 30% of respondents predicting they will be using more than 20 services in the next year – a 58% increase from current levels. Twenty-five percent of respondents also said the leading problem in implementing Web Services are changing or lack of industry standards, a 67% increase from last year. A July 2006 article in SDA Asia Magazine reported that "The awareness of ser-

vice-oriented architecture benefit as an IT enabler is now extensive. There is a growing and widespread acceptance of SOA technology, especially in large enterprises." And according to a recent Aberdeen Group survey, at least 90% of companies are on their way to adopting SOA today."

Despite these reports, SOA is still a relatively new approach, which many businesses are deploying for the first time. How can they avoid the pitfalls that are inherent in treading on any new technology path? One answer is to share the experience of the IT community. In the few years of SOA adoption, a number of best and worst practices have already emerged, and these lessons learned map not to just application (code) development but also to other phases of SOA solution construction.

What Is SOA?

The Open Group defines SOA as an architectural style that supports *service orientation*, a way of thinking in terms of services, service-based development, and the outcomes of services where a *service*:

- Is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit, provide weather data, or consolidate drilling reports);
- Is self-contained;
- *May be* composed of other services; and
- Is a "black box" to consumers of the service.

An *architectural style* is the combination of distinctive features in which architecture is performed or expressed. The SOA architecture has the following distinctive features:

- It's based on the design of the services – which mirror real-world business activities comprising the enterprise (or inter-enterprise) business processes.
- Service representation uses business descriptions to provide context (i.e., business process, goal, rule, policy, service interface, and service component) and implements services using service orchestration.
- It places unique requirements on the infrastructure – it's recommended that implementations use open standards to realize interoperability and location transparency.
- Implementations are environment-specific – they're constrained or enabled by context and must be described in that context.
- It requires strong governance of service representation and implementation.
- It requires a "litmus test" that determines a "good service."

SOA projects are complex in that they span layers from business goals and processes down to low-level technical implementation details as described in the excellent paper by Ali Asranjani and illustrated in Figure 1. Successful SOA implementation is hard. The key points to remember are:

- Build an internal agreement for using a SOA as an approach for an initial project
- Start with something useful but manageable
- Business and IT need to agree on an area of focus
- Plan for governance and skills development
- Plan to deliver success early and control costs

A useful approach is to have an initial exercise to scope the work and the first project followed by, concurrently, i) governance development, ii) competency development, and iii) an initial manageable project. However, even when this approach is adopted, there can be problems. Many groups in an organization are involved in building the solution for a SOA project. In addition, because SOA is so new, it's almost guaranteed to involve technologies and techniques that organizations have inadequate experience of. This combination solution involves much of an organization and the requirement to take on many new technologies and methodologies causes several opportunities for trouble, many of which are known and can be avoided with good planning.

Patterns and Anti-Patterns

Architects know how to use patterns that represent known workable solutions to given problems to develop the solution architecture. But SOA projects can also benefit from anti-patterns, or known unworkable solutions. Anti-patterns are the design choices to avoid and, while people can often learn more from failures than successes, they are less frequently discussed. In fact, knowing about anti-patterns can be useful across many project roles, including project managers, analysts, and developers as well as architects.

A *pattern* is a known-to-work solution to some problem. An excellent analysis by

Jonathan Adams and others identifies several levels of patterns:

- **Business patterns** that identify the interaction between users, businesses, and data.
- **Integration patterns** that tie multiple business patterns together when a solution can't be provided based on a single business pattern.
- **Composite patterns** that represent commonly occurring combinations of business and integration patterns.
- **Application patterns** that provide a conceptual layout describing how the application components and data in a business pattern or integration pattern interact.
- **Runtime patterns** that define the logical middleware structure supporting an application pattern. Runtime patterns depict the major middleware nodes, their roles, and the interfaces between these nodes.
- **Product mappings** that identify proven and tested software implementations for each runtime pattern.
- **Best-practice guidelines** for design, development, deployment, and management of e-business applications.

A pattern specification typically includes the name of the pattern, the problem solved by the pattern and forces that apply, a description of how the pattern works, a diagram (e.g., a sequence diagram) that illustrates the pattern, an implementation section (e.g., a UML model) and, finally, examples of how the pattern can be used.

An *anti-pattern* is an attractive apparent solution to a problem that is known not to work. The term was originally used to refer to a design pattern gone wrong. More precisely, an anti-pattern specification describes a *poor* solution to a problem, explains why this solution looks attractive and why it turns out to be bad, and offers better solution(s) to the same problem.

Anti-pattern descriptions provide a common vocabulary and an awareness of pitfall situations. They identify the symptoms, consequences, root causes, and potential replacement solutions for anti-patterns. An operational definition of an anti-pattern involves specifying the *problem*, the *symptoms*, the *root-cause(s)* of the problem, the *consequences* of building a solution using the anti-pattern, and a *solution* that presents a remedy to using the exposed anti-pattern.

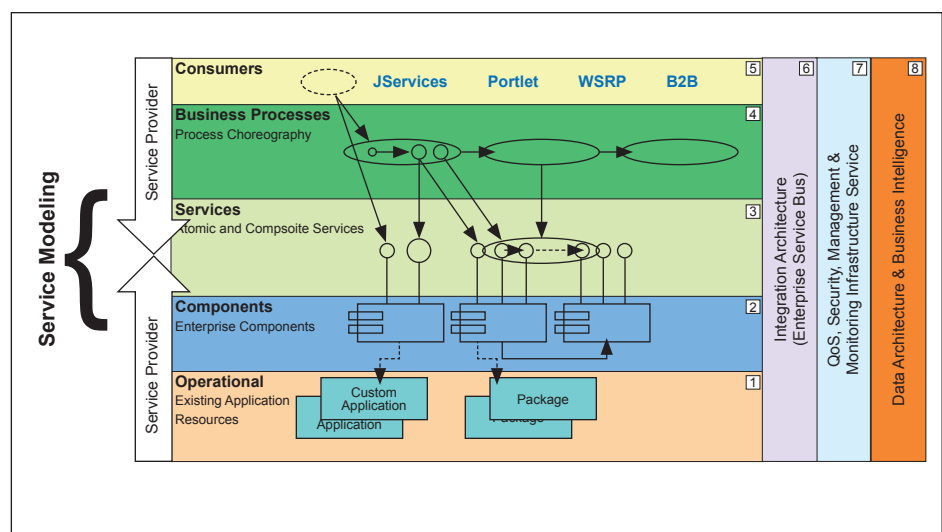


Figure 1: A SOA is composed of processes, services, and components. At the heart of a SOA is the service model that defines services and components that realize them

Example Anti-Patterns for SOA

To illustrate the concept, let's look at a few SOA anti-patterns in detail equating SOA with Web Services, chatty services, and point-to-point services.

Example SOA Anti-Pattern: Equating SOA with Web Services

A common mistake made by architects of SOA solutions is to equate a SOA solution to one that simply has Web Services. If the Web Services aren't aligned with the business reason why a SOA solution was desirable in the first place, the solution isn't *service-oriented*. The following is a description of the anti-pattern using the terminology described above:

- **Problem:** Equating SOA with Web Services
- **Symptoms:** Symptoms of this anti-pattern include replacing existing APIs with Web Services *without* proper architecture and when services aren't business-aligned.
- **Consequences:** The consequence of adopting this bad practice is the eventual proliferation of Web Services. It's also likely that those existing systems won't have interfaces that reflect the requirements of the service requestors. When services aren't aligned with business processes, the resulting solution greatly reduces the opportunities for reuse, for example, in composing new processes using existing services.
- **Root Cause:** The main cause of this anti-pattern is *haste*, i.e., an architect assuming "I can take shortcuts and do this SOA stuff quickly and cheaply." This also results from ignorance in the area of transforming an IT architecture to a Service Oriented Architecture that's driven by business needs.
- **Solution:** The solution is to apply proper service modeling techniques such as IBM's Service-Oriented Modeling and Architecture (SOMA) methodology, which will help in effectively mapping the business processes to SOA by identifying and specifying services. This approach (shown in Figure 2) facilitates the development of a viable transition plan from the current architecture to a Service Oriented Architecture with a defined SOA value proposition. Education on the difference between a Service Oriented Architecture and Web Services is also important. Most of today's production Web Service systems aren't based on Service Oriented Architectures – they're based on simple remote procedure calls, or point-to-point messaging via SOAP, or well-structured integration

architectures. Most of today's Service Oriented Architectures not only use Web Services – they also use mature technologies such as FTP and batch files. Asynchronous messaging and Web Services technology are principally used in newer systems being integrated into a SOA. Hence, Web Services can be only *part* of the answer.

Example SOA Anti-Pattern: Chatty Services

Another commonly seen anti-pattern is to realize a business service by implementing a number of Web Services where each Web Service exchanges a tiny piece of data with the others. We refer to such services as *chatty services*. Figure 3 shows an example of chatty services between the presentation and business layers of the SOA architecture. The following is a description of this anti-pattern:

- **Problem:** Realize a service by implementing a number of Web Services where each communicates a tiny piece of data.
- **Symptoms:** The symptom of this bad practice is that the implementation will eventually have a large number of services.
- **Consequences:** Chatty services result in degradation in performance and costly development. This anti-pattern also burdens the consumer with the task of aggregating these services to realize any benefit.
- **Root Cause:** The root causes of such an implementation are mimicking an API implementation, and being so excited by the service concept that *everything* becomes a BPEL-defined business service, with no benefit and excessive cost. Chatty services can also be a consequence of too rigid service interfaces. This typically happens when client/server technology is simply being replaced by services with rigid interfaces in the process of transformation to a Service Oriented Architecture. Such a naïve transformation to SOA typically introduces operations that haven't been modeled with variability, resulting in a multiplication of operations and hence chatty services
- **Solution:** The most effective solution is to refactor the design to combine individual pieces of data into one document. It's important not to use fine-grained calls. For example, to get the employee data that includes the employee's first name, last name, and office phone number, it would be a bad practice (that results in chatty services) to invoke a Web Service for each of these three elements. A better solution is to encapsulate the first name, last name, and office number in a single document and invoke a single Web Service to retrieve the employee data.

At the heart of SOMA is the identification and specification of services.

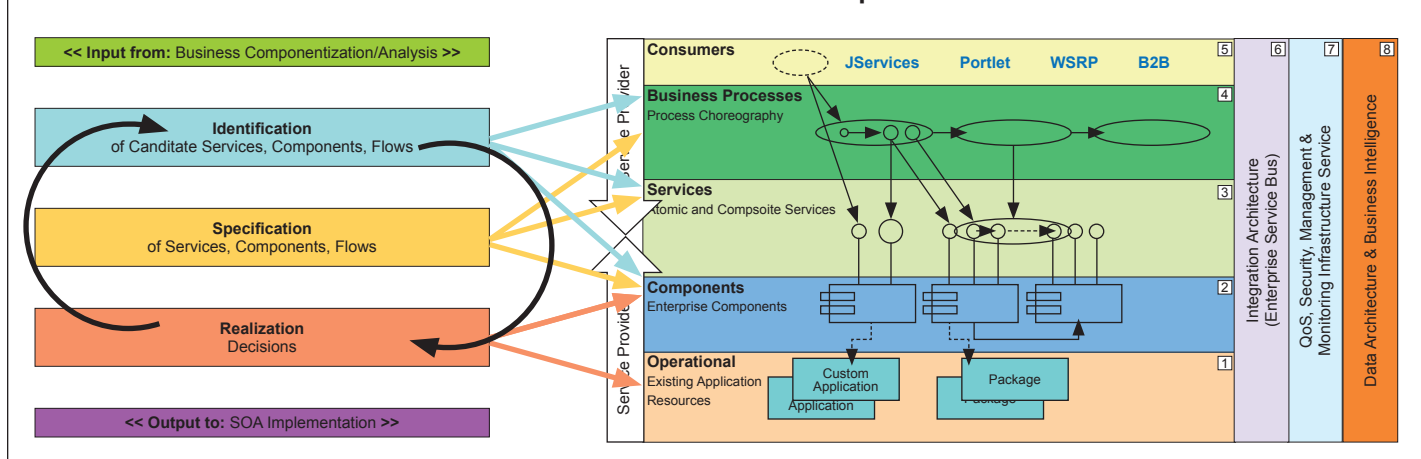


Figure 2: SOMA service modeling.

REGISTER TODAY AND SAVE!

Rich Internet Applications: AJAX, Flash, Web 2.0 and Beyond...

www.AjaxWorldExpo.com

AJAXWORLDTMEAST

CONFERENCE & EXPO



NEW YORK CITY

THE ROOSEVELT HOTEL LOCATED AT MADISON & 45th

**SYS-CON Events is proud to announce the
AjaxWorld East Conference 2007!**

The world-beating Conference program will provide developers and IT managers alike with comprehensive information and insight into the biggest paradigm shift in website design, development, and deployment since the invention of the World Wide Web itself a decade ago.

The terms on everyone's lips this year include "AJAX," "Web 2.0" and "Rich Internet Applications." All of these themes play an integral role at AjaxWorld. So, anyone involved with business-critical web applications that recognize the importance of the user experience needs to attend this unique, timely conference – especially the web designers and developers building those experiences, and those who manage them.

BEING HELD MARCH 19 - 21, 2007!

We are interested in receiving original speaking proposals for this event from i-Technology professionals. Speakers will be chosen from the co-existing worlds of both commercial software and open source. Delegates will be interested in learning about a wide range of RIA topics that can help them achieve business value.

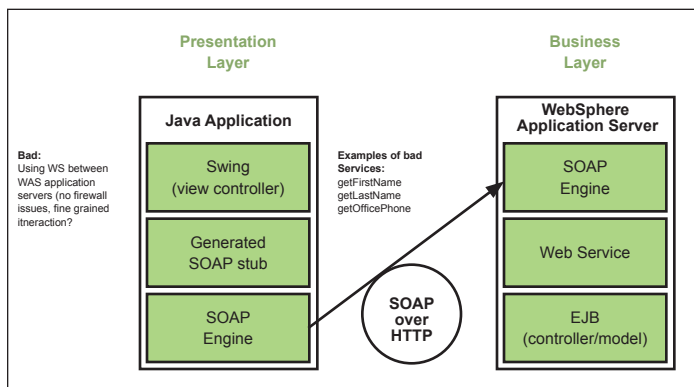


Figure 3: Chatty Services Example

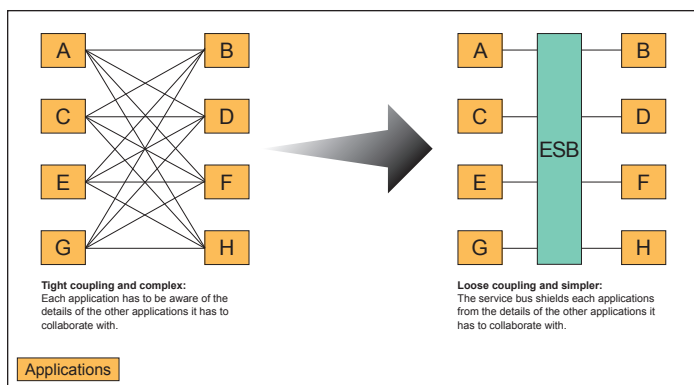


Figure 4: Point-to-point integration versus Enterprise Service Bus

Here again, education on the difference between API and service and on the granularity of services is important. A recommendation to identify the exposed services is to apply litmus tests on the candidate services. A *litmus test* for a service is one that will answer all of the following questions:

- Does the service provide a required unit of business functionality that supports business processes and goals?
- Does the service promote composability with other services?
- Can the service description be externalized?
- Does the existence of the service help in eliminating redundant implementations, i.e., promote reuse?
- Does the candidate service have a long enough lifecycle to be considered useful?

Example SOA Anti-Pattern: Point-to-Point Services

Implementing point-to-point services is an alternative to having a middleware layer that will route the service calls to service implementations. In such an approach, each requestor must individually access the corresponding service provider. Such practice is an anti-pattern and the following description of the anti-pattern explains why:

- **Problem:** Replacing middleware with point-to-point Web Services as an integration approach.
- **Symptoms:** Using XML or SOAP over HTTP between applications to effect communication between applications.
- **Consequences:** Point-to-point integration emerges as the de facto integration pattern. This pattern promotes an integration complexity of the order of $N*(N-1)$, where N is the number of services, resulting in error-prone management of the architecture.

- **Root Cause:** The root cause for this anti-pattern is usually a view that an integration layer, usually called an Enterprise Service Bus (ESB), adds:
 - Complicated new technology
 - A single point of failure
 - Cost (for the ESB software and supporting hardware)

In fact, an ESB can be a key component, which enables much greater flexibility and reduces the complexity of the solution by providing a common point for interconnection to services. ESB technology has also proven to be robust and scalable.

- **Solution:** The solution that scales and results in more manageability (with order N complexity) is to use an intelligent connector such as an Enterprise Service Bus (Figure 4). An ESB enables applications to work together simply and efficiently to support business needs. Point-to-point integration results in tight coupling where each application has to be aware of the details of all other applications with which it can potentially collaborate. In the Enterprise Service Bus case, on the other hand, the coupling is loose and simpler. The bus also allows an application to be unaware of the details of the other applications that it collaborates with.

Solution Levels

The examples described all relate to architecture/design but, as experience with SOA accumulates, anti-patterns are beginning to emerge at all levels of the SOA solution process. The following table shows an example at each level.

SOA solution level	Example(s)
<i>Project Management & Methodology</i>	Not planning incremental build cycles
<i>Modeling</i>	Not formally modeling business processes
<i>Architecture/Design</i>	Not understanding the full scope of included systems/interfaces in the solution Security can be added later after system design
<i>Assembly/Development</i>	Falling into the build-from-scratch rather than reuse existing services trap
<i>Performance Management</i>	Failing to start verifying performance early in the project
<i>Testing/QA</i>	Failure to test business process requirements
<i>Deployment</i>	Not planning for incremental and continuous deployments
<i>System Management</i>	Not allowing for SLA management at the process level
<i>Problem determination</i>	Not planning for problem determination across a process

Example SOA Anti-Pattern:

No Incremental Builds of the SOA Solution

This SOA anti-pattern is an example of a bad practice that can be exercised by a project manager:

- **Problem:** Assembling the complete solution, only late in the lifecycle.
- **Symptoms:** Each sub-team works independently but the whole end-to-end solution is only tested in a system integration test at the end of the development.
- **Consequences:** Point-to-point integration emerges as the de facto integration pattern. This pattern promotes an integration complexity of the order of $N*(N-1)$, where N is the number of services, resulting in error-prone management of the architecture.
- **Root Cause:** Problem determination and problem isolation are left until the system integration test stage, when they are very hard to do. Performance problems in individual sub-teams' efforts and in end-to-end integration aren't seen and resolved early.
- **Solution:** Build the solution in stages:
 - **Build 0:** Install the vendor-supplied software, e.g., middleware, and validate that all components install successfully. This will confirm that the required products have all been assembled and should also verify that patch level, operating system; disk space, and memory requirements have been met.
 - **Build 1:** Create a few simple end-end services. There should be no more than one or two, and it's important that they are truly end-to-end. An example might be to look up a customer, assuming that the customer database is part of the solution. This can be done from a portlet in the solution portal. It should exercise the portal, the ESB, the back-end database, and the services it ex-

poses. Ideally, this would also exercise the security subsystem, at least to the extent of validating a single initial user. This step will prove that all the solution components can work together or will highlight problems among them at an early stage in the project.

– **Build 2...Build n:** Build more solution elements, releasing them to testing in meaningful increments. A key part of a SOA is that it supports being able to assemble services into processes. This capability must be exercised early in the project to confirm that services are indeed composable into the required processes.

Capturing the Experience

So that all enterprises can benefit from the experience of early adopters of SOA, it's important that companies deploying SOA, companies providing SOA solutions, and companies providing tools communicate with each other. Participation in industry bodies and consortia can be an excellent way of doing that. Outside the commercial arena, free of the pressure to pick a product or make a sale, people can exchange ideas and discuss approaches to realizing SOA.

The Open Group is a vendor-neutral consortium whose vision is boundary-less information flow in and between organizations. It sees SOA as the prime architectural approach by which enterprises can realize that vision. Its SOA Working Group exists to develop and foster common understanding of SOA to facilitate alignment between the business and information technology communities. It does this by conducting a work program to produce definitions, analyses, recommendations, reference models, and standards to assist business and information technology professionals in and outside the Open Group to understand and adopt SOA. Participation

Looking to Stay Ahead of the i-Technology Curve?

Subscribe to these
FREE Newsletters >

Get the latest information on the most innovative products, new releases, interviews, industry developments, and i-technology news

Targeted to meet your professional needs, each newsletter is informative, insightful, and to the point.

And best of all – they're FREE!

Your subscription is just a mouse-click away at **www.sys-con.com**

IT solutions

NET JOURNAL

JDJ

WebServices

Information
STORAGE+
SECURITY
JOURNAL

LinuxWorld

wireless

LINUX BUSINESS WEEK

XML JOURNAL

MX developer's journal

WebSphere JOURNAL

wldj

COLOFUSION

SYS-CON MEDIA

The World's Leading i-Technology Publisher

in forums such as this helps people learn from others and capture common experience for the benefit of the wider community.

It's particularly important that anti-patterns are identified and described, and catalogued in a structured way. In the SOA Advanced Technology team at IBM, for example, an effort is under way to identify SOA anti-patterns from experiences with SOA customer engagements. The goals of this effort are, in addition to identifying the anti-patterns, to catalogue them for future reference, and to develop educational material on anti-patterns. And the practice of detecting anti-patterns at all stages of the SOA solution lifecycle is being enforced too.

Conclusions

SOA use is growing explosively. But when an enterprise uses any new approach, mistakes are likely, and SOA is no exception to this rule. For this reason, it's vital that the experience of the first adopters of SOA is captured for the benefit of all enterprises. Identification and documentation of anti-patterns – attractive apparent solutions that are known not to work – will help enterprises avoid mistakes and successfully implement SOA.

We hope that we have provided some useful thoughts on how to avoid known mistakes in a SOA project. Some of the most important lessons are:

- Be sure to include skills development in your plan.
- Plan for incremental releases, building on one another and testing as you go.
- Web Services are an important technology but, by themselves, they aren't SOA. SOA is an *architectural* approach first and foremost!
- Don't get trapped into thinking more services is a better solution. Having the right services is the better solution. Avoid anti-patterns like *chatty services*.
- Adding point-to-point connections doesn't help; look to the Enterprise Service Bus pattern as way to simplify your architecture.

These aren't the only lessons. There are many anti-patterns to avoid. We are continuing to do research in this area and plan to publish further articles on the topic and encourage you, the practitioners, to join in this work, and get involved with other people and organizations working on SOA.

Where To Learn More

The Open Group SOA Working Group provides a very active forum for staying abreast of recent SOA developments. The group intends its work to be highly accessible to participants at all levels. You can find out more about the Open Group SOA Working group at <http://www.opengroup.org/projects/soa/>.

Besides the SOA Working Group, there are many sources for pattern and anti-pattern information. Below we list some we believe will be particularly useful. ■

References

1. Jonathan Adams, Srinivas Koushik, Guru Vasudeva, and George Galambos. *Patterns for e-Business: A Strategy for Reuse*. MC Press. 2001.
2. D. Alur, J. Crupi, and D. Malks. *Core J2EE Patterns: Best Practices and Design Strategies*. Prentice Hall/Sun Microsystems Press. ISBN: 0130648841.
3. B. Dudley, S. Asbury, J. Krozak, and K. Wittkopf. *J2EE Anti-Patterns*. Wiley Publications. ISBN: 0471146153.
4. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, Addison-Wesley. 1995.

5. D. Hovemeyer and W. Pugh. "Finding Bugs is Easy." <http://www.cs.umd.edu/~pugh/java/bugs/docs/findbugsPaper.pdf>
6. D. Reimer, E. Schonberg, K. Srinivas, H. Srinivasan, B. Alpern, R. D. Johnson, A. Kershenbaum, and L. Koved. *SABER: Smart Analysis Based Error Reduction*. In *ISSTA*. 2004.
7. William J. Brown, Raphael C. Malveau, Hays W. "Skip" McCormick, and Thomas J. Mowbray. *Anti-Patterns: Refactoring Software, Architectures, and Projects in Crisis*. Paperback. March 20, 1998)
8. Norbert Bieberstein, Sanjay Bose, Marc Fiammante, Keith Jones, and Rawn Shah. *Service-Oriented Architecture (SOA) Compass: Business Value, Planning, and Enterprise Roadmap*. Developerworks. Hardcover. IBM Press
9. Olaf Zimmermann, Mark R. Tomlinson, and Stefan Peuser. *Perspectives on Web Services: Applying SOAP, WSDL, and UDDI to Real-World Projects*. Springer Professional Computing. Hardcover. ISBN: 3540009140.
10. Steve Jones. "SOA Anti-Patterns." CGI: <http://www.infoq.com/articles/SOA-anti-patterns>. Also check out Steve's blog @ <http://service-architecture.blogspot.com/> Search for "anti-patterns" or "patterns."
11. Ali Asranjani, *Service-oriented modeling and architecture, How to identify, specify, and realize services for your SOA*. IBM Developerworks. November 2004. <http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/>
12. SOA Survey article, http://www.govtech.net/magazine/channel_story.php/100246
13. "Growth in SOA and Web Services Implementation." SDA Asia. July 25, 2006. http://www.sda-asia.com/sda/features/psecom,id,459,srn,2,nodeid,21,_language,Singapore.html

About the Authors

Anthony Carrato is the executive IT architect for Asia-Pacific for the Enterprise Integration Solutions team in IBM's Software Group focusing on SOA delivery. In this role Tony is responsible for a team of IT architects who help IBM clients define and implement SOA projects in Asia-Pacific. Tony has over 30 years of IT experience, concentrating in financial services and telecommunications. His experience includes application development, e-business systems, architecture frameworks, security, capacity planning and performance, network design, and large-scale transaction systems. He has held a variety of senior technical positions in Australia, Hong Kong and the U.S.

Dr. Harini Srinivasan is on the SOA design requirements team at IBM Software Group – a team that works with customers and IBM architects to capture SOA-specific requirements and articulate them to relevant IBM SOA technology groups including the SOA product divisions. Before joining the SOA design requirements team, she spent 10 years at IBM Research as a research staff member and has worked on customer-focused research topics such as performance analysis tools, JVM runtimes (including instrumentation), and C++ object models. Her background is in program analysis (static and dynamic analysis for performance and debugging), parallel and distributed systems including compiling explicitly parallel programs, and runtimes for Java and C++.

Dr. Chris Harding is the forum director for SOA and semantic interoperability at The Open Group, an open forum of customers and suppliers of IT products and services. He has been with The Open Group for over 10 years. Chris began his career in communications software R&D. He then spent nine years as a consultant, specializing in voice and data communications, before moving to his current role. Recognizing the importance of giving enterprises quality information at the point of use, Chris sees information interoperability as the next major challenge, and frequently speaks or writes on this topic. He has a PhD in mathematical logic and is a member of the British Computer Society (BCS) and the Institute of Electrical and Electronics Engineers (IEEE).



Visit the *New* www.SYS-CON.com Website Today!

The World's Leading *i*-Technology News and Information Source

24/7

FREE NEWSLETTERS

Stay ahead of the i-Technology curve with E-mail updates on what's happening in your industry

SYS-CON.TV

Watch video of breaking news, interviews with industry leaders, and how-to tutorials

BLOG-N-PLAY!

Read web logs from the movers and shakers or create your own blog to be read by millions

WEBCAST

Streaming video on today's i-Technology news, events, and webinars

EDUCATION

The world's leading online i-Technology university

RESEARCH

i-Technology data "and" analysis for business decision-makers

MAGAZINES

View the current issue and past archives of your favorite i-Technology journal

INTERNATIONAL SITES

Get all the news and information happening in other countries worldwide

JUMP TO THE LEADING i-TECHNOLOGY WEBSITES:

<i>IT Solutions Guide</i>	<i>MX Developer's Journal</i>
<i>Information Storage+Security Journal</i>	<i>ColdFusion Developer's Journal</i>
<i>JDJ</i>	<i>XML Journal</i>
<i>Web Services Journal</i>	<i>Wireless Business & Technology</i>
<i>.NET Developer's Journal</i>	<i>Symbian Developer's Journal</i>
<i>LinuxWorld Magazine</i>	<i>WebSphere Journal</i>
<i>Linux Business News</i>	<i>WLDJ</i>
<i>Eclipse Developer's Journal</i>	<i>PowerBuilder Developer's Journal</i>



Long-Running Transactions in SOA

Transaction management in long running service based activities

WRITTEN BY ANSHUK PAL CHAUDHURI, BIJOY MAJUMDAR, AND SUNNY SAXENA

➤ Most organizations that have tried have been successful in implementing a pliable Service Oriented Architecture (SOA) paradigm. Analysts have come out with strategies to translate existing applications into SOA-compliant systems using a staggered approach. The rewards reaped come in the form of low-cost maintenance and agility in their business, along with reusable and self-contained services. But there are still challenges in this form of service-based architecture and solutions need to be devised.

One of the biggest hurdles has been coordinating technology-agnostic services into a single long-running unit of work that produces predictable results. The transactions running across multiple services over multiple domains need to be synchronized to maintain business integrity. Currently organizations depend on proprietary solutions to coordinate transactions for data consistency. This article will walk you through the definition of long-running transaction in SOA and its challenges then talk about the various approaches to resolving the issue while retaining the characteristics of a service-based architecture.

ACID Applications

Applications utilize multiple services across different modules or layers to serve a particular business need. For example, security authentication, service information, EIS information, updating services need to coordinate in a business unit termed a transaction that comprehends data consistency and business integrity in an organization.

Transactions are a set of operations that must be executed entirely or not at all. The fault-tolerance mechanism of managing transactions is to maintain the so-called ACID properties: A – Atomicity (all or none), C – Consistency (the resource must start and end in a consistent state), I – Isolation (make the transactions appear isolated from all the other operations) and D – Durability (once notified, the transaction will persist). ACID provides concurrency in operations and retains data integrity.

ACID properties are easier to implement on transactions that run only a short time because during a transaction the resources are held in a *locked state*. Transactions that run for a long time can't afford to lock up resources. Till date, an ACID transaction assumes closely coupled systems that aren't an SOA-mandated environment. So the ACID properties in a long-running activity need to be applied so that locking doesn't occur, or if it does, then the duration of the locking is as short as possible.

Long-Running Transactions in Service-Crowded Systems

To understand the concept of a long-running transaction, we need to look first into the various lifetimes of a transaction. A *transaction lifetime* can be defined as the minimum amount of time a transaction is kept open. This time period can be anywhere from a few seconds to several hours. A transaction with a short lifetime can begin and end in a matter of seconds, while a long-running transaction can be alive for minutes, hours, even days depending upon the underlying business requirements and implementation. Transactions with a short lifetime are easy to handle since the resources they use can be locked for the time required by maintaining the ACID properties. But the same strategy can't be applied to long-running transactions. Locking up resources for a long time can seriously hamper the application's performance bringing in unnecessary deadlock situations and long wait-times. Any transaction left in an *open state* for an indefinite amount of time qualifies as a long-running transaction.

The following scenarios make long-running transactions possible:

1. A transaction with lots of database queries
2. B2B transactions
3. Batch processes
4. Pseudo-Asynchronous activities within a transaction

In a long running transaction with multiple queries made to the database, the failure of any single query would result in a *transaction failure* that requires rolling back completely to the previously saved *safe state* of the database. The scenario might be across different data sources across different administrative domains.

Batch processes run for long periods of time, usually for hours. Regularly backing up sensitive data is an example. In most cases, batch processes only involve reading data and hence not many transactional issues are encountered. But in certain cases these long batch processes can include modifying the data. A failure during that operation would require an equally long rollback process.

Pseudo-asynchronous activities are used in concurrent activities but the transactions are resumed at some kind of notification. Such operations can be trivial to handle as the control is passed on completely and there is a complex or no way back to reach the sender once the activity is completed. This results in a complex scenario where an independent or intelligently handled rollback needs to be initiated on the source.

In a SOA each functionality is separated as a service. So, a certain application may use many *services* to provide a defined functionality. The principles of SOA define *services* as separate platform- and system-independent entities – each capable of functioning on their own, thus providing reusability and interoperability across disparate platforms.

A long-running transaction creates a number of problems in a SOA architecture. As long as a transaction is limited to a closed environment, catching faults or exceptions and triggering the appropriate rollback mechanism can easily be defined in the underlying application architecture. For example, a transaction involving a database as a resource would already have mechanisms defined in it to handle errors and do rollbacks. Even in a distributed database environment these things can be taken care of. Imagine the same situation in an open SOA scenario where each transactional query is executed on an altogether different platform or system. How a rollback would be implemented in such a case is just one of the immediate questions that comes to mind.

Let's consider a scenario where the transaction involves the participation of three different services – each performing a particular operation. Only if all three operations are successful would the transaction be deemed a *success*. Any other outcome would result in the transaction being marked as a *failure*. Then, if and when the transaction fails, appropriate recovery measures have to be implemented. And to top it off, we can lock a resource only for the time when the service *local* to the resource is operating.

Troubles Within

Let's look into the problems encountered with long-running transactions in SOA. They can be referred to as failure cases:

1. The participation of multiple services results in multiple endpoints being invoked during one cycle of the transaction. Any of these services can be down at the instance when the transaction is in process.
2. SOA boasts of loosely coupled systems. Maintaining transactions is only possible in closely coupled systems.
3. The services involved can be based on any platform. Because of the disparity among the underlying implementation of the services, a context can't be deployed across the services to manage the transactions.
4. The current status in the flow of the transaction can't be known at a given instance.
5. If asynchronous services are involved in the transaction they can't be reached back, unless the service information is explicitly passed on.
6. Resources can't be kept in a *locked state* for long periods of time. To free a resource once the service is done with it, it must release it. Doing this can cause a problem later on if the service fails and a rollback is issued throughout all the services.
7. Alternate methods need to be devised to perform the appropriate recovery operations. In most cases these methods are either platform-specific or too dependent on the underlying business process.

Viable Methodologies

Any methodology that tries to implement transaction management for a long-running transaction scenario in a SOA needs to make sure to:

1. Uniquely identify a transaction across the various participating services
2. Guarantee that the data is delivered and the notifications are sent

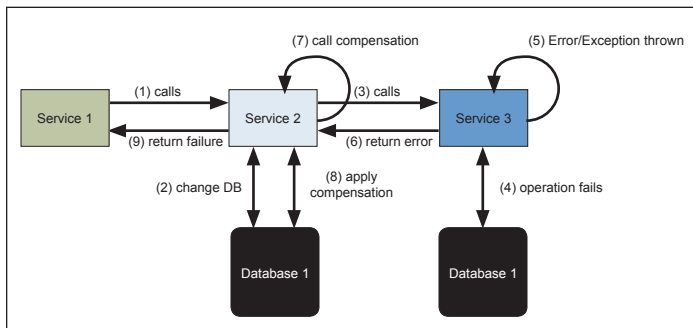


Figure 1: Compensation methodology in a synchronous process

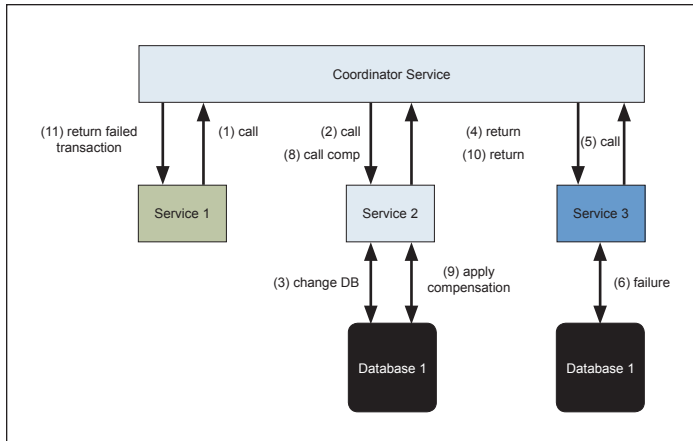


Figure 2: Coordinator service

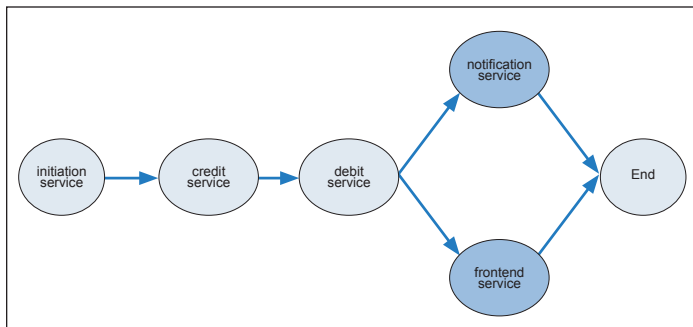


Figure 3: Business case

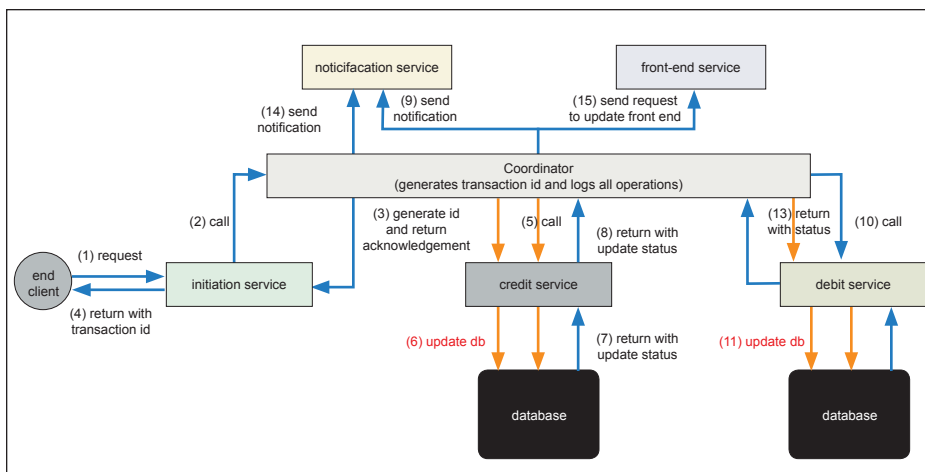


Figure 4: Money transfer scenario (with compensations in red)

3. Some compensation must be provided in case something goes wrong during the transaction
4. Errors in asynchronous services have to be addressed

Keeping these points in mind, the following are some ways to manage long-running transactions in SOA:

1. A compensation methodology
2. Transaction coordinator

Methodology 1: Compensation

In an ideal situation any changes done during a long-running transaction must be reverted back to the original content in case there's a failure somewhere else along the flow of the transaction. This is precisely what happens in a closed environment and is known as a rollback. In a SOA architecture, a situation might arise where a rollback isn't an option. In that case, instead of a rollback, compensation is provided. For example, in WS choreography, the self-reliant services pass control messages back and forth to notify the participating services of a rollback operation.

Compensation may be defined as the most logical change applied to the resource to maintain data consistency and integrity. How it's constructed depends on the governing business rules and underlying technical implementation of the services. In certain cases, compensation can include a rollback. In the example above, if the transaction fails at the third service (the transaction is uniquely identified by an id throughout its lifetime), we need to perform a compensatory operation at the previous service to negate the effect caused by the transaction. So, if the second service sent out an e-mail announcing that it has implemented the changes, a compensatory operation would be to send another e-mail announcing the failure of the transaction and that the changes have been undone. A synchronous process showcasing the scenario is illustrated in Figure 1.

But what if the services participating in the process are asynchronous, as one would expect in a long-running transaction? One way would be to save states and service information.

Methodology 2: Transaction Coordinator

A more appropriate solution would be to orchestrate the process using a transaction manager or process coordinator. Instead of inter-service communication, the services would be answerable to the coordinator, which in turn would handle all the transaction and compensation scenarios. Once again the transaction will be uniquely identified throughout the transaction cycle by an id. This would help the coordinator perform compensatory operations on the required set of data. The coordinator can manage the service information as well. This would solve any issues with asynchronous services. Figure 2 illustrates the coordinator service. This kind of methodology is used mostly in service orchestration-type applications and is a more centralized approach unlike methodology 1.

Case study – Money Transfer Scenario

Consider a *money transfer* scenario where a complete transaction process involves five different services. All five services are separated by virtue of both system and the

language of implementation.

The first service, the *initiation service*, is exposed to the client to pick up the user input. It validates the necessary input parameters and processes the transaction by sequentially executing the credit service and the debit service. Then the system notifies the stakeholder and the internal logs for auditing.

With no transaction context involved in this processing, the services are executed independently with no knowledge of the member service status. There's no way for the executed services to rollback and for specific reasons:

1. Service status isn't shared
2. Non-availability of co-ordination federation in the processing
3. Compensation services for revoking the services

Let's take one of the approaches and bridge the gaps. The solution is to have a coordinator orchestrating though the services and managing the transaction context. The coordinator invariably stores the status of the services and maintains the transaction supporting the ACID properties.

The *coordinator* receives the input and generates an id to uniquely identify the transaction. An acknowledgement is sent to the initiation service as RECEIVED. The *initiation service* notifies the client about the start of the process and provides the unique transaction id. The client can use this transaction id to monitor and track the transaction. The *initiation service* is now ready to take further client input. The *coordinator* maintains a log to record each operation it performs. The log is created against the transaction id.

After generating the id for the transaction, the *coordinator* calls the external service of the bank, which accepts the money. This *credit service* takes the necessary input and starts updating the records in the database. Depending upon the style of the compensation, state information is saved before the update process initiates. Once the update takes place successfully, an acknowledgement to the *coordinator* is sent.

The *coordinator* then logs the changes and proceeds to call the *debit service*. The *debit service* makes the necessary changes to the local database to reflect the debit. The debit process follows the same pattern as the credit process. On successful operation, a DEBITED acknowledgement is sent to the coordinator. The *coordinator* notifies each service involved of successful individual transactions at each step by means enacts the 2PC execution. When there's a failure, the coordinator runs the compensation service for each activity.

Conclusion

The long-running transaction is designed specifically for business interactions that take a long time. The intention is to tie the logical single business-to-business unit of work across heterogeneous domains. Each methodology depends on the architecture of the system and the existing assets in the organization. Technical analysts need to differentiate such special transaction in the SOA study and deal with them through special defined methodologies. ■

References

1. William Cox. "Transactional Web Services."
<http://dev2dev.bea.com/pub/a/2004/01/cox.html>
2. Pat Helland. "Why I hate the phrase Long running Transactions..."
<http://blogs.msdn.com/pathelland/archive/2004/08/12/213552.aspx>
3. Wikipédia
Atomic Transactions:

http://en.wikipedia.org/wiki/Atomic_transaction

Database Transactions:

http://en.wikipedia.org/wiki/Database_transaction

Two-phase commit protocol:

http://en.wikipedia.org/wiki/2-phase_commit

ACID properties:

<http://en.wikipedia.org/wiki/ACID>

About the Authors

Anshuk Pal Chaudhuri is working at the Web Services COE (Center of Excellence) at Infosys Technologies, a global IT consulting firm, and has substantial experience in publishing papers, presenting papers at conferences, and defining standards for SOA and Web Services. The Web Services COE specializes in SOA, Web Services, and other related technologies. His current focus is on Syndeo – the Enterprise Service Bus. His current research interests are WS-Security and various open source products. He is also working on different binding frameworks for XML.
anshuk_palchaudhuri@infosys.com

Bijoy Majumdar is a member of the Web Services COE (Center of Excellence) at Infosys Technologies, a global IT consulting firm, and has substantial experience in publishing papers, presenting papers at conferences, and defining standards for SOA and Web Services. Prior to Infosys, Bijoy Majumdar worked as a technical architect and was instrumental in designing enterprise solutions with leading-edge technologies.
bijoy_majumdar@infosys.com

Sunny Saxena currently works with the SOA and Web Services Centre of Excellence in SETLabs, the technology research division at Infosys Technologies, India. His interests range from Web Service security platforms, transaction management to aspect-oriented development models.
sunny_saxena@infosys.com

SOA WSJ Advertiser Index

Advertising Partner	Web Site URL	Phone #	Page
ACTIVE ENDPOINTS	ACTIVEBPEL.ORG/SOA		4
AJAXWORLD	WWW.AJAXWORLDXPO.COM	201-802-3022	33
AJAXWORLD BOOTCAMP	WWW.AJAXBOOTCAMP.COM	201-802-3022	29
ALTOVA	WWW.ALTOVA.COM	203-929-9400	2
CROSSCHECK NETWORKS	WWW.CROSSCHECKNET.COM	888 276 7725	15
FIORANO	WWW.FIORANO.COM/DOWNLOADSOA		19
FORUM SYSTEMS	WWW.FORUMSYSTEMS.COM	801-313-4400	51
HOSTMYSITE	WWW.HOSTMYSITE.COM		27
JACKBE	WWW.JACKBE.COM	240-744-7620	9
PARASOFT	WWW.PARASOFT.COM/WSJMAGAZINE	888-305-0041 (X-3501)	52
SOFTWARE AG	WWW.SOFTWAREAG.COM/TAMINO		13
SYS-CON EMAIL NEWSLETTERS	WWW.SYS-CON.COM	201-802-3022	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. 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 SOA Web Services 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.

Where's i-Technology Headed in 2007?

—Annual Poll of Industry Prognosticators—

by Jeremy Geelan

At the end of each year, when SYS-CON informally polls its globe-girdling network of software developers, industry executives, commentators, investors, writers, and editors, our question is always the same: where's the industry going next year? Every time, the answers are surprisingly different from the year before, and of course throw light not just on where the industry is going but also how it's going to get there, why, because of who, within what kind of time-scale — all that good stuff. Enjoy!

About the Author

Jeremy Geelan is group publisher of SYS-CON Media and is responsible for the development of new titles and technology portals for the firm. He regularly represents SYS-CON at conferences and trade shows, speaking to technology audiences both in North America and overseas.
jeremy@sys-con.com



Ruby on Rails • JRuby • AJAX • Rules-Based Programming



JASON BELL

Enterprise Developer, Editorial Board Member: Java Developer's Journal

My predictions for 2007....

1. Incremental mainstream adoption of Ruby on Rails

It's going to happen, isn't it? Keep an eye out for Sun's offering of JRuby. Whether this is the death of other open source scripting languages like Groovy remains to be seen. Ruby has been a wake-up call and has now drawn the line dividing serious scripting languages from "hobby" languages (ones that wouldn't see enterprise adoption). For me, my job just got a whole lot easier, a whole lot quicker.

2. A slowdown in the AJAX hype

I think the shine has worn off. There are some nice applications about but at the end of the day it's a Web page with some very fancy JavaScript.

3. 2007 is the year of the business rule

Rules-based programming will be big business. With the likes of JBoss acquiring Drools it's certainly an area to keep an eye on.

LAMP • REST • ATOM • Apple



DAVID HEINEMEIER HANSSON

Creator of (Ruby on) Rails

My predictions for 2007....

1. 2007 will be the year where LAMPers finally decide

To stop being neutral about the WS-* mess and pick the side of REST: the next wave of Web APIs will stop supplying both a SOAP and REST API and just go with the latter.

2. On the leading edge, we'll see the same for RSS vs ATOM

For techies in the know, ATOM will become the assumed default syndication format and that'll mark the slow decline of RSS (though more as a technology than as a brand, RSS will remain synonymous with feeds).

3. Apple will continue to trounce everyone else for the preferred geek platform.

The stigma of being a Web programmer still using Windows will increase.

Vista • Office 2007 • Zune • AJAX • Ruby • Java Ruby on Rails • Flash Memory



GARY CORNELL

*Founder & Publisher,
Apress*

My predictions for 2007....

1. IE 7 will have a fast adoption curve and so Firefox will cease gaining market share.
2. Vista will have a slow adoption curve.
3. Office 2007 will have a slower adoption curve.
4. Oh, the Zune will have no adoption curve.
5. The AJAX bandwagon will gain even more speed.

6. Ruby's momentum will slow down as Python and PHP frameworks to combat Rails grow in popularity.
7. The open-sourcing of Java will have no effect whatsoever on Java's slow decline in favor of dynamic languages (Ruby, Python) and C#.
8. Sales of high powered desktop will slow.
9. Apple will no longer gain market share for its desktops and will stabilize at its current meaningless level.
10. Ultra lightweight notebooks based on flashmemory with instant on/off will start coming out in large numbers.

SOA & Web 2.0 "Outside-In SOA" Semantic Web • AJAX



DAVID S. LINTHICUM

*CEO,
The Linthicum Group*

My predictions for 2007....

1. The worlds of SOA and the Web 2.0 blur

together. While many who think SOA don't think Web 2.0, and many who think Web 2.0 don't think SOA, those days will come to a fast end in 2007. So, what does this mean to those standing up SOAs today? It's clear that many of the services we consume and manage going forward will be services that exist outside of the enterprise, such as subscription services from guys like Salesforce.com, or perhaps emerging Web services marketplaces from guys like Strikellon, Google, Amazon, and others. This is outside-in SOA, in essence reusing a service in an enterprise not created by that enterprise, much as we do today with information on the Web. Thus, those services outside of the enterprise existing on the Internet create a Universal SOA, ready to connect to your enterprise SOA, perhaps providing more value.

- #### 2. The rise of the Semantic Web.
- The Semantic Web is the abstract representation of data on the World Wide Web, based on the Resource Description Framework (RDF) standards and other standards. Although this notion has been around for some time, in 2007 it will greatly affect how we design, build, and deploy Web 2.0 applications and SOAs, providing a mechanism to track and leverage application semantics, local and remote.

- #### 3. Enterprise applications continue to move outside the enterprise.
- With the success of Salesforce.com and many others, we'll continue to see applications move to the Web including accounting, CRM, HR management, logistics, inventory management, etc. While many Global 2000 companies will fight this trend, the success of the younger and more nimble up-starts will drive this movement quickly.

- #### 4. The success of AJAX drives traditional software back to the drawing boards.
- With the ability to finally provide dynamic rich content and applications over the Web, traditional software vendors will find that they need new products to play in this new world. Indeed, as Google Mail is giving Microsoft fits, so will other more innovative Web-delivered applications leveraging rich client technology such as AJAX. Entire interfaces will have to be rewritten to support AJAX, and end users will demand that we move away from traditional pump-and-pull HTTP programming.

Mobile AJAX • “Mobile Web 2.0” • SMS • LBS • Flash Lite • On-Device Portals



LUCA PASSANI

Wireless Guru & Technology Evangelist, Openwave

My predictions for 2007....

1. AJAX will still be hyped, but we will still see no mobile AJAX-based killer apps, only proofs of concept.
2. JAVA ME will not gain much more ground. Too fragmented. Games and some other apps. No killer apps though.
3. What people call “Mobile Web 2.0” is not Web browsing. Saying that mobile and Web will converge is trendy in some environments these days. This is wrong and that’s hardly surprising: people buy phones to make calls, not to browse the Web, so why should we expect phones to get so much better at browsing the Web?
4. SMS will still represent 80% or more of data traffic. The rest will be downloads: ringtones, wallpapers and games. WAP will be mostly used as a discovery mechanism to get to those contents. Reformatting proxies to adapt Web content for mobile will be implemented by most operators. They will increase browsing a bit, but nothing earth-shattering.
5. Not sure about Location-Based Services. LBS have been on the verge of explosion for some time now.
6. Flash Lite will make significant progress in Europe and North America, also on operator portals.
7. On-Device Portals are an interesting development: content gets pushed to devices while the user isn’t watching and they may decide later to buy it or not. This will be trendy next here. It will be interesting to see which actual implementations of the concept deliver.
8. More people will realize that device fragmentation is one of the main hurdles for mobile.

Flash Memory • AJAX Productivity • Red Hat • Vista • Notebooks • Ubuntu



MARK HINKLE

Editor-in-Chief, Enterprise Open Source Magazine

My predictions for 2007....

1. **Flash-bootable PCs** – It’s been a long-time coming but laptop PCs will start booting from flash memory. This will make a huge difference in battery life. Intel will lead the way pushing their NAND flash boot memory on a new laptop platform and Apple will be among the first to adopt. The One Laptop per Child initiative will also provide a demonstration of the first zero disk drive PCs albeit small. Devices like this will inspire creativity on higher end models especially as the price of non-volatile memory continues to drop.
2. **New Crop of AJAX Productivity Applications** – While the buzz around AJAX may fade, the number of robust new AJAX-enabled applications will increase. These applications will be built on evolving AJAX frameworks like Dojo and Rico and commercially backed platforms like OpenLazlo. Of course every new start-up will be secretly hoping for Google to make a bid and join the family that has been expanded this year by Writely and Jotspot.
3. **Red Hat Will Become an Acquisition Target** – Someone will make a bid on the #1 Linux vendor. Maybe Oracle who has done a number on the leading Linux vendor with Unbreakable Linux will take advantage of Red Hat’s near 52-week low. Uncertainty and ambiguity in the enterprise Linux market will send Red Hat looking for another partner to avoid being swallowed by the DB maker. Maybe IBM will become Red Hat’s white knight.
4. **Open Source Everywhere** – More and more companies will open source legacy products and launch new ones under open source licenses. Database vendor Ingres is going to set the standard that other more conservative infrastructure vendors will follow. Look for new open source initiatives from major infrastructure vendors like BMC, VMware, and even Microsoft.
5. **Microsoft Vista Launch Will Boost Sales of Other OSes** – Microsoft’s launch of Vista will start to prompt hardware refreshes which can be nothing but good for Apple. Apple already has momentum, Intel hardware, dropping prices and all the tumblers are becoming aligned for it to creep above its measly 5% market share. Linux desktop vendors will likely see a few defectors from the Redmond camp, though big ships turn slowly. Look for Ubuntu to be the Linux desktop distribution of choice.
6. **Half of All New PCs Will Be Notebooks** – PC buyers are buying more notebooks every quarter and sometime in 2007 the number of shipping notebooks will match the number of desktop PCs or come very close.

Open Source Java • General Public License v2 • GPL v3



TONY WASSERMAN

Professor of Software Engineering Practice at Carnegie Mellon West and Executive Director of the Center for Open Source Investigation (COSI)

My predictions for 2007....

Consequences of Open-Sourcing Java...

The open-sourcing of Java under the GPL 2 license will have a ripple effect on various technical and business issues in 2007:

- First, people will closely study the Java source code and find one or more serious bugs, at least one of which has been there since the earliest days of Java.
- Second, a real-time systems vendor will fork the source code, as permitted by the GPL, and create a variant that is “tuned” for real-time applications. This step will be the focus of a major debate within the Java community.
- Third, the open-sourcing of Java will have a positive impact on the adoption and use of open source software in general.
- Fourth, the use of the GPL 2 for open-sourcing Java will inhibit the completion and acceptance of the GPL 3 proposal.

IT Enabled Services • Web TV • Visual AJAX IDE • Microsoft Atlas • Apache XAP



COACH WEI

Founder, Chairman, & CTO, NexaWeb

My predictions for 2007....

1) IT Enabled Services is going to fly high in 2007. As a result, we will see:

- A lot more venture capital investments into IT Enabled Services;
- Of course, a lot of startup activities in IT Enabled Services (new company creation, merger and acquisition);
- There will be some significant moves made by "traditional, big companies" into IT Enabled Services too. For example, some of the possibilities are:
 - Massive reality shows on the Web, instead of being on TV. Can you imagine "American Idol" on the Web? Speaking of this, I think highly of Yahoo's initiative into this area, including its recent acquisition of Bix.
 - A major entertainment company (NBC, ABC, etc.) fully embracing

Web TV.

2) AJAX grows up – which means the following are available and useable:

- Visual AJAX IDE (solving the ease-of-development issue. Most likely based on Eclipse ATF);
- Declarative AJAX Framework (solving the ease-of-development issue. Most likely based on Microsoft Atlas and Apache XAP);
- Adoption of AJAX within less leading-edge enterprises.
- AJAXWorld Conference overtakes JavaOne conference. JavaOne is being renamed as JavaScriptOne Conference.

3) Growing adoption of Web 2.0 technologies within the enterprise

- Enterprise Mashup Server emerges as a product category.
- Less leading-edge companies start to adopt Web 2.0 technologies.

4) The IPO market shows signs of opening up

- One or two Web 2.0 companies go public, the majority of the exits are acquisitions.

WS-BPEL 2.0 • BPM & Web 2.0 • SOA • XSLT • JSON



JOHN EVDEMON

Architect, Microsoft,
with the Architecture Strategy Team
focusing on BPM and SOA

My predictions for 2007....

E.F. Schumacher, a well-known British economist, once wrote: "I cannot predict the wind but I can have my sail ready." With that thought in mind here are ten predictions and hopes to help get your sails ready for 2007:

- The WS-BPEL 2.0 specification will finally be approved as an OASIS standard.** Adoption of WS-BPEL will initially be slow, driven by customer demand. BPEL will evolve beyond a "check box requirement" if people begin using it as a foundation for defining process profiles (conceptually similar to how people use WSSecurity today). An updated mapping from BPMN to WS-BPEL will also be published.
- The convergence of BPM and Web 2.0 begins.** BPM is about improving performance by optimizing key processes. Web 2.0 is about capturing the wisdom of crowds (or as O'Reilly puts it, the architecture of participation). The convergence of BPM and Web 2.0 enables collaborative development and tagging of sub-processes, establishing a "process folksonomy" where the best processes can evolve organically. Collaboration can occur over simple but highly scalable pub/sub mechanisms (like Atom or SSE). Lightweight tools will enable users to model or reuse sub-processes using a broad set of metadata. While this is an exciting opportunity, there are several technical and non-technical issues that must be addressed before this convergence becomes a reality.
- Improvements in SOA management and governance.** Tools, frameworks and platforms will emerge that better enable:
 - Defining and enforcing service development guidelines
 - Modeling, managing and enforcing operational policies (e.g., security, service level agreements and others)
 - Service simulations (what-if scenarios, impact analysis, etc.)
 - Modeling and managing service dependencies

- Service provisioning and de-provisioning
- Configuration management

- Workflow isn't confined to the datacenter anymore.** Lightweight, extensible frameworks like Windows Workflow Foundation (WF) enable workflow in places where it may not have been previously considered.
- Better UI Experiences.** Declarative user interfaces will enable rich user experiences that can be easily modified or extended with simple mechanisms like XSLT. Familiar business applications like Office provide the user interface to back-end line-of-business systems. The line between AJAX-based UIs and rich desktop UIs will blur, enabling users to enjoy both connected and occasionally-connected experiences. Tools and guidance will make building, testing and deploying these composite UI experiences much easier.
- A new category of architecture emerges:** Software + Services. It is hard these days to find an architectural concept that is not somehow tied to services. The line between Web services, SaaS and traditional applications will blur to the point where the location, contract and hosting of a service are less important than the capabilities exposed by the service.
- JSON without AJAX.** We'll start to see more people using JSON to address the XML bloat problem outside of simple AJAX-based applications. The downside is that this may result in more tightly-coupled applications.
- Events and states instead of EDI-style messaging.** Lightweight frameworks will empower developers to start thinking about solutions in terms of event notifications instead of simple messages passing from point A to point B. Hierarchical state machines enable state synchronization across complex, federated processes.
- We stop talking about SOA and "just do it."** Sometimes we spend more time arguing about IT trends than actually using them. In 2007 the tools and specifications we need for enterprise-strength, loosely-coupled solutions have finally arrived – it's time to roll up our sleeves and get to work.
- IT finally admits that there is no silver bullet.** Every year I hope to see this happen and every year my hopes are crushed by buzzword-of-the-minute hype machines. (Hey I can dream, can't I?)

SaaS • Open Source • Mobile • Enterprise IM • Social Networks • China • Virtualization • RFID • Internet Video • Prediction Networks • Intelligent Wikis



BOB ZUREK

Director of Advanced Technologies with IBM Information Integration Solutions

My predictions for 2007....

The Z Generation CIO & IT Professional

The Z Generation is typically described as those who are born sometime during the early 2000s and continue to the 2017 time frame. So what will these Generation Z IT Professionals be experienced with. Here's my prediction for the Top Ten characteristics (and this is just the tip of the iceberg):

- 1. Grew up in the world of SaaS and open source** and wonders why you would ever license and install software. If you still needed to install software, then it should be available in the form of open source. Expects all internal projects to be developed using the open source model.
- 2. Grew up with a mobile technology** and wonders how anyone could run a business without it. Insists everything be available on a highly portable digital device and everyone in the organization have a device. No exceptions.
- 3. Grew up knowing how to leverage the power of social networks** for the benefit of the corporation. This includes the ability to build out these networks and use them to help build new products and technologies. Generation Z CIOs will have a huge advantage as they have grown up as participants in many social networks. China will be a big source of these networks. Websites will be built by the Z Generation CIO to invite outsiders in to help build new and innovative products that have yet to be thought of by the enterprises internal employees.
- 4. Grew up using Instant Messaging** and will insist that the enterprise use IM as a priority over email and that email will only be used if the communications can't be done using the features of the future enterprise IM platform.
- 5. Will tap into offerings such as TopCoder.com** to supplement project teams. There will be a world of competing Topcoder.com like sites where the best coders in the world will be found to solve very complex algorithms and other challenging software projects facing the IT department. China will be a major provider of these teams.
- 6. Grew up with a complete understanding of the value of virtualization** and therefore, their datacenter will be virtualized and the IT operating fabric will be grid-based, tapping the power of external grids of CPU.
- 7. RFID Everywhere.** The Z Generation will be the ones that take RFID to new heights. Everything that is taggable will be tagged and tracked.
- 8. CIO and Z Generation IT Professionals** will leverage the power of Internet video by taking advantage of companies like BrightCove Networks which will bring knowledge workers engaging channels like "The Customer Service Channel", "The Corporate Strategy Channel", "The M&A Channel" and others. I can see companies like Harvard Business Review and others producing content for these channels. Imagine the "The DataCenter Channel." The topics are endless and will be as easy to find as bringing up your favorite search engine. New content will be generated targeted for companies like "The IBM Channel" or the "GE Channel." I would love to see "The Institute of The Future Channel"
- 9. Intelligent Wikis** will be the primary source of knowledge in an enterprise and will eventually do what data warehousing did for business intelligence. Furthermore, new internal employee-generated communities will spin up to voluntarily invent new projects during their off-hours to showcase their creativity that is typically not known by the employer.
- 10. CIOs will aggressively adopt Prediction Networks** as part of the core business strategy to better help the enterprise gauge where everything from sales to new product development will be successful.

Web Service Orchestration • Web Services Explosion



ADAM KOLAWA

Co-Founder & CEO, Parasoft

My predictions for 2007....

1. I anticipate a significant demand for Web service orchestration in the upcoming year, especially in the United States. Many organizations now have at least one Web service, and a growing number already have two or more related Web services. Managing multiple related Web services is considerably more challenging than managing the same number of separate, unrelated Web services. To use these related Web services to achieve your business goals, you need to consider how high-level operations pass through the Web services, then determine how to implement this high-level flow— from start to finish. This can be accomplished in two ways:

- By programmatically coding the application logic required to tie the involved elements together.
 - By using an orchestration tool to direct the flow through the involved elements, which remain separate.
- I predict that the latter method will be the favorite because it is easier.

2. I also expect an explosion of Web services because they are so easy to expose. Once exposed, Web services basically create interfaces which can be reused. This will significantly reduce the amount of code that needs to be written, which will in turn cut the demand for "bare bones" development.

Server Virtualization • Container-Based Hosting • Linux Rails • Django • Agile Development



BRANDON HARPER

Senior Software Developer at Acxiom Corp.

My predictions for 2007....

1. **Server virtualization** is just getting started, and will really make itself known in the coming year. Once we start seeing the quad core CPU architectures as a part of standard infrastructure, it really starts making a lot of sense to start deploying and managing servers and applications as virtual entities rather than specific pieces of hardware. This helps manage the cost and pain of software configuration management, take advantage of being able to process many tasks simultaneously because of hardware support, as well as allows legacy hardware to be retired in favor of applications running on virtual servers.
2. **Container-based hosting** is the new kid on the block, and will also start making its presence known in the upcoming year. Commonly labeled as "grid" hosting (which is a technical misnomer if you understand distributed computing), it essentially claims to be an infinitely scalable hosting platform. This technology still seems to be halfbaked at the moment, but you could have said the same thing about Linux ten years ago.
3. **People who normally wouldn't use Linux** start to explore it and even replace Windows with it permanently. With Vista, Microsoft seems to be moving to a model in which the Windows operating system is a method to police users with DRM and other nonsense rather than provide developers with a good platform on which to use hardware, which is what operating systems are really supposed to be. A lot more consumers who haven't noticed this happening in the past will stand up and notice this year.
4. **Dynamic languages and frameworks** will continue to make leaps in popularity and adoption. Given the current squeeze on technology talent in the US, companies are going to have to learn how to do more with fewer resources. Moving to dynamic languages and frameworks as well as other simplification such as varying Agile software development practices will enable this to take place. I think the obvious leading candidates here are Ruby on Rails and Django.
5. **The enterprise will embrace** ways to simplify development by continuing to embrace open source software and Agile Development strategies. While there are a lot of cries to the effect of Ruby on Rails replacing Java, I think that's complete nonsense as Java is a language and Ruby on Rails is a framework. Rapid development languages will certainly make some inroads, particularly where heavy tools have been used to build simple applications, Java is still going to be a major part of the service-oriented enterprise for years to come because of the power and tools it provides as well as its industry support.

Open Sun • iPod Uno • IT2 Microsoft VAPOR



RICHARD MONSON-HAEFEL

Award-Winning Author & Senior Analyst, Burton Group

...And in Other 2007 News

1. **Jonathan Schwartz open sources Sun Microsystems**
In a move that will surprise everyone Sun Microsystems will announce that it will open source its entire company. Sales, marketing, finance, and even operations will be open to the community for anyone to contribute.
2. **Apple computer announces the iPod Uno**
The size of a match stick with no screen or controls, the iPod Uno plays one song in a constant loop. Despite its limited capabilities, the tiny device becomes an instant hit and a cultural icon.
3. **In what is heralded as the seminal article on the subject, Tim Berners-Lee mentions "IT2"**
Overnight the term morphs into "IT 2.0," spawning thousands of blog entries and press articles, a dozen books, five conferences, and millions of dollars in venture capital. It turns out that the original article, incomprehensible to most readers, was actually another attempt to explain the Semantic Web and the IT2 reference was just a typo.
4. **Microsoft will create the first CMO (Chief Marketing Officer) position**
The new CMO will immediately change his own title to Chief Command & Control of Packaging Officer (C3PO) and then announce that Vista will be delayed and renamed Microsoft Virtualization Application Program Operating system Reloaded (Microsoft VAPOR).

AJAX Over-use • JSF • Relational Object Mapping • Macs



BILL DUDNEY

Editor-in-Chief, Eclipse Developer's Journal

My predictions for 2007....

1. **AJAX will continue** to gain momentum as folks continue to have the epiphany that Web 1.0 UI is not good for users. Overuse of the technology will be a real problem. JSF will finally start to become a de facto as well as actual standard due to its ease of integration with AJAX.
2. **Open Source enablement** will continue to be a hot spot for VC investment. I don't think the perfect business model will emerge in '07 though so the market will still remain 'immature.'
3. **Java Persistence API** will bring relational object mapping to the long tail of the market. Early adopters will be wondering what all the hype is because the technology is so old in their eyes.
4. **Macs** will continue their 'thought leader' adoption curve. This is not the year they start to penetrate the corporate IT department.



Merlin Uses SOA Expertise to Strengthen USTRANSCOM Web Services

(Washington) – Merlin International, a federal IT solutions provider, has announced that its engagement with USTRANSCOM has been expanded to include upgrades to the agency's Web services infrastructure. These enhancements will establish a metadata repository and make all relevant metadata available to the Defense Information Security Agency (DISA), allowing USTRANSCOM to meet data-sharing requirements established by the Office of the Secretary of Defense (OSD). www.merlin-intl.com



IdeaBlade Extends Web Services Support with DevForce 3.3

(Emeryville, CA) – IdeaBlade, Inc, a provider of enterprise application development tools for the Microsoft .NET Framework, has announced a new release of the DevForce Productivity Suite, IdeaBlade's flagship solution. DevForce 3.3 extends support for Web services and service-oriented architectures (SOA), simplifies user interface development, and introduces advanced application customization capabilities. DevForce's enterprise application framework, tools and application servers enable ISVs and enterprises to bring Internet-enabled applications to market more quickly and at a lower cost. www.ideablade.com



TIBCO Arms Organizations with New Tools and Expanded Resources to Create Roadmap for SOA Execution

(Palo Alto, CA) – TIBCO Software Inc. has announced an interactive service-oriented architecture (SOA) planning guide to help organizations identify gaps and accelerate the planning of a scalable service-based environment. The planning guide is available for free on TIBCO's newly enhanced SOA Resource Center (<http://www.tibco.com/solutions/soa/default.jsp>). In addition, the SOA Resource Center now features an easy-to-navigate user interface and a role-based approach to navigation and content for every phase of the SOA planning life-cycle. www.tibco.com



Active Endpoints Announces Support for WS-BPEL 2.0 with New ActiveBPEL 3.0

(Shelton, CT) – Active Endpoints, Inc., a provider of SOA orchestration products and services, has announced the availability of ActiveBPEL 3.0. The ActiveBPEL product family includes open source and commercial SOA orchestration solutions that are standards-compliant and platform-neutral, forming the foundation for fast, cost-effective business and systems integration. Among other important capabilities, ActiveBPEL 3.0 comprehensively supports the forthcoming WS-BPEL 2.0 standard, which will be officially published early in 2007. www.active-endpoints.com



The Open Group Announces SOA Governance Project

(San Francisco) – The Open Group, a vendor- and technology-neutral consortium focused on open standards and global interoperability within and between enterprises, has announced the creation of The SOA Governance Project, an effort that will investigate how enterprises can better control and facilitate the development of a service-oriented architecture (SOA). Driven by members of The Open Group's recently formed SOA Working Group, the new initiative will create a reference framework for SOA governance that can be used by organizations as they manage both new and ongoing SOA initiatives. www.opengroup.org



IBM Extends Product Lifecycle Management with SOA to Improve Executive-Level Decision Making

(Armonk, NY) – IBM has announced plans to create a Product Development Integration Framework to better link the design and development of products with core business processes, elevating product lifecycle management and development from an engineering-centric function to a strategic business process that improves executive decision-making abilities across the enterprise. www.ibm.com



Xcalia Secures Venture Financing to Expand Business and Market SOA Software Platform

(Palo Alto CA / Paris) –Xcalia, a provider of dynamic integration software, has announced that it has raised an additional two million euros in venture financing to continue its growth. The funds come from European software investors Innovacom, Iris Capital and I-Source Gestion, all of which have supported the company in previous stages of growth. www.xcalia.com

SOA in the Small

WRITTEN BY AJIT SAGAR

➤ At my firm, Infosys Technologies, I have come across several clients who are actively trying to explore, consider, adopt, embrace, or become completely immersed in SOA. Here is a typical call I've received, where our client rep says, "Ajit, we've got a very critical meeting with the CIO of company ABC. He is very excited about moving his entire organization toward SOA. Can you come and present our SOA capabilities to the client on Monday?"

Can I help here? Sure I can. As a principal architect in our technical consulting group, that is a part of what I do – what I'm paid for. However, to engage in a meaningful conversation and to really add value to that meeting, it's very important to answer the questions:

- How large is this initiative?
- What is the client's understanding of SOA?
- Do they know what type of a platform they are considering?
- What is the size of the application portfolio they are looking to "service-orient?"
- What LOBs (Lines of Business) are involved in this initiative?

After having gone through some of these initial meetings, there is a basic question that comes to mind – can you actually do "SOA in the small"? To me, the answer is "No." I don't believe that an organization can successfully implement an SOA without having a substantial application/business footprint behind it. Moving to an SOA is not a trivial task and requires substantial changes in organizations – cultural, organization, technical, etc. Granted the benefits can be very lucrative, but it does require a non-trivial investment.

For people who have been engaged in distributed development, enterprise architecture, component development, etc., SOA is not a new concept. In fact these participants in the enterprise will argue that SOA is the "Same Old Architecture." There is definitely a large degree of truth in that statement. SOA is not Web services. SOA is not necessarily Java, .NET, and does not necessitate the usage of the latest tech-

nologies. However, these are very feasible options for undertaking a new initiative to service-enable applications that were not service-enabled before.

The key concept behind SOA is the ability to bridge the business-IT gap and promote business agility. And the key to success around an SOA initiative is not just the architecture, but also the process – SOA governance, IT strategy, service definition, translation of business services to technical services, and so on. Accomplishing many of these requires substantial overhead, which can only be justified if there is a larger initiative that can realize the benefits of service orientation.

Does this mean smaller initiatives, LOBs, or organizations can't participate in SOA? That's not what I am saying. However, the most important factor to consider as you

participate in such initiatives is to understand what the bigger picture is. Which SOA initiative are you a part of? What part of the business value chain do you participate in? If you don't know the answer, there is no point in service enablement, because you have not identified your service consumer.

On the other hand, if you have understood the bigger picture, then you are a participant in the governance process (because that part is handled in the larger service-oriented enterprise), your services are well understood (because you are working with the appropriate IT strategy team), and you will effectively plug into the right service-oriented architecture (because there is a team who is responsible for defining it). On the other hand, you may be a part of the governance, IT strategy, or architecture team in the larger business initiative. In these cases, "SOA in the small" can be implemented because the aspects of "SOA in the large" are already defined. ■



About the Author

Ajit Sagar is a principal architect with Infosys Technologies, Ltd., a global consulting and IT services company. Ajit has more than 15 years experience ranging from programmer, lead architect, director of engineering, and product manager for companies from 15 to 25,000 people in size. At SYS-CON, Ajit has served as JDJ's J2EE editor, was the founding editor of XML Journal. Ajit is a frequent speaker at SYS-CON's Edge series of conferences, JavaOne, and international conferences. He has published 125+ articles.
ajitsagar@sys-con.com

How Much Will Your SOA Cost?

Here's a sensible, no-nonsense approach to costing out SOA

WRITTEN BY DAVID S. LINTHICUM

➤ I'm consulting now...at the project and strategy levels...and finding that a lot of real work needs to be done to get SOAs up and running. For most organizations, the first step of their SOA project is to figure out how much this SOA will cost. So you can budget appropriately and get the funding.



Typically it's expressed as:

$$\text{Cost of SOA} = (\text{Cost of Data Complexity} + \text{Cost of Service Complexity} + \text{Cost of Process Complexity} + \text{Enabling Technology Solution})$$

For instance:

$$\text{Cost of Data Complexity} = ((\text{Number of Data Elements}) * \text{Complexity of the Data Storage Technology}) * \text{Labor Units})$$

- The number of data elements being the number of semantics you're tracking in your domain, new or derived.
- The complexity of the data storage technology, expressed as a percentage between 0 and 1 (0% to 100%). For instance, relational is a 0.3, object-oriented is a 0.6, and ISAM is a 0.8.

So at \$100 a labor unit, or the amount of money it takes to understand and refine one data element, we could have:

$$\text{Cost of Data Complexity} = (((3,000) * .8) * \$100) \text{ or, the cost of data complexity} = \$150,000$$

Or, the amount of money needed to both understand and refine the data so it fits into your SOA, which is a small part of the overall project by the way.

If you get this, you can get the rest of the cost analysis procedure; just reapply the same notions to: *Cost of Service Complexity, Cost of Process Complexity, Enabling Technology Solution*

Some things to remember:

1. This is not really metrics (e.g., function points) because we really don't have historical data to abstract here. In essence, you need to use your own project management and project costing methods; just apply them to this new approach, using the formulas I'm suggesting above.
2. Count on 10%-20% variations in the cost for the simple reason that we haven't walked down this road before. As we move from project to project, we'll get better at costing out a SOA.
3. Make sure you factor in at least two major mistakes like selecting the wrong vendor or hiring the wrong architect. You may encounter more, but it will almost never be less.
4. Make sure to change cost estimates as scope creep occurs, and it always does. The nice thing about using formulas such as the ones I'm expressing here is that, as change occurs, you can quickly see the effect on the budget. Moreover, as change occurs later in the SOA projects, the cost of change goes up exponentially.

Finally, here's a sensible, no-nonsense approach to costing out SOA. While the actual numeric assumptions may be debatable, it's the approach that's refreshing. It would be great to see people start using this model, sharing any data points and providing feedback so it could be refined. Clearly, this would benefit the IT community a lot. I plan to develop a much more sophisticated model in the near future. If you're interested, let me know. I'll post it on my Web site. ■

It's a good first step, but most organizations that want to build an SOA don't have a clue about how to approach the cost estimate. In many cases, they grossly underestimate the cost of their SOA, hoping their bosses and accountants won't notice later. In other words, go in low to get the approval, and reveal the higher costs later after it's too late...the investment has been made. Not a good management practice, if you ask me, but a pretty common one.

So, how do you calculate the cost of an SOA? While you can't cost out an SOA like a construction project, many of the same notions apply, including: Understand the domain, understand how much required resources cost, and understand how the work will get done. Moreover, understand what can go wrong and account for it.

Here are some very "general" guidelines:

Budget to budget. The problem that I'm seeing over and over again is that cost estimates are provided without a clear understanding of the work that has to be done. Indeed, you need to budget some time to create the budget. This means understanding the domain in detail, including the:

1. Number of data elements
2. Complexity of data storage technology
3. System complexity
4. Service complexity
5. Process complexity
6. New services needed
7. Enabling technology
8. Applicable standards
9. Potential risks

About the Author

David S. Linthicum (Dave) is an internationally known application integration and service oriented architecture (SOA) expert.

david@linthicumgroup.com

SOA
MAKE YOUR ^ SECURITY MOVES WISELY...



XWALL

WEB SERVICES
FIREWALL



XRAY

WEB SERVICES
DIAGNOSTICS



VULCON

VULNERABILITY
CONTAINMENT SERVICE



SENTRY

SOA SECURITY
GATEWAY

PUTTING TOGETHER THE PIECES FOR THE WORLD'S MOST DEMANDING SOA SECURITY SYSTEMS

FORUM SYSTEMS ENTERPRISE SOA SECURITY SOLUTIONS:

- ▶ TRUSTED SOA MIDDLEWARE
- ▶ WEB SERVICES SECURITY
- ▶ XML ACCELERATION

W W W . F O R U M S Y S T E M S . C O M



FORUMSYSTEMS

THE LEADER IN WEB SERVICES & SOA SECURITY

Complex and evolving systems are hard to test...



Parasoft helps you code smarter and test faster.

Start improving quality and accelerating delivery with these products:

Awarded
"Best SOA Testing
Tool" by Sys-Con
Media Readers

SOAtest™

InfoWorld's 2006
Technology of
the Year pick for
automated Java
unit testing.

Jtest™

Automated unit
testing and
code analysis
for C/C++ quality.

C++test™

Memory errors?
Corruptions?
Leaks?
Buffer overflows?
Turn to...

Insure++™

Easier Microsoft
.NET testing by
auto-generating
test cases,
harnesses & stubs

.TEST™

Automate
Web testing
and analysis.

WebKing™

PARASOFT®

We make software work.™

Go to www.parasoft.com/WSJmagazine • Or call (888) 305-0041, x3501