

OUR NAME HAS CHANGED!
FORMERLY SOA SERVICES JOURNAL

FOR ALL THE LATEST NEWS
AND INFORMATION
VISIT:

www.SOA.SYS-CON.com

SOA WORLDTM

M A G A Z I N E

FEBRUARY 2007 / VOLUME: 7 ISSUE 2

Racing without a ***Speedometer***

**IS YOUR ACCELERATOR
REALLY SOLVING YOUR
NETWORK PROBLEMS?**
PAGE 10

3 Getting on the Grid
SEAN RHODY

4 SOA Project Staffing Plan
DAVID S. LINTHICUM

6 10 Principles of SOA
STEFAN TILKOV

14 BPEL's Growing Up
DAVID SHAFFER AND MANOJ DAS

20 Testing the "REST"
V. NIRANJAN

26 The Software-as-a-Service Model
ROGER BOTTUM

28 Equal Web Site Access
DAVID CUMMINGS

**32 Re-engineering a Legacy
Enterprise IT System**
CHUNG-YEUNG PANG



PLEASE DISPLAY UNTIL APRIL 3, 2007

\$6.99US \$7.99CAN





Gear up for development excellence

Take off with the Altova MissionKit, and discover
the secret to savings on top software tools.

Spied in the Altova MissionKit 2007:

- The world's leading XML development tools
- Plus application design, data management, and modeling options for Software Architects

The Altova MissionKit 2007 bundles Altova's intelligent application development, data management, and modeling tools at 50% off their regular prices. Available in a variety of configurations tailored to the needs of Software Architects and XML Developers, the Altova MissionKit delivers the highest functionality and best product value. It's your first-class ticket to the power, speed, and simplicity of Altova's award-winning product line. Save a bundle!

Download the Altova MissionKit 2007 today: www.altova.com

INTERNATIONAL ADVISORY BOARD

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

TECHNICAL ADVISORY BOARD

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

EDITORIAL

Editor-in-Chief

Sean Rhody sean@sys-con.com

XML Editor

Hitesh Seth

Industry Editor

Norbert Mikula norbert@sys-con.com

Product Review Editor

Brian Barbash bbarbash@sys-con.com

.NET Editor

Dave Rader davidr@fusiontech.com

Security Editor

Michael Mosher wjsjsecurity@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 © 2007 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.

Getting on the Grid

WRITTEN BY SEAN RHODY



One of the most interesting aspects of being a consultant is that I get exposed to any number of different facets of system design in the course of an assignment. While I tend to focus more on application and integration work, I find it fascinating to deal with the concepts of services in the context of infrastructure.

In the past, I've been called upon to design Service-Oriented Infrastructures (SOI) – the hardware and platform software, along with customizations for the needs of the actual deployment environment – instead of creating an application architecture. SOI is really a different way of looking at the concept of services – from the viewpoint of the operational staff.

Most of the time, we tend to view services from a line of business or at the very least an application development perspective. There's nothing wrong with viewing SOA this way – it's very important to break down the artificial walls created by applications into discrete services, and it provides great flexibility to the line of business, allowing for swift creation of composite applications. In *SOA World Magazine*, we give a great deal of attention to these questions.

We also look at the concept of "support services" on a fairly regular basis. Even when a business doesn't have a great deal of redundant applications, you can be fairly certain that the set of "support services" available will still be significant. These can be things such as security, auditing, logging as well as business intelligence and business activity monitoring that transcend a single application and become common services in support of line of business operations.

These support services quickly become ubiquitous, and we find that in reality, although they do not support direct business activity (i.e., they are not front-of-the-house operational services that directly produce revenue), they are in many ways just as important, if only from a risk avoidance and mitigation standpoint. Fail to have this house in order and you can be in a world of pain during an audit.

Infrastructure is what's on my mind this month, as far as services go. We're looking at SOA testing and the concepts of grid computing as they relate to SOA in this issue. And that got me thinking. Testing itself can be considered a service, at least for organizations that treat it with respect and rigor. The wealth of information that testing provides can be set up as a service to corporate dashboards and reporting systems that monitor the efficiency and effectiveness of a large IT organization. Obviously, this is mainly for large IT shops, but we can see that the principles of service orientation can be applied even to areas that are seen as supporting processes, not just to main-line business processes.

Grid computing, with the ability to bring capacity on line and to bear on a problem as needed provides another stunning opportunity to move from traditional means of operation to a service platform. Bringing CPUs to bear on a problem in a dynamic fashion, assigning additional network capacity to deal with peak loads, and allocating private connections on the fly in response to security needs are just a few of the capabilities that infrastructure vendors are building into their hardware and operating software.

Even more interesting is the concept of virtualization applied to a farm of CPUs and storage devices. Imagine a world where there are no discrete servers at all – instead, you ask for and receive a server image, complete with virtualized storage. Environments can be switched on the fly, so you no longer need three or more testing environments in order to do all your development and testing. Or even a world where you can lease out your excess capacity dynamically and recover it according to a set of SLAs that are also managed and monitored by the same management software that watches your SOA line of business services. It's here already, waiting for you to use it. Welcome to SOI. ■



About the Author

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

SOA Project Staffing Plan

Some rough guidelines

WRITTEN BY **DAVID S. LINTHICUM**

A few of my clients are now looking to staff their first inroads into SOA, their first project where something actually happens beyond the investigation. So...how many people are needed on the project? Who are they? What are their roles? Here are some rough guidelines based upon my experience thus far.

The Who (Not the Band)

You're going to need an eclectic array of skills to do SOA right, including:

- Project leader/architect
- Data specialist
- Security specialist
- Native systems specialist
- Service development specialist
- BPM/orchestration specialist
- Governance specialists
- Testing and deployment specialist
- Project archivists
- External services specialists



Note: the technology analysis and selection role is innate to all of the above.

The What

While many of the above titles are self-explanatory, it does help to define them in a bit more detail. Indeed, roles within the creation of a SOA could be a bit confusing, and the dynamics of a SOA team still need some understanding.

The project leader/architect is the person responsible for the delivery of SOA, on time, on budget, and meeting the objectives outlined when the investment was made. Typically, this is an IT project manager with an understanding of SOA, but in smaller organizations this could be the enterprise architect or even the CIO.

Data specialists are responsible for all data-related analysis, design, and deployment. Typically they have an understanding of all native data layers within the problem domain, as well as metadata and data design (logical and physical), including middleware and data abstraction layers. They also have knowledge of how data is bound to services, and work closely with the service developers.

Security specialists make sure the security that goes into the SOA is thought about at each stage of the process. SOA security (typically, identity management) needs to be systemic. This cannot be an afterthought, and a plan must be created and implemented during the project.

Native systems specialists are experts in the native systems that exist in the problem domain. In other words, they understand the operating systems and hardware, as well as application and networking interfaces. They can do performance tuning and some light development.

Service development specialists build services using service development tools, and have an understanding of how these services link back to the data layer(s) and link forward to the orchestrations or processes. They are high-end developers, really, who understand how to design, build, test, and deploy services.

BPM/orchestration specialists are those who both understand the processes as well as automate them within an orchestration layer, such as a BPEL tool or process integration engine.

— continued on page 24

About the Author

Dave is the CEO of the Linthicum Group, LLC, (www.linthicumgroup.com) a consulting organization dedicated to excellence in SOA product development, implementation, and strategy. david@linthicumgroup.com

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

Advertising Sales Director

Megan Mussa megan@sys-con.com

Advertising Sales Manager

Andrew Peralta andrew@sys-con.com

Associate Sales Managers

Lauren Orsi lauren@sys-con.com

Corinna Melcon corinna@sys-con.com

SYS-CON EVENTS

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.



_INFRASTRUCTURE LOG

_DAY 15: This project is out of control. The development team's trying to write apps supporting a service oriented architecture...but it's taking FOREVER!

_DAY 16: Gil has resorted to giving the team coffee IVs. Now they're on java while using JAVA. Oh, the irony.

_DAY 18: I've found a better way: IBM Rational. It's a modular software development platform based on Eclipse that helps the team model, assemble, deploy and manage SOA projects. The whole process is simpler, faster and all our apps are flexible and reusable. :)

_The team says it's nice to taste coffee again, but drinking it is sooo inefficient!



Rational

Download the IBM Software Architect Kit at:
IBM.COM/TAKEBACKCONTROL/FLEXIBLE



10 Principles of SOA

A frame of reference — for — SOA-related discussions

WRITTEN BY STEFAN TILKOV

➤ In many customer engagements, I need to establish a basic set of SOA principles. The following sections introduce fundamental principles that a Service Oriented Architecture (SOA) should expose. They are not introduced as absolute truth, but rather as a frame of reference for SOA-related discussion. You'll note that the first four are based on Don Box's four tenets, although over time they may have acquired a personal spin.

1) Explicit Boundaries

Everything needed by the service to provide its functionality should be passed to it when it's invoked. All access to the service should be via its publicly exposed interface; no hidden assumptions should be necessary to invoke the service. "Services are inextricably tied to messaging in that the only way into and out of

a service is through messages." A service invocation should – as a general pattern – not rely on a shared context; instead service invocations should be modeled as stateless. An interface exposed by a service is governed by a contract that describes its functional and non-functional capabilities and characteristics. The invocation of a service is an action that has a business effect, is possibly expensive in terms of resource consumption, and introduces a category of errors different from those of a local method invocation or remote procedure call. A service invocation isn't a remote procedure call.

While consuming and providing services certainly should be as easy as possible, so it's undesirable to hide too much of the fact that an interaction with a service takes place. The message sent to or received from the service, the service contract, and the service itself should all be first-class constructs within the SOA. This means, for example, that the programming models and tools that are used should at least provide an API that exposes these concepts to the service programmer. In summary, a service exposes its functionality through an explicit interface that encapsulates its internals; interaction with a service is an explicit act, relying on the passing of messages between consumer and provider.

2) Shared Contract and Schema, not Class

Starting from a service description (a contract), both a service consumer and a service provider should have everything they need to consume or provide the service. Following the principle of loose coupling, a service provider can't rely on the consumer's ability to reuse any code that it provides in its own environment; after all, it might be using a different development or runtime environment. This principle puts severe limits on the type of data that can be exchanged in an SOA. Ideally, the data is exchanged as XML documents validatable against one or more schemas, since they are supported in every programming environment one can imagine.

3) Policy-driven

To interact with a service, two orthogonal requirement sets have to be met:

1. The provider's functionality, syntax, and semantics must fit the consumer's requirements,
2. The technical capabilities and needs must match.

For example, a service provider may offer exactly the service a consumer needs, but offer it over JMS while the consumer can only use HTTP (because it's implemented on a .NET platform); a provider might require message-level encryption via the XML Encryption standard, while the consumer can only support transport-level security using SSL. So even in cases where both partners have the necessary capabilities, they might have to be "activated" – e.g., a provider might encrypt response messages to different consumers using different algorithms based on their needs.

To support access to a service from the largest number of differently equipped and capable consumers, a policy mechanism has been introduced as part of the SOA toolset. While the functional aspects are described in the service interface, the orthogonal, non-functional capabilities and needs are specified using policies.

4) Autonomous

Related to the explicit boundaries principle (5.4.1.1), a service is autonomous in that its only relation to the outside world – at least from the SOA perspective – is through its interface. In particular, it must be possible to change a service's runtime environment, say from a lightweight prototype implementation to a full-blown application server-based collection of collaborating components without affecting its consumers. Services can be changed and deployed, versioned, and managed independently of each other. A service provider can't rely on the ability of its consumers to adapt to a new version of the service quickly; some of them might not be able, or willing, to adapt to a new version of a service interface at all (especially if they're outside the service provider's sphere of control).

5) Wire Formats, not Programming Language APIs

Services are exposed using a specific wire format that has to be supported. This principle is strongly related to the explicitness of the boundaries principle, but introduces a new perspective: To ensure the utmost accessibility (and so, long-term usability), a service must be accessible from any platform that supports the exchange of messages adhering to the service interface as long as the interaction conforms to the policy defined for the service. For example, it's a useful test of conformance to this principle to consider whether it's possible to consume or provide a specific service from a mainstream dynamic programming language such as Perl, Python, or

Ruby. Even though none of these may currently play any role in the current technology landscape, this consideration can serve as a litmus test to assess whether the following criteria are met:

- All message formats are described using an open standard or human-readable description
- It's possible to create messages adhering to those schemas with reasonable effort without a specific programmer's library
- The semantics and syntax for additional information necessary for successful communication, such as headers for purposes such as security or reliability, follow a public specification or standard
- At least one of the transport (or transfer) protocols used to interact with the service is a standard network protocol (or is accessible via one)

6) Document-oriented

To interact with services, data is passed as documents. A document is an explicitly modeled, hierarchical container for data. The self-descriptiveness, as outlined in the previous section, is one important aspect of document-orientation. Ideally, a document will be modeled after real-world documents, such as purchase orders, invoices, or account statements. Documents should be designed so that they are useful in the context of a problem domain, which may suggest their use with one or more services.

Similar to a real-world paper document, a document exchanged with a service will include redundant information. For example, a customer ID might be included along with the customer's address information (although the customer ID would be enough). This redundancy is explicitly accepted since it serves to isolate the service interface from the underlying data model of both service consumer and service provider. When a document-oriented pattern is applied, service invocations become meaningful exchanges of business messages instead of context-free RPC calls. While not absolutely required, it can usually be assumed that XML will be used as the document format/syntax.

Messages flowing between participants in an SOA connect disparate systems that evolve independently of each other. The loose coupling principle mandates as little dependence on common knowledge as possible. When messages are sent in a Distributed Objects or RPC infrastructure, client and server can rely on a set of proxy classes (stubs and skeletons) generated from the same interface description document. If this is not the case, communication ceases on the assumption that the contract doesn't support interaction between those two parties. For this reason, RPC-style infrastructures require the synchronized evolution of client and server program code.

This is illustrated by the following comparison. Consider the message:

2006-03-1347113

and compare it to:

```
<order>
  <date>2006-03-13</date>
  <product-id>4711</product-id>
  <quantity>3</quantity>
</order>
```

While it's obvious that the second alternative is human-readable and the first is not, it's also notable that in the second case a

following the principle of loose coupling, a service provider can't rely on the consumer's ability to reuse any code that it provides in its own environment

participant that accesses the information via a technology such as XPath will be much better isolated against smaller non-breaking changes than one that relies on the fixed syntax. Conversely, using a self-descriptive message format such as XML while still using RPC patterns, such as stub and skeleton generation, serves only to increase XML's reputation as the most effective way to waste bandwidth. If one uses XML, the benefits should be exploited too. (See this paper for an excellent discussion of why many current Web Services stacks fail this test.)

7) Loosely Coupled

Most SOA proponents will agree that loose coupling is an important concept. Unfortunately, there are many different opinions about the characteristics that make a system "loosely coupled." There are multiple dimensions in which a system can be loosely or tightly coupled, and depending on the requirements and context, it may be loosely coupled in some of them and tightly coupled in others. Dimensions include:

- **Time:** When participants are loosely coupled in time, they don't have to be up and running at the same time to communicate. This requires some way of buffering/queuing in between them, although the approach taken for this is irrelevant. When one participant sends a message to the other one, it doesn't rely on an immediate answer to continue processing (neither logically nor physically).
- **Location:** If participants query for the address of other participants they intend to communicate with, the location can change without having to reprogram, reconfigure, or even restart the communication partners. This implies some sort of lookup process using a directory or address that stores service endpoint addresses.
- **Type:** In an analogy to the concept of static versus dynamic and weak versus strong typing in programming languages, a participant can either rely on all or parts of a document structure to do its work.
- **Version:** Participants can depend on a specific version of a service interface or be resilient to change (to a certain degree). The more exact the version match has to be the less loosely coupled the participants (in this dimension). A good principle to follow is Postel's Law: Service providers should be implemented to accept as many different versions as possible, and thus be liberal in what they accept (and possibly even tolerant of errors), while service consumers should do their best to conform to exact grammars and document types. This increases the overall system's stability and flexibility.
- **Cardinality:** There may be a 1:1 relationship between service consumers and service providers especially in cases where a request/response interaction takes place or an explicit message queue is used. In other cases, a service consumer (which in this case is more reasonably called a "message sender" or "event source") may neither know nor care about the number of recipients of a message.

- **Lookup:** A participant that intends to invoke a service can either rely on a (physical or logical) name of a service provider to communicate with, or it can do a lookup operation first using a description of a set of capabilities instead. This implies a registry and/or repository that can match the consumer's needs to a provider's capabilities (either directly or indirectly).
- **Interface:** Participants may require adherence to a service-specific interface or they may support a generic interface. If a generic interface is used, all participants consuming this generic interface can interact with all participants providing it. While this may seem awkward at first, the principle of a single generic (uniform) interface is at the core of the World Wide Web's architecture.

It's not always feasible or even desirable to create a system that's loosely coupled in all of the dimensions mentioned above. For different types of services, different tradeoffs have to be made.

8) Standards-compliant

A key principle to follow in an SOA approach is reliance on standards instead of proprietary APIs. Standards exist for technical aspects such as data formats, metadata, transport, and transfer protocols, as well as for business-level artifacts such as document types (e.g., in UBL).

9) Vendor-independent

No architectural principle should rely on any particular vendor's product. To transform an abstract concept into a concrete running system, it's unavoidable to decide on specific products, both commercial and free/open source software. None of these decisions must have implications on an architectural level. This means relying on both interoperability and portability standards as much as is reasonably possible. As a result, a participant can be built using any technology that supports the appropriate standards and not be restricted by any vendor roadmap.

10) Metadata-driven

All of the metadata artifacts in the overall SOA need to be stored in a way that enables them to be discovered, retrieved, and interpreted at both design time and runtime. Artifacts include descriptions of service interfaces, participants, endpoint and binding information, organizational units and responsibility, document types/schemas, consumer/provider relationships, etc. As much as possible, using these artifacts should be automated by either code generation or interpretation and become part of the service and participant lifecycle. ■

About the Author

Stefan Tilkov is co-founder and a principal consultant at innoQ, a consulting firm with offices in Germany and Switzerland. He focuses on enterprise architecture consulting for Fortune 1000 companies, which currently translates to assessing SOA maturity and deriving the appropriate steps for a roadmap leading to a service-oriented enterprise. Stefan blogs at www.innoq.com/blog/st/

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

Racing without a Speedometer

Is Your Accelerator Really Solving Your Network Problems?

WRITTEN BY HON WONG

➤ Accelerators, Application Delivery Controllers (ADCs), application switches...whatever you call them, this new breed of performance enhancing appliances is selling like hotcakes. The market for these turbochargers has zoomed from zero to over \$1 billion in the span of a few years, heading towards what Gartner estimates will be \$3.7 billion in annual sales in 2008.

Yet even as IT organizations are spending up to \$100,000 a pop on these boxes, few have any idea how well they're working. These high-octane accelerators are like Formula One racecars without a speedometer — the pedal is to the metal, but is it fast enough? Racing without a speedometer might be fine on an empty test track, but as enterprises deploy increasingly complicated and crowded Service Oriented Architectures (SOAs) with hundreds or even thousands of Web Services, not knowing the speed you have to spare can be a prelude to an ugly crash.

Fortunately, there are ways to measure performance, ranging from using separate monitoring appliances to several vendors' plans to build speedometer technology directly into the accelerators.

This article will discuss the actual impact of acceleration on network and application problems in Service Oriented Architectures using real-world examples, and compare and contrast the efficacy and efficiency of the various speedometers available.



SOAs, Web Services, and the Network

Service Oriented Architectures and Web Services are revolutionizing application development. Web Services are loosely coupled modules of code that communicate with each other and the outside world via specific standard protocols such as SOAP-formatted XML. The key is that interactions with each Web Service follows a rigorous standard, which abstracts the application logic so that the users or other Web Services trying to access a particular service can do so without requiring any knowledge of the underlying network and server infrastructure that the Web Service runs on. This allows application developers to focus on application development, rather than learning network topology or server configurations.

Developers can write new Web Services or they can use products that expose existing code as a Web Service. Those same developers, or any others for that matter, can then rapidly create complex “composite” applications out of these component Web Services, saving time by reusing common Web Services for multiple applications.

Using these flexible technologies, enterprises can lower the cost of Web application development, shorten the development cycle, and improve the flexibility of business processes. This increased speed and adaptability are very appealing to businesses, especially as approaches like server virtualization, “Web 2.0,” and Software as a Service (SaaS) become more prominent in the enterprise. When business runs on the Web, every business has to run on Internet time.

However, while in theory Web Services and SOAs abstract away low-level issues like network performance (one of the reasons that they are wildly popular with application developers), in practice, they actually increase the impact of network issues and the importance of performance monitoring.

When Applications Attack: How Slowdowns Impact SOAs

While Web Services and SOAs may allow application developers to avoid considering issues such as server capacity and network topology while creating a new application, the applications that they create must run in the real world. No matter how many layers of logical abstraction separate the application code from the actual infrastructure, the fact remains that at the end of the day, Web Services still rely on CPUs executing code, and networks carrying packets of information back and forth between them.

As a result, network and server slowdowns will still impact the performance of Web Services and SOAs. In fact, it's likely that the Web Services built into SOAs are more vulnerable to slowdowns than their predecessor applications. Composite applications, by definition, weave together a complex web of interdependent services. And thanks to the aggressive deployment of reusable components, a single Web Service slowdown may have far-reaching consequences for many different applications, rather than taking down a single monolithic application, as may have been true in the past.

Moreover, abstraction doesn't come for free. Using the SOAP protocol to exchange information via XML may be a much cleaner method from a code standpoint, but from a network standpoint this XML traffic is bulkier than the communications it replaces. Even when network conditions are good, Web Services and SOAs require increased bandwidth, and when things go wrong, this added overhead goes straight to the performance bottom line.

Clocking the Impact of Acceleration

As a result of the vulnerabilities of Web Services and SOAs to performance issues, many organizations have turned to application acceleration appliances to improve performance to acceptable levels. These network appliances use various techniques to improve application performance, including compressing the data to be transmitted, caching commonly requested data, off-loading non-essential processing such as SSL processing from Web servers to a dedicated device, and prioritizing bandwidth for critical applications.

There are two principal types of application accelerators, symmetrical and asymmetrical. Symmetrical accelerators put acceleration appliances on both the end-user and the data center ends of the network pipe. Some vendors sell pairs of appliances that can, for example, accelerate the speed at which a branch office can access files in a centralized data center. These symmetrical accelerators play a critical role in WAN strategy, but less so in Web Services and SOA because of their highly distributed nature. When Web Services users can exist outside the boundaries of the enterprise, it's important to have an acceleration product that can serve anyone or anything trying to access the service in question.

The second kind of application accelerator, and the primary focus of this article, is the asymmetrical accelerator that puts one or more acceleration appliances in the data center and does all the compression at the server rather than at the branch office or client level. These asymmetrical accelerators have become common for both enterprise and customer-facing applications, and leading vendors have done a brisk business in accelerator boxes by helping application owners get the performance that they need from their new applications, including Web Services and SOA deployments.

For example, one Midwestern state needed to replace the 20-year-old mainframe system it used to distribute information about criminal cases and defendants (www.citrix.com/English/aboutCitrix/caseStudies/caseStudy.asp?storyID=22594). The state upgraded to a

modern Service Oriented Architecture based on the Microsoft .NET platform. The new Courts Information System (CIS) consolidated this vital information, as well as making it accessible to all of the different courthouses (87 in all), judicial officials, and courthouse employees across the state's court system, including criminal, juvenile, civil, family, small claims, mental health, probate, and traffic courts.

But while this new system offered great improvements in functionality, it also imposed greater demands on the available infrastructure. The project's technical lead observed, "In the past, the mainframe sent small packets of information to the green screens in each county. [The project] is browser-based, which gives users a much richer set of graphical features but also generates more traffic and larger XML and HTML data packets. If we wanted to run the new system over existing pipes then we knew that highly efficient bandwidth compression would be a key success factor."

The state turned to application switches to accelerate the performance of the applications and services running on its Microsoft infrastructure. Yet while these acceleration appliances were able to achieve and document an 80% reduction in bandwidth usage, they didn't quantify the accompanying improvements in end-to-end response times.

Measuring Feeds and Speeds

The key to measuring the impact of application acceleration is to be able to monitor the before-and-after performance of the relevant Web Services, to be able to watch a "speedometer" as the application revs from 35 to 100 mph. And just as there are several different flavors of acceleration appliances, so there are also several different ways to measure speed.

Service-based

The oldest variety of measurement is service-based monitoring. Service bureaus operate a global network of monitoring stations that transmit synthetic requests to the Web pages you want tested. You can then log into the service provider's system to see how quickly your SOA could respond to artificial requests from a variety of locations around the globe.

These service-based measurement tools are a good first step, since they require little upfront investment, and can also be used for testing pre-launch applications that don't yet have any real load on the system. However, they are less helpful in monitoring production Web Services, since they can't replicate all the different locations that might call a service, or replicate the complex interactions generated by real system load.

Appliance-based

The next variety of measurement device is the appliance-based speedometer. These monitoring appliances sit in your data center alongside your acceleration appliances, generally off a mirror port. The appliance-based speedometers use network sniffing techniques to estimate the end-to-end response times that your Web Service users experience.

Unlike the service-based measurement, these appliance-based devices can monitor all the requests and transactions of true production load, making them much more helpful for production monitoring. However, because they're completely passive, their data reflects a statistical estimate rather than an actual measurement. In addition, the cost of these appliances generally rivals those of the actual accelerators themselves, which means that you're paying just as much money and setting aside just as much room in your data center for your speedometer as for your actual motor (the accelerator).

Software-based Speedometers

The final variety of measurement tools is the software-based speedometer. These speedometers measure the actual end-to-end response times of your Web Services.

Like the appliance-based devices, software-based speedometers can monitor all the requests and transactions of true production loads. Yet unlike the appliance-based devices, software-based speedometers offer a much leaner, more efficient solution that doesn't require additional boxes in your data center.

Because they can provide complete end-to-end tracing of .NET and J2EE applications as well as detecting Web Service slowdowns, end-to-end monitoring software can also trace the root cause of those slowdowns on the back-end, down to individual method calls and SQL queries.

As a drawback, however, software-based speedometers require IT groups to install software rather than simply plugging in an appliance.

Beyond Measurement: Tuning the Engine

Of course, even if you have the right measurement device to determine the speed of your Web Services, you're job still isn't done. Let's say your speedometer tells you that your apps are crawling along at 35 mph when the desired speed is 65. Now you need to know whether to replace the spark plugs, change the carburetor, or maybe just find a smoother road. What you need is actionable information that allows the IT organization to tune the infrastructure and application development to tune the app.

Nor can this information be limited to aggregate data from the various tiers of the infrastructure, which masks information about individual users and transactions. Most tools for tuning deal with how the aggregate performance is affected by tuning the areas they point out as bottlenecks. However, the CEO and CIO must also deal with specific user complaints caused by specific transaction issues. These are only visible after the transaction has taken place and there are no tools to preemptively alert executives to these problems. Like IT they only hear about them after the irate call. More importantly, they have no visibility into the impact these slowdowns are having on the business – from revenue to reputation.

Combining an accelerator appliance with a speedometer that allows the diagnosis of specific issues offers IT organizations a complete solution.

For example, a regional multiple listing service (MLS) experienced a series of service outages that affected a key application for its subscribers (www.f5.com/solutions/success/nwmls_ss.html). This application gave real estate agents access to a centralized database of listings, market analysis, tax information, and zoning on a broad range of properties. Naturally, performance was critical.

The problem was that the system was running into infrastructure capacity issues. For example, the MLS would often have problems with its Internet Service Providers. According to their network administrator: "We'd be forced to wait while the ISP's webmaster tried to diagnose a routing problem and determine whose fault it was."

By implementing a load balancing solution, the regional MLS was able to accelerate application performance and increase availability as well. "Customers no longer have to pay the price if one

of the providers has system problems or is getting flooded," the network administrator said. "There's no more down time or waiting while third-party ISPs try to find the issue. There's no more latency in failing over or waiting for our routers' tables to update through the Internet. It's a whole new level of high availability for Internet connectivity – failover is instantaneous."

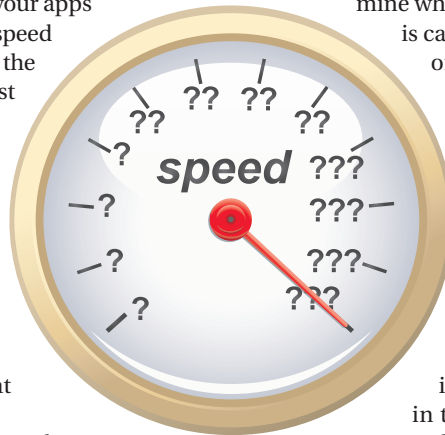
In another case, a Web 2.0 company used an acceleration appliance in conjunction with a software-based speedometer to accelerate its Web Services, as well as to identify and fix individual code issues (www.symphoniq.com/solutions/videos/wallop.asp).

In the end, even the best application accelerator cannot fix flaws in the infrastructure or code. The company brought in an end-to-end transaction management tool to measure the impact of the acceleration appliance, as well as for its ability to tag-and-trace sluggish real user actions through the application infrastructure to pinpoint hotspots and bottlenecks. This data lets developers and operations staff quickly work around infrastructure problems or application anomalies to ensure a smooth deployment and production ramp.

"[Symphoniq's product] TrueView gives my developers a clear roadmap to problem resolution using real user data to determine which tier of the application is sub-par, and what is causing the slowness," said the company's VP of engineering. "With a few clicks, the product identifies the servers, method calls, or SQL queries that are causing the problem, saving a lot of time we would otherwise have to spend reproducing, investigating, or debating over the details."

Conclusion

As we've seen, the rapid adoption of Web Services and Service Oriented Architectures is increasing, rather than decreasing, the importance of network performance in the enterprise. As a result of the pressure exerted by the added complexity and overhead of SOAs, companies are increasingly turning to application acceleration vendors to improve performance. However, while these solutions work, they leave IT organizations dangerously in the dark about actual performance conditions. The answer to these issues is to choose the right speedometer for your enterprise. The different types of speedometers — service-based, appliance-based, and software-based — offer different functionality, but with a shared goal of allowing IT organizations to understand the true gains of acceleration. Once these gains are documented and analyzed, IT organizations should look to their tools to provide the actionable information needed to tune the infrastructure and code for optimum performance. ■



About the Author

Hon Wong enjoys solving the problems that result from increasing system complexity and has founded three successful companies that do so: Ecosystems, NetIQ, and Symphoniq. With EcoSystems, he set out to manage client/server systems. Next he founded NetIQ to keep Wintel systems running and developed the technology underlying early versions of Microsoft's MOM. With Symphoniq, he addresses the requirements of the Web, including Web Services, SOA, and Web 2.0 — the most complex and distributed applications ever created. Hon has spoken at numerous conferences including Interop 2004, is published in DM Review and Communications News, and is a prolific blogger (symphoniq.com).

Fiorano SOA™ 2006

The Quickest path to an SOA

- ✕ FioranoMQ™ 2006 – world's fastest, most scalable JMS
- ✕ Fiorano ESB™ 2006 – CAD/CAM for distributed applications
- ✕ Fiorano BPEL Server – simplifying business process orchestration
- ✕ Fiorano Tools – BPEL Studio, Mapper, FEPO, etc
- ✕ Fiorano Components – 60+ pre-built adapters



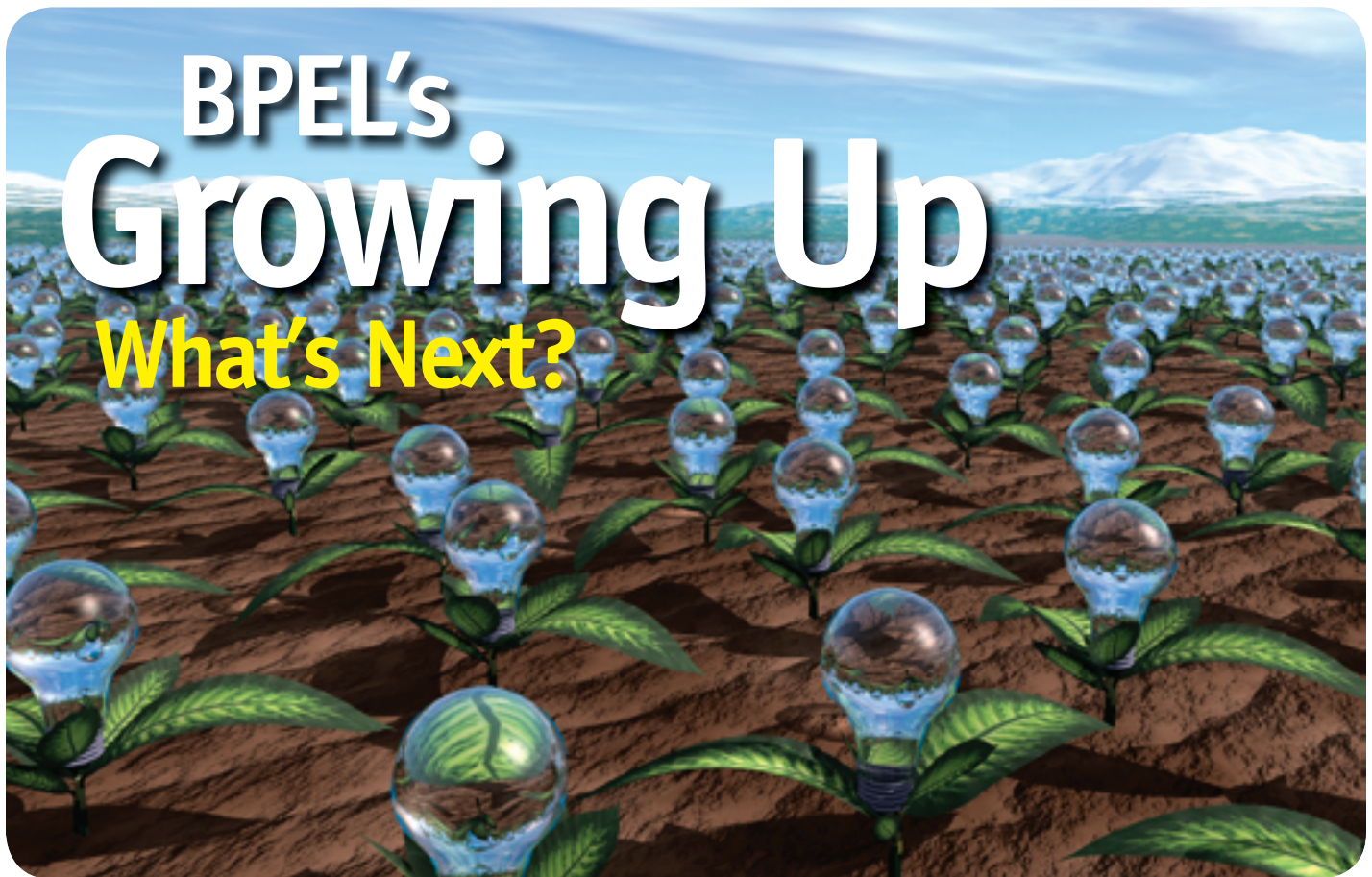
Benefits

- ✕ Adherence to popular industry standards - JMS, COM, .NET, JCA, JMX, BPEL, SOAP, etc.
- ✕ Multi-language, Multi-platform, Multi-protocol
- ✕ Unmatched Scalability and High Performance
- ✕ Quick, Measurable ROI

Download your copy of Fiorano today!

www.fiorano.com/downloadsoa

Fiorano®
Enabling Change at the speed of thought



BPEL's Growing Up

What's Next?

WRITTEN BY DAVID SHAFFER AND MANOJ DAS

➤ IT architectures have evolved to include process orchestration as a fundamental layer due in no small part to the emergence and widespread adoption of the WS-BPEL standard. WS-BPEL, also known as Business Process Execution Language or just BPEL, is a standard owned by OASIS that provides rich and comprehensive orchestration semantics. This article will provide a brief overview of how BPEL came to be what it is today and then focus on the latest developments in the BPEL standard and where we believe this standards area will go over the next few years. In particular some of the key areas for growth in this space include the standardization of human workflow support and better integration with process modeling and analysis tools and standards.

A Brief History of BPEL

Much of BPEL's broad adoption and acceptance comes from its ancestry. Historically, BPEL emerged when IBM and Microsoft joined forces and merged their proprietary workflow languages, WSFL and XLANG, respectively, into a next-generation business process language specification that at the time was called BPEL4WS (Business Process Execution Language for Web Services). BPEL4WS 1.0, known as the "license plate standard," was first publicly released in August 2002 by IBM, BEA, Microsoft, SAP, and Siebel Systems (BEA, SAP, and Siebel joined the initiative just before the publication of the specification.) Though BPEL was a new specification at the time, even then it was relatively mature due to its underpinnings, having been built on workflow languages that had been production tested over several years. This has benefited BPEL greatly, making it much more mature than its age would otherwise indicate. In fact, an educational institution in the Netherlands has published a site with an academic analysis of the comparative richness of workflow languages that demonstrates that while BPEL isn't perfect, it's better than any of its predecessors at implementing different workflow patterns. (see <http://is.tm.tue.nl/research/patterns/standards.htm>)

What BPEL Is Today

After BPEL4WS 1.0, a minor 1.1 upgrade was released in May 2003 and was submitted officially to OASIS. A WS-BPEL technical

committee was assembled that is now one of OASIS' largest technical committees. From there, the standardization process slowed evolution significantly (as often happens...) and since then, OASIS has been working to release WS-BPEL 2.0. As such, most of the public implementations today of BPEL tools are based on the BPEL 1.1 specification. However, as BPEL 2.0 is very close to being finalized, several vendors including Oracle have implemented varying degrees of BPEL 2.0 functionality in currently shipping products.

In any case, BPEL has become deeply entrenched in the enterprise IT toolkit. We now see developers get excited about working on BPEL projects because it keeps their skills up-to-date. To get a sense of the current adoption of the BPEL standard, a search on a job search site, SimplyHired.com, for job postings with BPEL in their description yields hundreds of current job postings for BPEL-related positions (<http://www.simplyhired.com/index.php?ds=sr&q=BPEL>).

Where We Are Going

People who are looking to point out weaknesses in the BPEL standard often comment that it does not include support for human workflow tasks. This is true, of course, even with the BPEL 2.0 standard. However BPEL does have rich support for asynchronous services and so one approach is to build a human workflow service engine that BPEL processes can then call out to. With this architecture, human tasks and manual steps can be incorporated in 100% standard BPEL process flows, just like any other asynchronous service. This architecture was detailed in a Web Services Journal article "BPEL Processes and Human Workflow" dated April 12, 2006 by Matjaz Juric and Doug H. Todd (<http://webservices.sys-con.com/read/204417.htm>). This approach has been adopted by several vendors including Oracle; we believe this provides a very clean architecture while the standardization process catches up in this area.

Going forward, we're already seeing the next generation of standards around BPEL being discussed. For example, the "BPEL4People" effort was first announced in late 2005 and is intended to standardize an approach similar to the one described above for incorporating human workflow tasks in BPEL processes. Besides being one of our favorite standards acronyms, BPEL4People is an important area of work since most business processes span both systems and humans. It also answers the question once and for all as to whether BPEL is properly pronounced "bepple," "bee-pull," or "bee-pell." (Answer – it must rhyme nicely with "people".)

Another area we see evolving is tighter integration between a process implementation language like BPEL and standards like BPMN that describe a business process modeling notation – a business analyst-friendly visual representation of a process. Since BPEL says nothing about the visual representation of a process and BPMN says nothing about the save format, they would seem like a perfect match. In practice, there are still some gaps to be filled, but in general we believe that tighter coupling between the standards (and tools) for business analysts and process developers will be a fantastic development for the IT world at large.

In the next sections we look in more detail at these growth areas that will expand the reach of business process standards and help BPEL achieve its full potential.

Process Orchestration

Business processes span services, applications, and human activities; these processes need to be orchestrated in an agile fashion with end-to-end control, visibility, and rich exception management. Process orchestration is the heart of Business Process Management enabling creation of executable business processes from services and human activities.

Process Orchestration requirements include:

- Sequencing, including serial, parallel, and other control flow patterns
- Exception handling including error conditions, transactions, and compensation
- Data flow and transformation
- Event handling including timers and other out-of-band events

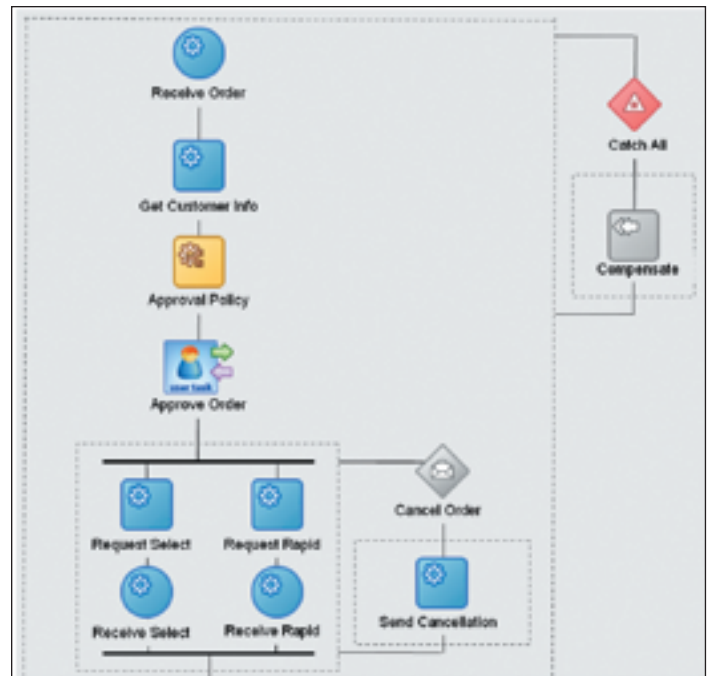


Figure 1: A visual representation of a BPEL process

BPEL today addresses these requirements in a mature fashion. Some of the salient features of BPEL include:

- Rich sequencing semantics including parallel and asynchronous processing
- A compensation-based long-running transaction model
- Rich scoped fault-handling capabilities
- Asynchronous event handling allowing time-based alerts as well as out-of-band events like a status request or cancellation event
- A Web service-based model for process decomposition and assembly: each BPEL process exposes a Web Services interface and can be easily composed into other higher-level compound flows
- Standard use of XML and XPath for data access and manipulation

BPEL Adoption

While the formal standardization of BPEL is nearing completion at OASIS, it has already gained significant market traction. Platform vendors such as Oracle are shipping commercial BPEL engines today and a Google search shows at least four available open source

BPEL implementations. Eclipse has a project in progress to add comprehensive support for the authoring and editing of BPEL processes. And we are personally aware of thousands of production BPEL processes deployed by hundreds of customers. Finally, many companies now support BPEL as an interchange format. For example, modeling tool vendors including IDS Scheer and others support the generation of BPEL from their process modeling and analysis tools, which can then be shared directly with developers.

Since the BPEL 2.0 standard has yet to be released, most of the commercial and customer implementations of BPEL are based on the 1.1 specification. All of this enthusiasm around BPEL 1.1 leads to two natural questions:

- 1) What improvements can be expected in BPEL 2.0?
- 2) What migration effort will be required for current implementations to move to BPEL 2.0?

WS-BPEL 2.0

WS-BPEL 2.0, commonly known as BPEL 2.0, is the evolution of BPEL 1.1 as a public OASIS standard and is currently available as a draft for public review. In fact, by the time this article is published, standardization of WS-BPEL 2.0 may be complete. Some of the salient changes in BPEL 2.0 from BPEL 1.1 are:

- **Improved data access** – Data access has been improved in BPEL 2.0. Variable XPath binding makes BPEL variable data available in greatly simplified XPath expressions (where a \$variable syntax can be used without requiring use of bpws:getVariableData). The example shown below illustrates how much simpler and more readable the BPEL 2.0 syntax is. BPEL 2.0 also adds support for variables based on XSD complex types.
- **Improved data manipulation** – BPEL 2.0 enables complex XML

Requirement: We have a *message*, that has a *part*, whose root element has unbounded element (aka array) *item*. We need to select the *item* whose index is *counter*.

BPEL 1.1:
`expression="bpws:getVariableData('message','part',
concat("/ns:root/item [", bpws:getVariableData('counter'), "]"))"`

BPEL 2.0:
`$message.part/item[$counter]`

Figure 2: Simplified data access

transformations in BPEL processes by introducing `bpel:doXsl-Transform()`. Other improvements include inline variable initialization and making the assign activity extensible.

- **New activities** – BPEL 2.0 introduces some new activities, the most significant of which is `forEach` that executes a series of activities a variable number of times, either serially or in parallel. Early implementations of parallel `forEach` have been available for several years in the Oracle BPEL Process Manager as the `flowN` activity and were contributed to the OASIS discussion, which resulted in the new `forEach` activity. Other new BPEL 2.0 activities include `repeatUntil`, which repeats activities until a specified condition is true, and `extensionActivity`, which improves the ability of BPEL implementations to provide new activities.
- **Enriched Fault Handling** – BPEL 2.0 adds finer-grained controls

Requirement: We have an *Order* with multiple line items. We need to invoke some processing for each line item in parallel.

```
<forEach counterName="idx" parallel="yes">
  <startCounterValue>1</startCounterValue>
  <finalCounterValue>count($Order/LineItem)
</finalCounterValue>
  <scope>
    <variables>
      <variable name="item" ...>
        <from>$Order/LineItem[$idx]</from>
      </variable>
    </variables>
    ... Process item ...
  </scope>
</forEach>
```

Figure 3: Parallel `forEach` activity

in the catch construct and a rethrow activity. Some advanced fault-handling features have also been introduced such as a termination handler.

- **Advanced Message Operations** – The most common example in

Rethrow activity is very useful for clean up prior to propagating a fault. For example, the default fault handler behavior is:

```
<catchAll>
  <sequence>
    <compensate />
    <rethrow />
  </sequence>
</catchAll>
```

If custom clean up is needed, the call to `compensate` may be replaced by appropriate calls.

Figure 4: Rethrow activity

this category is that BPEL 2.0 allows local partner links, another area where existing implementations can already be found. Note that having local partner links is a requirement in many cases, but is specifically needed in the `forEach` example above. If you want to be able to invoke many services in parallel with parallel `forEach`, you would need to create a separate partner link locally in each loop iteration and BPEL 2.0 explicitly allows this.

- **Syntax Improvements** – Many changes in BPEL 2.0 are syntactical in nature; examples include changes from 'switch' to 'if-elseif-else' from 'terminate' to 'exit' and differentiating between cases of 'compensate' by calling them 'compensate' and 'compensateScope'.
- **Clarifications** – A number of changes in BPEL 2.0 clarify ambiguities in the 1.1 spec. Most of these have been driven by vendors with current BPEL implementations.

Migration from BPEL 1.1 to 2.0

As mentioned, many organizations are in production with processes built around the BPEL 1.1 specification and this group, including companies starting BPEL development today, is very interested in the effort that will be required to migrate from version



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

In the very near future, WS-BPEL 2.0 will further mature the BPEL standard while providing a clean migration path for current BPEL implementations

1.1 to 2.0. We see three factors that suggest this upgrade will be smooth and non-intrusive for most companies:

- **Backward Compatibility** – BPEL 1.1 and 2.0 syntaxes are mostly complementary. This enables vendors to provide backward-compatible support for BPEL 1.1 constructs along with BPEL 2.0 implementations.
- **Syntactic Transforms** – Many of the differences between BPEL 1.1 and 2.0 are purely syntactic. This allows XSLT transforms and other tools to automate much of the migration work. We can have a high degree of confidence in this because the BPEL 1.1 specification has been out for several years and the 2.0 standard has changed very little over the past year.
- **Shared Pain** – Finally, where tools can't do all the work and some manual migration is required, the large numbers of customers with this same need will encourage a broad set of best practices documents and examples to be made available.

Longer-Term: Process Model 'Round Tripping'

Involving business stakeholders in the analysis and modeling of business processes usually leads to fewer development iterations and a closer match between implementation and business requirements. This is a very common desire and in many cases the driver for adopting BPM. While BPEL vendors provide easy-to-use graphical tools for creating and editing BPEL processes, the very fact that BPEL processes are so detailed as to be executable makes these tools too complex for most business users. Instead, business users need to be able to specify higher-level process blueprints that can then be filled in by developers to make them executable.

Business Process Modeling Notation (BPMN) is a standard from OAG to address the above requirement. Using BPMN as a standard notation, business users may create process blueprints and share them with developers and other stakeholders. BPMN maps in a straightforward fashion to BPEL and developers can work on the BPEL view of the process to make it executable. It is possible for vendor tools to maintain round-tripping by anchoring the BPEL to the BPMN blueprint, though this requires extensions to one or both standards, in their current forms. This is an area in which we are actively working and have publicly demonstrated a common meta-model format that can be shared between a high level process modeling tool and an implementation level developer tool. By main-

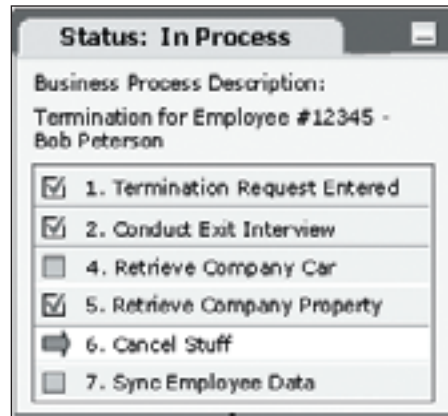


Figure 5: Peoplesoft you are here list

taining the high-level analyst annotations in the runtime BPEL code, we believe that the nirvana of full round-tripping between business analyst tools and developer tools can be achieved. In fact, the support for maintaining the high-level process model at runtime has already been leveraged by our Peoplesoft group for what they call a “you-are-here-list” that shows end users the runtime state of a business process based on the high-level process states defined by business analysts.

Conclusion

Process orchestration is already a key component of integration, SOA, and BPM. BPEL 1.1 is a mature process orchestration

language widely supported by vendors and is successfully addressing process orchestration requirements at many production customer sites today.

We believe customers who adopt BPEL and related standards for their BPM and SOA architectures will gain the following benefits:

- Comprehensive functionality
- Knowledge portability
- Toolset extensibility and “hot-pluggable” products
- Interoperability
- Vendor portability

In the very near future, WS-BPEL 2.0 will further mature the BPEL standard while providing a clean migration path for current BPEL implementations. Longer-term standards such as BPEL4People and BPMN will work closely and cleanly with BPEL to complete the puzzle and will help increase the market adoption of BPM by mainstream enterprises. ■

About the Authors

David Shaffer is senior director of product management for Oracle Fusion Middleware. He leads product management for Oracle's integration products including Oracle Enterprise Service Bus, Oracle BPEL Process Manager, Oracle Business Activity Monitoring, and Oracle Business Rules.

david.shaffer@oracle.com

Manoj Das is senior manager of the Product Management Group for Oracle Fusion Middleware. His focus is on BPEL and business rules. Manoj joined Oracle from the Siebel acquisition where he was responsible for driving the next-generation process-centric application platform.

manoj.das@oracle.com

Achieve Point-and-Click SOA™

exteria™

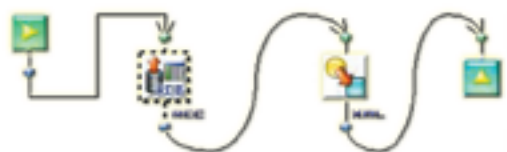
Business Automation Platform

*Get on the SOA
fast track!*

NEXT STOP: extentech.com for
FREE Exteria Downloads
Demos and Whitepapers

High-Speed Productivity

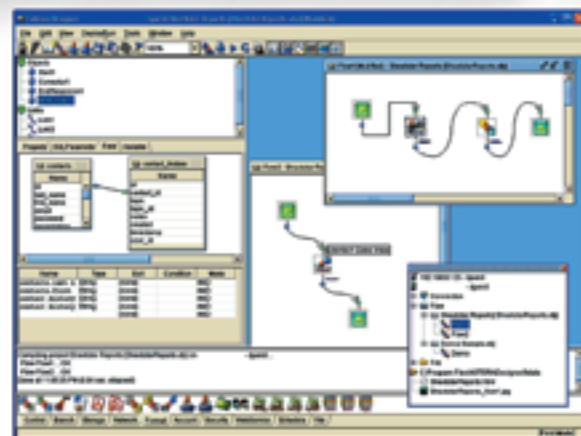
Exteria is the ideal modeling tool and web services platform for rapid SOA implementations.



Model data flows graphically in the code-free designer then publish with a click to XLS, PDF, SOAP, XML, and more -- all without writing any code!

Spreadsheet Automation

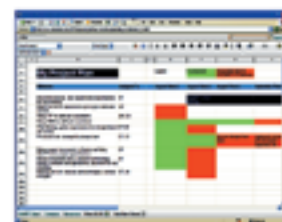
SOA is all about re-use. There is no need to reinvent business logic already available in spreadsheets. Leverage this asset by mapping data flows to formula cells and publishing them as RSS feeds, SOAP web services, or to an AJAX web spreadsheet.



The Exteria data flow modeler features dozens of connectors including LDAP, SOAP & XML.

works with:
sheetster™

Integration with Sheetster.com web spreadsheets enables realtime reporting and collaboration.



includes:



Leverage the power of Java Spreadsheet Automation with the best ExtenXLS ever!

Copyright 2007, Extentech Inc., Portions Copyright 2001-07 Infoteria Corporation. All Rights Reserved.

extentECH
exceed expectations

sales@extentech.com
call 415.759.5292
<http://extentech.com/itsg>

Testing the “REST”

Techniques and strategies for testing non-SOAP-based Web Services

WRITTEN BY V. NIRANJAN

➤ Software testing is almost the “last line of defense” in the software lifecycle between the software vendor and the customer. Testing is the phase that completely validates your architecture, design, and development. If not done properly, testing can carry bugs that can directly impact an organization’s revenue. So huge stakes hinge on this part of the lifecycle.

Web Services have now become a mainstream technology, and are steadily reaching maturity as a technology. The proliferation of Web Service technologies can be gauged from the amount of commercial and enterprise implementations that are currently taking place.

The certificate of adoption of Web Services can be gauged from the fact that two of the leading open system technologies, i.e., J2EE and the .NET Framework have now included Web Services in their standard offerings.

With all the attention that Web Services have been getting in the past year, there are remarkably few sources of information about testing Web Services, in particular REST-based Web Services. We’ll try to provide some insight into the techniques and strategies for testing REST-based Web Services.

Web Services Styles (SOAP versus REST)

The most popular Web Service implementation styles for most projects would boil down to deciding between SOAP (Simple Object Access Protocol) and REST (Representational State Transfer). Enterprise architect communities have people backing both styles pulling the enterprise projects in one direction or another. The essence of both implementation styles is basically passing XML messages over the wire generally using the HTTP protocol. However, the internal workings of both styles are vastly different from each other.

SOAP-based Web Services

SOAP-based Web Services are based on a set of standard specifications backed by the World Wide Web (W3C) and OASIS consortia. This is the primary reason for the success of this implementation style. It has all the technology vendors agreeing to the basic set of standards that allows technologies to become interoperable. There are also other reasonably mature standards like WS-Security that address the non-functional aspects.

SOAP-based Web Service implementations are based on three important constituents.

The Simple Object Access Protocol

SOAP is a stateless, lightweight, XML-based, one-way message exchange protocol. It basically acts as the medium of communication between the service consumer and the service producer. SOAP specifications define an extensible framework by which application

specific information can be exchanged.

SOAP isn't specifically tied to a particular protocol but it's been widely used with HTTP. One of the biggest advantages of SOAP is its ability to handle high levels of complexities. It can deal with basic data types, structs, even arrays.

SOAP doesn't address the semantics of the application-specific messages used for communication between the service consumer and the provider. The SOAP skeleton contains a SOAP envelope with an optional SOAP header element and a compulsory SOAP body element. Any processing error is sent as a SOAP fault in the SOAP message.

The SOAP header element is optional. However it's used to send authentication and session information. Although this doesn't lie within the SOAP protocol, it does provide the flexibility that's exploited by standards like WS-Security. The header element is meant to encapsulate extensions to the message format without having to couple them to the payload or modify the fundamental structure of SOAP.

The SOAP body element is mandatory and contains the application specific information that has to be exchanged between the consumer and the provider. This payload could be a remote procedure call, a XML document that's interchanged.

Web Services Description Language (WSDL)

The Web Service Description Language is an XML-based interface definition and description format used to describe and publish Web Services in a standard manner. The WSDL definition lets consumers identify the location of the service, the request messages that the service requires, and the corresponding output message format that the consumer will get.

WSDL lets service providers expose only their interfaces describing the location and the message formats and completely hiding the underlying implementation. So as long as the service interaction is on the basis of the defined WSDL as the interface, the service provider can keep changing the implementations without it affecting the consumer.

Universal Description Discovery and Integration (UDDI)

The UDDI specifications let businesses publish their services in private and public registries so businesses and people can find and transact with one another easily and dynamically. UDDI lets a business:

- Describe its business and its services
- Discover other businesses that offer desired services
- Integrate with these other businesses.

UDDI is made up of three different elements. Each of the elements isn't required for a listing so not all listings in UDDI registries contain all the elements. But the more elements in a listing, the more easily a service can be found and bound to.

The three elements are:

- A **"white pages"** – This contains the basic contact information for each Web Service listing. It generally includes basic information about the company, as well as how to make contact.
- A **"yellow pages"** – This has more details about the company and includes descriptions of the kind of electronic capabilities the company can offer to anyone who wants to do business with it. It uses commonly accepted industrial categorization schemes, industry codes, product codes, business identification codes, and

the like to make it easier for companies to search through the listings and find exactly what they want.

- A **"green pages"** This is what allows someone to bind to a Web Service after it's been found. It includes the various interfaces, URL locations, discovery information, and similar data required to find and run the Web Service.

REST-based Web Services

In the Web Services world REpresentational State Transfer (REST) is a key design idiom that embraces a stateless client/server architecture in which the Web Services are viewed as resources and identified by their URLs. Representational State Transfer (or REST) is based on the concept of transferring state between two systems on a network. The concepts of REST are pretty similar to the workings of the Web. With REST, the pieces of data that are transferred by the Web Service must be identified individually using a standard addressing scheme. A URI uniquely identifies any kind of network resource for which a unique address can be created.

REST leverages the successful aspects of existing Internet protocols. To cap off the REST model, the URI and an HTTP verb (GET, POST, PUT, DELETE) are used together to issue a request to a REST Web Service to access and manipulate the server-side state, or data record, identified by the URI. The actual XML message is contained in the HTTP request and security is provided by HTTPS, which is the secure version of HTTP.

The concepts of REST are further described below.

Representation

- Resources are first-class objects
- Resources are obtained as complete representations.

Resource

- A Web page is a representation of a resource
- Resources are described by URIs.
- Consumers can retrieve a specific representation of the resource.

State

- "State" represents the state of the application state as represented by a collection of resources
- The state is always maintained and traversed to and fro between the server and the client

Since REST Web Services ride the ubiquitous HTTP protocol and piggyback on the Web infrastructure, their presence is omnipotent and can be polymorphic in nature. This can be explained by the various number of REST-style services that currently exist on the Web. These include a client/server style, stateless and cached with a uniform interface, even code on-demand that is exemplified by Java applets.

You can locate resources by a Universal Resource Identifier (URI), and the operations that might be performed against those resources are defined by the HTTP specification. The core operations include:

- **GET** – This operation returns a state representation of the identified resource. You can determine the state by a number of contextual factors such as who is submitting the request, the parameters of the operation (passed in as either HTTP headers or query string parameters), and the current session state maintained by the service provider.
- **POST** – This operation performs some form of application-spe-

cific update to the identified resource. The behavior of this operation is completely up to the service implementing it. The data returned by this operation is also completely up to the application.

- **PUT** – This operation creates a new resource at an identified location (URI). The operation input must include a state representation of the resource. It is up to the service to create a resource based on that state representation.
- **DELETE** – The DELETE operation destroys a resource at the identified location (URI).

Testing Web Services

As we've seen, Web Services both REST as well as SOAP are going to be components that expose their functionality over HTTP. Hence we could potentially expose our services over an intranet or the Internet, which implies that the visibility of these Web Services is generally pretty high. There are, of course, some critical differences in the significance of intranet-based Web Services as opposed to the ones that will be exposed to the public domain. Intranet-based services lie within the boundaries of the enterprise and hence can be controlled and monitored with access restricted only to employees. However, an Internet-based service can be accessed by anybody and so we have far lesser control over those services. It's pretty much the same as the difference between an intranet and the Internet itself.

We'll look briefly at the various aspects of testing Web Services in general and then how it's done for REST-based services.

Functional Testing or Black Box Testing

This testing is carried out to ensure that the Web Service functions per the design and the requirements. These only deal with proposed inputs and expected outputs. Service test cases are generally mapped to business use cases that outline all the permutations and combinations of inputs and outputs.

REST-based Web Services are pretty unique since they basically correspond to the various operations outlined above. We suggest a couple of different possible scenarios to address the challenges involved.

The application to be tested could have an implementation of strict REST-based Web Services. We'll elaborate on this further with an example, say a product catalog in which items can be created, viewed, deleted, and updated. This is pretty similar to the CRUD operations that would happen for the items in the underlying database.

- When a new item is created it is achieved by the PUT operation and it in turn creates a new URI like `www.xyz.com/items/item-new01`
- When an item is to be viewed it is achieved by the GET operation by using the URI `www.xyz.com/items/showitem?itemid=I001`. This returns the URI `www.xyz.com/items/itemnew01`
- When an item has to be edited it is achieved by the POST operation by using the URI `www.xyz.com/items/itemid=I001`
- When an item has to be deleted it is achieved by the DELETE operation by using the URI `www.xyz.com/items/itemid=I001`

In such cases, while the former can be functionally tested by normal Web site testing mechanisms and we can use the normal Web testing tools that are available today.

Now if the application doesn't have strict REST-based Web Services, the Web Service responses could be slightly different. For example, it could return an XML over HTTP, instead of an URI representation of a resource. So when an item is to be viewed it could

return an XML instead of `www.xyz.com/items/itemnew01`. This particularly tends to happen to promote interoperability, so that non-browser-based clients can access these Web Services too.

This case is a bit more complex. Currently there aren't many tools available that can cater to this case.

Currently most testers build up their test cases and the input data in Excel sheets and verify it against the expected output XML. It probably makes sense to build a small tool that can do the following:

- Read the input parameters from the Excel sheet
- Invoke the appropriate service.
- Store the response provided by the service
- Compare the response against the expected response using tools like XMLDiff and provide reports

Non-Functional Requirements Testing

The other fundamental aspect of architecture and design is to ensure that the software meets all the non-functional aspects like security, reliability, availability, and scalability. These facets are generally the most ignored by software designers as well as testers. These prove costly mistakes because a functionally perfect system is of no use if it can't be used by a specified set of users in a predictable response time. There are also serious ramifications if the system isn't secure and allows unauthorized users access.

Hence non-functional testing is as significant as functional testing.

Security Testing

As previously mentioned, Web Services are software components that could be exposed over the Internet. In such scenarios security becomes a primary concern, Authentication and authorization concerns have to be tested thoroughly. In most scenarios for REST-based services, authentication is taken care of by using the HTTPS protocol. Most enterprises prefer to add an additional layer of authentication using either basic HTTP authentication or a complex token-based security mechanism. Basic authentication generally deals with access control lists. So testing should cover the aspects of "malicious testing," where the system is tested for controlled access by a specific and ratified set of users.

Another common security lapse in a REST Web Service is its vulnerability to SQL injection attacks. The similarity between the various HTTP requests to the CRUD database operations is indicated in the section above. It leaves a poorly designed Web Service highly vulnerable to SQL injection attacks. So the test strategy should contain test cases to cover this angle as well.

Amazon's private "shared-secret" token is a complex implementation of security that uses HMAC signing to show proof of possession of the public token. Although this kind of security implementation in REST Web Services diffuses the basic advantages of REST by reinventing the wheel, this security model exists. Testing these security models is a lot more complex and is more along the lines of testing a PKI infrastructure including the token issuing authority.

Authorization is generally designed using filters in combination with the resource control lists. Although this is no different from standard authorization testing mechanisms, it's imperative that the testing strategy includes tests for authorization. This ensures that a user can access only what he's supposed to access.

Another common area of testing is the audit trail of requests and responses. Since most REST Web Services are GET-based or based on information retrieval rather than altering the state of the resource on the server, it becomes very important to audit these requests. So the testing strategy should cover the audit trail mechanisms of these Web Services too.

“Businesses that ignore the potential of SOA will find themselves outpaced by rivals who improve their agility and transform themselves into new kinds of enterprises

— Yafim Natis, Gartner Analyst

2-DAY EVENT!

SOAWorld

Plus **2007**

Enterprise OpenSource

Conference & Expo 2007

TOPICS INCLUDE:

SOA Web Services

- > AJAX and SOA
- > Web 2.0
- > Universal SOA
- > Protecting Web Services
- > Troubleshooting SOA
- > Governance
- > Open-Source SOA
- > XBRL
- > Service Virtualization

Open Source

- > Open Source Business Models
- > Open Source ESB
- > OpenAjax Alliance
- > SaaS and Open Source
- > Spring, Hibernate and Eclipse
- > Seam
- > Open Source Penetration
- > Monetizing Open Source
- > Open Source Databases
- > AMQP
- > Open Source Middleware

June 25-27, 2007

Roosevelt Hotel / New York City

Register Online! www.SOAWorld2007.com

11th International
2007
SOAWorld™
CONFERENCE & EXPO

2007 is to many industry insiders shaping up to be a major inflection point in software development and deployment, with SOA, Web Services, Open Source, and AJAX all converging as cross-platform and cross-browser apps become the rule rather than the exception.

Accordingly the 11th International SOA Web Services Edge 2007 again seeks to offer comprehensive coverage and actionable insights to the developers, architects, IT managers, CXOs, analysts, VCs, and journalists who'll be assembling as delegates and VIP guests in The Roosevelt Hotel in downtown Manhattan, June 25-26, 2007

Co-located with the 2nd Annual Enterprise Open Source Conference & Expo, the event will deliver the #1 i-technology educational and networking opportunity of the year. These two conference programs between them will present a comprehensive view of all the development and management aspects of integrating a SOA strategy and an Open Source philosophy into your enterprise. Our organizing principle is that delegates will go away from the intense two-day program replete with why-to and how-to knowledge delivered first-hand by industry experts.

Visit soaeosconference.sys-con.com for the most up-to-the-minute information including... Keynotes, Sessions, Speakers, Sponsors, Exhibitors, Schedule, etc.

2nd Annual
ENTERPRISE > 2007
OPENSOURCE
CONFERENCE+EXPO

SOAEOSCONFERENCE.SYS-CON.COM

REGISTER ONLINE TODAY

SAVE \$200!

(HURRY FOR EARLY-BIRD DISCOUNT)

BROUGHT TO YOU BY:



» **SOA Web Services Journal**
focuses on the business and technology of Service-Oriented Architectures and Web Services. It targets enterprise application development and management, in all its aspects.



» **Enterprise Open Source Magazine**
EOS is the world's leading publication showcasing every aspect of profitable Open Source solutions in business and consumer contexts.



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

Exhibit and Sponsorship Info:

Call 201-802-3020 or email events@sys-con.com

Regression Testing

A regression test is normally a cut-down version of a functional test. Its aim is to ensure that the Web Service is still working between builds or releases. It assumes that some area of functionality worked in the past and its job is to check that it still does. However, versioning is difficult to achieve transparently in REST-based Web Services since each resource has its own representation. So if a resource changed in different builds or releases, it would have a new URI. However, less strict versions of REST Web Services allow a pure XML over HTTP and POST/PUT to be used interchangeably. They also have a version field either as part of the payload or query string. This lets them introduce versioning from a deployment perspective so different versions of the same service can co-exist while using the same URI. Although this violates basic principles of Roy Fielding's theory, unfortunately there are a lot many implementations out there. Hence regression tests need to account for versioning test cases when multiple versions of the same service will co-exist in the application ecosystem.

Load Testing

The aim of load and stress testing is to find out how well your Web Service scales as the number of clients accessing it increases. The load testing is generally done to find the acceptable benchmarks associated with the Web Service and identify the Service Level Agreements (SLAs) that can be specified with this service. These benchmarks then become the SLAs of the service. Load tests for REST-based Web Services are primarily based on the same concept as load testing Web applications or Web sites. However there are subtle differences that have to be looked at carefully. REST Web Services are XML-based services. So while you get the flexibility of XML, you also get the baggage that comes with it. XML is bulky and takes up more bandwidth. Large XML messages tend to take up lots of memory.

These XML messages also have to be parsed and using DOM parsers for large XML documents often leads to low memory situations. So it's imperative that the load testing strategy includes testing the performance of the services under an extreme load, particularly large XML payloads.

Conclusion

We have articulated the basic architectural Web Service paradigms, namely SOAP-style Web Services and REST-style Web Services. We looked at various aspects of testing REST-based Web Services. It helps to develop a testing strategy or methodology for REST-based services. We also mooted the idea of designing a tool based on available out-of-the-box components to test the services. We concluded that testing REST-based services is reasonably easier than SOAP-based services, where additional testing of standards, interoperability, and versioning is required. Testing is also required to see if the SOAP-based Web Services adhere to WS-I standards. ■

References

- <http://www.codeproject.com/Webservices/WebserVICetesting.asp>
- Roy Thomas Fielding. "Architectural Styles and the Design of Network-based Software Architectures." Doctoral dissertation. University of California at Irvine. 2000.
<http://roy.gbiv.com/pubs/dissertation/top.htm>

About the Author

The authors are interning and/or working as part of the Web Services COE (Center of Excellence) for Infosys Technologies, a global IT consulting firm, and have 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.
niranjan_v@infosys.com

SOA Teams — continued from page 4

These people need to understand both logical process designs, as well as how to deal with processes yet to be automated, workflow and implementation, or the solution to the process problem.

Governance specialists are just that. They figure out the role governance plays within a SOA, the right technology for the job, and how to implement it in the course of the project. In some instances the use of governance is contraindicated, so you have to be careful here.

Testing and deployment specialists are the people responsible for the development of a formal test plan for the SOA, and they test each layer/component to make sure that it's rock solid and ready for production.

The project archivist is responsible for keeping track of the various design artifacts that pop out of these projects, including business requirements, application semantic documentation (meta-data), services analysis and design documentation, process analysis and design documentation, test planning, etc. This makes it easy for others on SOA projects in the future to learn from the successes and mistakes of others.

External services specialists are people who look outside of the firewall to meet the services needs of the SOA. This means looking at SaaS providers, and other services you don't own, as potential solutions/components within the SOA.


How Many?

Your mileage may vary – a lot. My numbers are for a typical project, but here are some initial findings. My assumptions are: A dozen systems in the problem domain, each having separate data layers

that are physically collocated. In addition, I'm assuming medium complexity for the SOA, a reasonable budget, and the availability of training and outside consultants.

- **Project leader/architect:** Typically one for the project.
- **Data specialist:** .5 per data layer. Meaning, if you have 12 different databases or applications, you need six.
- **Security specialist:** Two per project – one who understands the existing security, and one who understands the special security requirements of SOA.
- **Native systems specialist:** One for each type of system, i.e., if you have a mainframe, Unix, and Windows NT, you'll need at least three.
- **Service development specialist:** One for every 100 services to be deployed. Typically you're going to do approximately 1,000 in a project that big, thus figure on 10 service development specialists.
- **BPM/orchestration specialist:** Four per project – one who understands and documents existing services, one to document new services, and two to build the services into the orchestration layer.
- **Governance specialist:** One per project.
- **Testing and deployment specialist:** Three per project – one to write the plan, and two to execute the plan.
- **Project archivist:** One per project.
- **External services specialist:** One per project.

Hopefully these estimates will prove helpful as you develop staffing plans and job descriptions. As new data points and experiences become available, I'll let you know. ■



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

AJAX ONE-DAY HANDS-ON INTENSE TRAINING!

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
March 19, 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
 - Overview 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
- Create 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

The Software-as-a-Service Model

Changing
the way
compliance
gets done



WRITTEN BY ROGER BOTTUM

➤ Fast, easy, consistent — today we expect services to be delivered rapidly, with ease, and the utmost precision. We expect instant access and on-demand service from our cable providers and online retailers so why don't the same principles apply to areas of real business needs such as compliance software? Now they do. The Software-as-a-Service (SaaS) model has begun to take an aggressive hold, offering a wide range of applications to a variety of industries. In many instances, SaaS has become the de facto choice for businesses looking to upgrade their key business solutions. Nowhere is this shift towards SaaS more evident than in the government risk and compliance (GRC) industry.

GRC applications impact all organizations, with prominent mandates ranging from enterprise risk management, Sarbanes-Oxley, and HIPAA to corporate codes of conduct, anti-money laundering statutes, operational risk, and COBIT. As regulations and mandates continue to grow, organizations must be able to document their compliance, whether for auditors or simply to demonstrate best practices to partners, vendors, and prospective clients.

Until recently, on-premise software, or traditional software, was the compliance model of choice, as it gave companies the ability to automate the compliance process. Once in place, these solutions alleviated much of the stress associated with the repetitious and detail-oriented tasks and subsequently allowed mandates to be addressed more easily than at any time previous.

However, in recent times, with the compliance landscape intensifying, the limitations of this approach began to show. Specifically, on-site solutions require manual installations of software directly to individual computers, making the rollout process inefficient and unable to address immediate areas of concern. In the case of product upgrades, the limitations grew even more evident, with the process requiring significant on-site man-hours that translated into an exorbitant expenditure of time and money.

Though on-premise software applications can be used in conjunction with the Internet, it was not until recently that programmers developed a new approach to software implementation, one harnessing the delivery capabilities of the World Wide Web. This new approach to software delivery is known as "Software-as-a-Service" and though SaaS has been around in principle for several years, it wasn't until recently that it began to gain momentum. In fact, a 2005 IDC Research study forecasted that worldwide spending on SaaS will reach \$10.7 billion by 2009.

The benefits of SaaS solutions are far-reaching and include:

- **Security** – SaaS applications have the advantage of being deployed on an external server. Unlike traditional software installs that rely on internal security to protect sensitive materials, SaaS applications are securely based in server farms where they are monitored and protected 24 hours a day. These servers automati-

- cally back up materials and applications, ensuring that even in the worst situation, information won't be compromised. Best of all, this entire process is blind to the customer, who no longer needs to assign an internal person to ensure operational security.
- **Ease and Speed of Installation** – One of the major roadblocks in a traditional software model is the time required to install and customize the solution. Depending on the size of the company, the installation process can take weeks and turn into months. Conversely, SaaS applications offer the shortest deployment time, since there's a minimal installation process. Though the program is installed on an offsite server, SaaS solutions still offer the customization capabilities that companies look for when adding a new product to their current solution set.
 - **Timeliness and Versatility** – In many industries, frequent and ongoing product upgrades are essential for companies looking to keep pace and ensure that they have the “latest” tools in place. Under the SaaS model, all updates are installed on the main server and users simply access the software through an Internet portal. A typical upgrade can be done in just minutes, a significant improvement over an on-premise software solution, which can take anywhere from 18 months to two years for a total Global 2000 company-wide upgrade to be completed.
 - **Bottom-Line Impact** – SaaS applications are strong performers, financially speaking. In general, the automatic upgrades delivered by SaaS programs result in a lower total cost of ownership (TCO) due to the fact that SaaS eliminates much of the internal IT effort dedicated to solution upgrades that can cost upwards of \$200,000 a year. In fact, compared to traditional on-premise installations, SaaS applications have an overall 20%-40% lower TCO, allowing companies to reallocate resources to other areas. Rapid installs also translate into a quicker return on investment, something all companies are interested in achieving.
 - **Subscription Benefit** – Since SaaS is subscription-based, vendors have a greater focus on quality of service (QoS) and support, rather than the single software upload. This approach forces them to stay on top of quality, maintenance, and service. The format also opens the door for customers to deliver direct feedback to the vendor, which results in continual improvement in the system that directly benefit the end user.

So now that you see the many of the benefits of SaaS, you must be asking, “Why are SaaS and GRC such a good fit?”:

- **Evolving best practices** – With the ever-changing particulars surrounding regulations, compliance software must evolve in-step and in real-time to meet new government-mandated changes and keep companies one step ahead. With the SaaS model, GRC practitioners aren't concerned with whether or not their current software includes the most recent updates, since these updates are continually and automatically installed as needed.
- **Reliability** – Having an outsourced program increases security and reliability because the solution is supported by a full-time staff devoted to monitoring and repairing an application. SaaS applications ensure that no matter what happens to a home server, the application itself will won't suffer, since it's housed in locations constantly monitored and backed up. When working with sensitive information associated with compliance mandates, a reliable, secure system is vital to ensure data is cared for appropriately. Additionally, in a world filled with last-minute deadlines, a reliable system ensures the ability to complete GRC tasks in a timely fashion.

- **Dramatic Scalability** – While some compliance programs may only touch 50 or 200 users to start, ultimately GRC touches everyone in a company. Using SaaS, it's easy to increase the access and usage across all company offices in a matter of minutes.
- **Cost** – SaaS applications are designed to facilitate multiple users working simultaneously on the same application, regardless of the user's location — a critical capability for a growing Global 2000 organization. This feature eliminates the duplication of efforts and downtime, increasing overall productivity.

	SaaS	On-Premise
Software Update	Included in the Subscription	Maintenance
Hardware and Software	Included in the Subscription	Duplicate hardware Duplicate supporting software (OS, database, Web servers)
IT Resources	Included in the Subscription	Hardware and software installation and configuration Application installation and configuration Copy data from an existing production environment to a test environment Apply software updates to a test environment Install a version to a production version

Table: SaaS versus an On-Premise Snap Shot

Conclusion

The capabilities of the Software-as-a-Service model are just being realized and the GRC industry is leading the charge. According to AMR research, nearly 40% of companies using software for the management of a GRC application prefer some form of the SaaS model. The fact is that the SaaS model fills the holes left by traditional software installations that, simply put, aren't capable of managing these timely requirements. Conversely, SaaS delivers a faster ROI and lower TCO with less risk, making it the ideal partner for today's GRC program. ■

References

- Erin Traudt and Amy Konary. “2005 Software as a Service Taxonomy and Research Guide.” IDC Research Report. June 2005.
- John Haggerty and Fenella Sirkisoon. “Spending in the Age of Compliance, 2006.” AMR Research report. Spring 2006.

About the Author

Roger Bottum is vice president of marketing and product management at Axentis.

Equal Web Site Access

How XML-Driven Content Can Provide Better Accessible Web Content for Persons with Disabilities

WRITTEN BY DAVID CUMMINGS

➤ In the CMS industry, we try to stay current with the latest and greatest trends for Web sites, Web apps, and the Internet in general. We work so hard to stay ahead of the market, in fact, that it's often easy to lose sight of where most businesses and organizations are in terms of their online presence.

To better understand our customer base, we at Hannon Hill undertook a primary research project in the summer of 2006 to determine how important Web standards are in the education and healthcare fields. What we found was not good: 86% of America's Best Colleges, and 99% of America's Best Hospitals, as ranked by U.S. News & World Report, aren't compliant with the HTML/XHTML Web standards recommended by the W3C.

New Legal Precedents for Web Accessibility Catalyst Change

Meanwhile, the legal push for Web accessibility (directly related to W3C HTML/XHTML Web standards) is increasing, even into the corporate sector. In September 2006, a legal precedent was set for Web accessibility when a federal judge sustained discrimination claims against retailing giant Target. This precedent means that retailers must make their e-commerce sites accessible to the blind under the Americans with Disabilities Act (ADA).

The problem isn't that companies don't want to provide disabled people equal access to their Web sites, but rather that many of them don't have a clear understanding of the inhibiting elements or how to resolve their sites' accessibility problems.

Web 2.0: New Technology, New Challenges with Accessible Web Content

Enter Web 2.0, and the problems can only get worse. New content formats and content management systems can make providing content that is accessible to a wide number of users with dis-

abilities, from limited motor functions to blindness, even more complex. For example, with an increase in visually oriented rich media comes an increased difficulty for Web browsers that translate content for blind users.

In preparation for the onset of Web 2.0, the WAI has already appointed an advisory committee to revise the Section 255 guidelines and Section 508 standards to address some of the issues that are beginning to surface. Since the new guidelines will end up being an extension of the current ones, anyone who's already developing and managing compliant sites will have a jump start on the rest. And that's where XML comes in.

XML Drives Accessible Web Content

To understand how these guidelines aid accessibility – and how XML can help – it's important to understand how individuals with disabilities might interact with the Web. When approaching a Web site design, ask yourself, "How do the blind or persons with color defected vision or limited motor skills navigate the Web?" And, to a large degree, improving a Web site's accessibility isn't geared directly toward assisting the disabled visitor, but toward helping the various available assistive technologies read the site better. For companies serious about improving their accessibility, it might make sense to invest in a Jaws screen reader, for example. To simulate limited motor skills just take away your mouse. Try navigating with only keystrokes, and you might be amazed at the challenges your own site presents. By designing your Web site so that assistive technologies can navigate it easily, you've made your site accessible.

Easier Content Management & Better Search Rankings: Accessibility-Compliant Web Sites Provide Additional Benefits

Assistive technologies aren't the only "machines" reading your Web site. By making your site standards-compliant, you'll discover other benefits. Search engines can index your site more accurately, improving search rankings as well as the user experience. Regardless of what precisely the advisory committee comes up with

Welcome to the Future of Video on the Web!

REGISTER NOW!
www.iTVcon.com

Early Bird: Save \$100!

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

iTVCON.COM
INTERNET TV CONFERENCE & EXPO 2007

June 25-26, 2007

The Roosevelt Hotel • New York City

“Internet TV is wide open, it is global, and in true ‘Web 2.0’ spirit it is a direct-to-consumer opportunity!”



For Sponsorship and Exhibitor Information:

www.iTVcon.com / 201-802-3020 / itvcon@sys-con.com

Welcome to the Future!

Did you already purchase your “.tv” domain name?

You can't afford not to add Internet TV to your Website in 2007!

Just as 2006 was the year of streaming video and the birth of “Internet TV,” 2007 is going to be the move beyond infancy toward the long-awaited and complete convergence of television and the Internet. Now that broadband is available to more than 100 million households worldwide, every corporate Website and every media company will now provide video content, not to mention live and interactive video Webinars and on-demand Webcasts, in order to remain competitive.

Internet TV is wide open, it is global, and in true “Web 2.0” spirit it is a direct-to-consumer opportunity. 20 years ago the advent of desktop publishing tools opened the doors for the creation of some of today's well-known traditional print media companies as well as revolutionized corporate print communications. Today, with maturing digital video production, the advent of fully-featured PVRs, and significant advances in streaming video technologies, Internet TV is here to stay and grow and will be a critical part of every Website and every business in the years to come.

It will also very rapidly become a huge challenge to network and cable television stations: Internet TV is about to change forever the \$300BN television industry, too.

The Internet killed most of print media (even though many publishers don't realize it yet), Google killed traditional advertising models, and Internet TV will revolutionize television the way we watch it today.

To be part of this change, register for the first Internet TV Conference & Expo 2007 today!



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

List of Topics:

- > Advertising Models for Video-on-demand (VOD)
- > Internet TV Commercials
- > Mastering Adobe Flash Video
- > How to Harness Open Media Formats (DVB, etc)
- > Multicasting
- > Extending Internet TV to Windows CE-based Devices
- > Live Polling During Webcasts
- > Video Press Releases
- > Pay-Per-View
- > Screencasting
- > Video Search & Search Optimization
- > Syndication of Video Assets
- > V-Blogs & Videoblogging
- > Choosing Your PVR
- > Product Placement in Video Content
- > UK Perspective: BBC's “Dirac Project”
- > Case Study: SuperSun, Hong Kong

Track 1	Corporate marketing, advertising, product and brand managers
Track 2	Software programmers, developers, Website owners and operators
Track 3	Advertising agencies, advertisers and video content producers
Track 4	Print and online content providers, representatives from traditional media companies, print and online magazine and newspaper publishers, network and cable television business managers

in the future, it would do companies well to pay attention. When companies improve their Web sites to meet W3C guidelines, they take the same approach to making a Web site accessible as they would to making physical walkways and structures accessible to persons with disabilities.

Besides all the obvious benefits, the thing that makes me really excited about using XML to create standards-based accessible sites is how much easier it makes managing a site. XML just makes things cleaner. It'll also force you to be more organized, which might make some people grumpy to start with, but ends up saving a lot of time.

'How-To' Use and Manage XML Content for Company Web Sites

Let's look at specific ways in which companies can use XML – both now and in the future – to make the Web more accessible.

XML, as a standard format, makes it easy to enforce certain rules. Take for instance this code snippet of an image:

```

```

What's wrong with it? Well, there's no text indication for a blind person. What is picture.jpg? Is it a picture of a kitten, a person, an object, etc? Assuming it's in fact a picture of a kitten, the proper XML tag would be:

```

```

A screen reader would then be able to relay information to a blind user about the image in context.

Another example of XML facilitating accessibility issues is that of different format outputs. Once the content is in XML it's easy to transform the XML into different output types like XHTML Mobile for cell phones, PDF for printing, and text-only for people with certain kinds of disabilities. A text-only version can be optimized for different screen readers.

Let's look at a more technical example. Most Web sites have a code structure where the page header comes first followed by the top navigation and then finally the meat of the page. Here's an example:

```
<html>
<head>
  <title>Page Title</title>
</head>
<body>
  <!-- Header HTML Goes Here -->
  <!-- Top Navigation HTML Goes Here -->
  <!-- Main Page HTML Goes Here -->
  <!-- Footer HTML Goes Here -->
</body>
</html>
```

The common accessibility technique is to have a hidden link that "jumps" to the main page content like follows:

```
<p class="hidden"><a href="#main">Jump to content</a></p>
```

This works fine but someone using a screen reader would have to choose that link on each page as they browse around. This becomes cumbersome and makes their experience less than ideal. With your site content stored as XML you can easily create a complete version

of your site that's text-only whereby people with disabilities will be greeted with the main content first. Let's look at that example:

```
<html>
<head>
  <title>Page Title</title>
</head>
<body>
  <!-- Main Page HTML Goes Here -->
  <!-- Footer HTML Goes Here -->
</body>
</html>
```

Once the XML infrastructure and re-purposing rules are in place, there's no additional burden put on the content authors to have this text-only version of the site made available for whosoever chooses it.

If you're still not getting excited about using XML to drive your Web site, check out a few organizations that have taken the plunge. Since our study was conducted, we've noticed more universities catching on. Just recently Carnegie Mellon University launched its new XML-driven Web site, www.cmu.edu. Without making any sacrifices, it now has an easy-to-manage site that's accessible and W3C-compliant (to see their compliant site results, visit <http://validator.w3.org/check?uri=http%3A%2F%2Fcmu.edu>).

You can also check to see if your own site is compliant, and what factors need to be updated by visiting the following link: <http://validator.w3.org/check?uri=http%3A%2F%2Fwww.w3.org%2F>.

Two of the greatest benefits of the Internet are its availability and reach. Web sites allow potential students to visit campuses on the other side of the country, and shoppers to buy at stores from the comfort of their home. Virtual experiences are increasingly becoming as important as face-to-face interactions, and equal Web site access is increasingly viewed as a right, not a privilege. Fortunately, with XML to help technology meet the demand, we can follow a few simple concepts and not only meet accessibility requirements now, but also to be prepared for future developments. ■

Additional Resources

- Hannon Hill white paper: "Maintaining an Accessible Site." www.hannonhill.com/resources/white-papers/accessibility.html
- Web Accessibility Initiative (WAI). www.w3.org/WAI/
- American Association of People with Disabilities (AAPD). www.aapd-dc.org
- "Dive Into Accessibility - 30 Days to a More Accessible Web Site." www.diveintoaccessibility.org/
- Rehabilitation Engineering & Assistive Technology Society of North America (RESNA). <http://www.resna.org/>

About the Author

David Cummings is founder, CEO, and president of Hannon Hill. A business and technology enthusiast, he founded Hannon Hill in 2001 to provide powerful XML Web content management (WCM) solutions, with an emphasis on personalized customer service. A WCM subject matter expert, his articles have been published on O'Reilly's XML.com and OnJava.com, besides popular Web development sites such as SitePoint.com. He has presented at the Silicon Valley Web Guild, for Silicon Networks, and the Independent Computer Consultants Association (ICCA). With more than 10 years of software development experience, David has written code for commercial applications in Java, PHP, ASP, and Visual C++. Before starting Hannon Hill, David worked as an engineer for IBM. David serves as chairman of the Technology Association of Georgia Content Management Society. A native of Tallahassee, Florida, He has a BS in economics from Duke University. For more information about Hannon Hill, please visit www.hannonhill.com. david.cummings@hannonhill.com

JavaOne™

Sun's 2007 Worldwide Java Developer Conference™

May 8-11, 2007

The Moscone Center
San Francisco, CA

JavaOne™ Pavilion: May 8-10, 2007



IT'S AN EXCITING TIME FOR THE ENTIRE JAVA™ TECHNOLOGY COMMUNITY.

With the evolution of the Java platform, new opportunities are emerging for developers, thought leaders, and entrepreneurs. That's why for 2007 the Conference is featuring an expanded program that embraces technologies outside the core Java platform while keeping Java™ technology at the center. You'll get the same in-depth content we have always focused on, plus more topics and issues relevant to today's evolving market.

LEARN MORE ABOUT:*

- > Java Technology
- > Scripting
- > Open Source and Community Development
- > Integration and Services-Oriented Development
- > Web 2.0
- > Compatibility and Interoperability
- > Business Management
- > And More



SAVE \$200**
REGISTER BY
APRIL 4, 2007

Please use priority code: J7PA1SOA

*Content subject to change.
**Offer not available on site.

Register Today @ java.sun.com/javaone





Leavening
SOA with MDA

Re-engineering A Legacy Enterprise IT System

WRITTEN BY CHUNG-YEUNG PANG

➤ For the past six years I've been engaged in a project to re-engineer a legacy enterprise IT system of a large international bank. The company still had applications with host terminal emulation that were to be replaced with Windows- and Web-based client applications. The host system was completed based on legacy applications that evolved since the 1970s. Any new applications had to interoperate with the existing legacy applications. A strategy to replace difficult-to-maintain legacy applications with new applications was implemented.

Within these constraints, Service Oriented Architecture (SOA) was introduced into the existing system. The dependency on the interfaces, performance, and transaction scopes of the legacy system prohibited any change to, or mix-

ing with, other programming languages. All host applications were still developed in COBOL, while Windows-based client applications were developed in Java or Web-based languages. A framework with many COBOL modules was built to support the development of applications that adhere to SOA. It took three years to integrate the framework and architecture into the existing IT system and fully interoperate with existing applications.

Despite the new architecture and framework, applications were developed in the same ad hoc style as in the 1970s and 1980s. The result was undocumented and error-prone systems. Applications were inflexible for code adaptation to standards. Reusability was low. Development relied heavily on experienced programmers.

To overcome the problems common to any software development, the establishment of component architecture with "plug and play" features for individual modules made sense. A componentized development approach allows for independent module testing. A full set of modules was written to support the architecture.

With the introduction of SOA, application development was greatly simplified. However, using the framework was still a challenge for traditional programmers. To combat this, three years ago we introduced and refined techniques based on Model Driven Architecture (MDA) and have used these techniques in the develop-

ment process. Since then, the techniques have been successfully applied to six IT projects for a variety of applications in treasury products, payments, credits, and securities. Following the principles of MDA, all systems are fully documented with models. As a consequence of this change in the process architects and analysts could drive development. Productivity has increased by a factor of three. Programming errors are minimized. Applications are flexible for code adaptation to standards. Reusability is very high. Through the use of models and code generation from the models, development is automated, effectively reducing the number of experienced programmers needed. The concept and design of the architecture, framework, and MDA techniques are presented below.

The Architecture

The system topology of the SOA is quite simple. It's illustrated in Figure 1. Client applications communicate with host applications via HTTP or Message Queue. XML is used for message exchange between client and host applications.

The software architecture for the host applications is illustrated in Figure 2. The interface is exposed using XML to transfer across environment divides. A security controller controls access to the services and data. The service mediator acts as the interface and exposes the service to other applications. It also controls the transaction, retrieves the XML message, parses the XML, sets up the context container, resolves and instantiates the requested service, and returns the XML message.

The service mediator calls the service controller, which in turn calls the components making up the application. Components are completely decoupled: they communicate only via the context container. Each component reads its input data from the context container and writes its output data into the context container. Thus each component can be tested separately as long as the data required for a component is inserted into the context container.

The Framework

Only the components are application-specific. Other elements in the architecture are all part of the pre-built framework. Within this framework, there is a complete set of facilities to handle XML in COBOL. This includes, for example, functions to parse XML, bi-directional mapping of XML structure into COBOL structure, as well as navigation and retrieval of data based on XPath, add nodes to an XML tree, and retrieve node value from an XML tree, etc.

The service mediator is a generic module that can handle different services from the service request information in the incoming

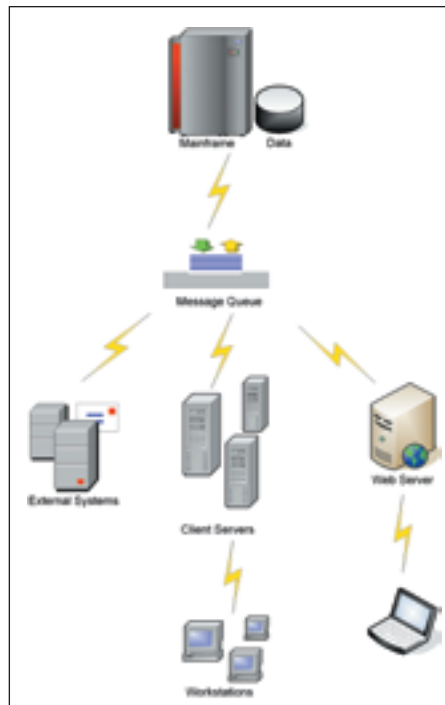


Figure 1: Simple system topology for Service Oriented Architecture

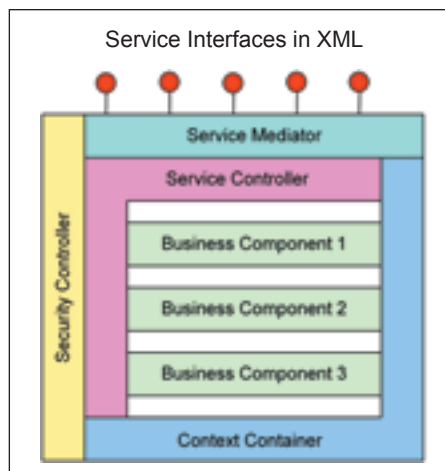


Figure 2: Component and context-based Service Oriented Architecture schema

XML message. The context container provides a set of APIs for application components to extract and deposit data. The data in the context container has a tree structure and a complete data structure can be added to a node or retrieved from a node.

The service controller is a finite state machine COBOL module. It performs the centralized controlling of the process in the business service based on the process description of state, event, and transition.

Meta-Information Generation via Models

For the bi-directional mapping of the XML and COBOL structure, we need meta-information. This meta-information is generated from UML class diagrams. Figure 3 shows an example of such a class diagram, which is used to generate a COBOL module containing the meta-information as well as a COBOL copybook for the data structure.

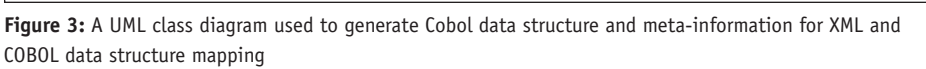
The service controller also requires a description of the process flow to provide state event transition information. This is generated from an UML activity diagram as shown in Figure 4. All program code for both of these cases is generated. The control flow description specifies which condition calls which module. As shown in Figure 4, the first state is an "Init" state. The module DEMOPF1 is called, which resolves the input. The outcome of calling this module returns an event variable in the service context, meaning either the "Customer is Company" or "Customer is Private." Depending on this event, the process flow controller would invoke the subsequent module.

To add a module, one only needs to modify the activity diagram by adding a new action to the diagram and generate from the model once more. To remove a module is just as simple. Hence a plug-and-play mechanism is provided. The only condition is that each module must adhere to the programming pattern using the context container for data input and output and the

module must update the event in the service context in the context container.

Application Module Development Using Models

Using XML services and the API from the context controller isn't necessarily easy. To simplify their use a full set of code patterns for different specific purposes is provided. These code patterns are parameterized and can be used as provided by passing the proper parameters. The discovery that code patterns covering many different purposes further simplified development. Code patterns can be used for business rules, accessing system functions, database interactions, etc. The patterns are reusable elements for general and specific purposes.



```
graph TD
    Init((Init)) --> ResolveInput[«Action Module»  
Resolve Input  
Module Name  
DEMOPF01]
    ResolveInput -- "Customer is Company" --> ProcessCorporate[«Action Module»  
Process Corporate Customer Data  
Module Name  
DEMOPF02]
    ResolveInput -- "Customer is Private" --> ProcessPrivate[«Action Module»  
Process Private Customer Data  
Module Name  
DEMOPF03]
    ProcessCorporate -- "Is OK" --> GenerateOutput[«Action Module»  
Generate Output  
Module Name  
DEMOPF04]
    ProcessPrivate -- "Is OK" --> GenerateOutput
    GenerateOutput -- "Is OK" --> End(( ))
```

The diagram illustrates the mapping of a C program to a control flow graph (CFG) and then to a control flow graph with labels (CFG-L). The top part shows the C code with labels like 'main', 'loop', 'break', and 'return'. The middle part shows the CFG with nodes and edges. The bottom part shows the CFG-L with labels and edges. A large blue arrow points from the C code to the CFG-L.

The COBOL data structure can be defined in a UML class model. Packages can be used to define the different sections of the structure as required in COBOL, such as the working storage and linkage section. The logic of a COBOL module can be modelled in form of a UML activity diagram. Each action of the activity can contain a COBOL code segment. Alternatively, it can be bound to a code pattern with specific pattern parameters. A complete COBOL module can be generated from such an activity diagram together with the class model. Standard templates are used to ensure that each module adheres to the programming pattern required for the framework, code standard, and platform. Actions are bound to the code patterns with values for the parameters via links as shown in Figure 5.

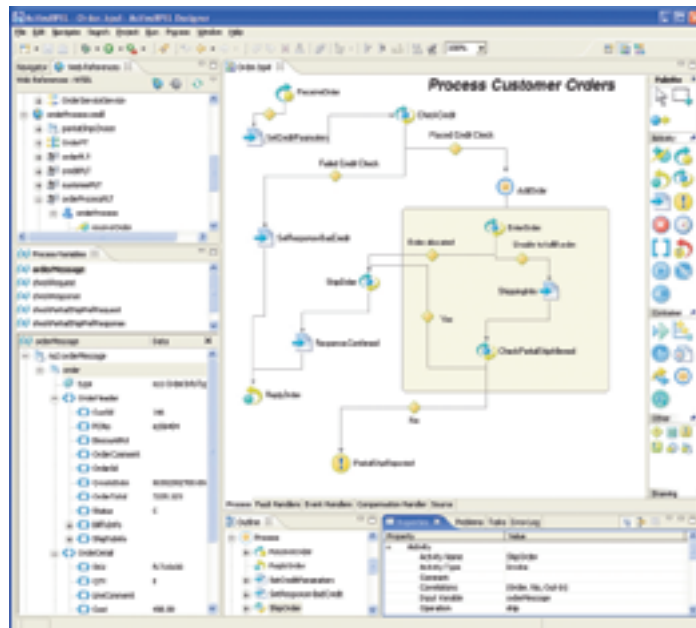
Conclusion

About the Author

Chung-Yeung Pang works as an IT consultant for international banks in Switzerland. He has been the main architect, a mentor to software development and modeling, and a project leader in different large IT projects. He received his PhD from the Cambridge University, UK.

34 February 2007

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**

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