SYS-CON

THE WORLD'S LEADING MAGAZINE DEDICATED TO WEB SERVICES TECHNOLOGIES

Visit us at www.WebServices.SYS-CON.com

# SOA WebServices JOURNAL

AUGUST 2006 / VOLUME: 6 ISSUE 8

## Creating Practical Portable Portlets

Extending Identity
**Management Solutions**

Oracle Business
**Activity Monitor**

Business Value
**Agile IT Architectures**

AJAXWORLD CONFERENCE & EXPO
October 2-4, 2006
Santa Clara Convention Center
Hyatt Regency Silicon Valley
Santa Clara, CA
EARLY-BIRD REGISTRATION! SEE PAGE 40 FOR DETAILS

*Altova® UModel® 2006 – The starting point for successful software development.*

ALTOVA® umodel® 2006

# Visualize works of software art

**Draw on UModel® 2006, and picture better programs based on UML™.**

**Spied in UModel 2006 Release 2:**

- Activity diagrams
- State machine diagrams
- Component structure diagrams
- Context-specific toolbars

Altova UModel 2006, the burgeoning new force in the software design space, is the simple, cost-effective way to draw on UML. Use it to interpret or create your software architecture. Decode Java or C# programs into clear, accurate UML2 diagrams, or outline applications and generate code from your plans. With all major diagram types, interoperability via XMI 2.1, and an artful user interface, UModel makes visual software design practical for programmers and project managers everywhere. Take the mystery out of UML! **Download UModel® 2006 today: www.altova.com**

ALTOVA®

SOA Web Services JOURNAL

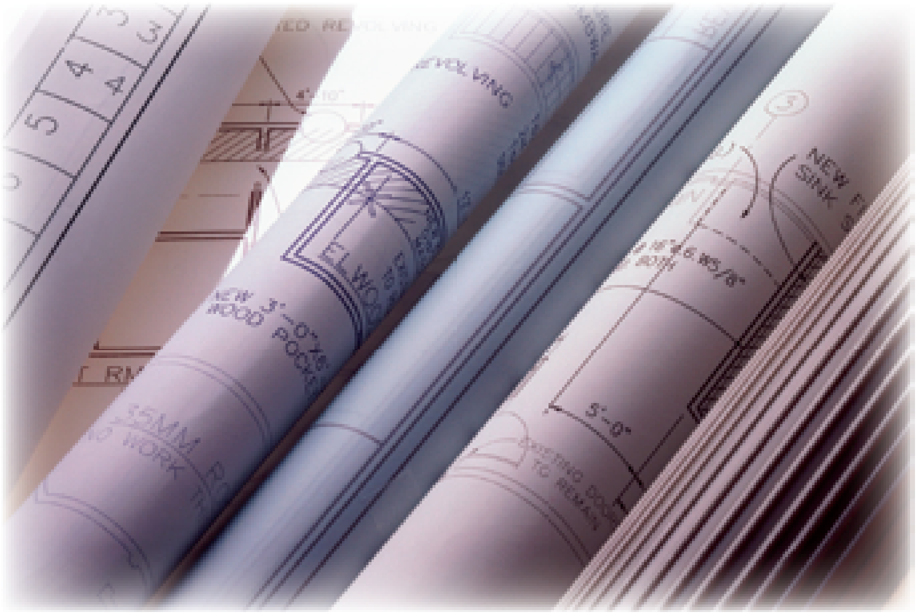
AUGUST 2006 / VOLUME: 6 ISSUE 8

# TABLE OF CONTENTS

Visit us online at WebServices.SYS-CON.com

## Inside This Issue

# BPEL is the SQL of SOA



**Get started building next-generation SOA applications with the leading vendor of BPEL technologies**

**Download BPEL tooling & server software today**

**activebpel.org/soa**

*active*BPEL

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

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.
477 Chestnut Ridge Road, Woodcliff Lake, NJ 07677

# The SOA Dichotomy

WRITTEN BY **SEAN RHODY**

As editor, I review a great many proposals for articles. A good portion of them deal with SOA, which is to be expected. When I review them, I'm reminded that there are two very different views of SOA, which in my opinion are both equally true. I call this the SOA Dichotomy, because these views seemingly contradict one another.

One of the views is that SOA makes things easier for the enterprise. Certainly this view has a great deal of merit and validity. Fully realized a service-oriented architecture allows an organization to fully leverage their investment in the real intellectual property of software – the service that it provides – without closely coupling that service to a single application. It's hard to overstate the impact this can make; it's truly a transformation of the way IT provides information services to the enterprise. Applications can now be assembled, and business processes composed from services and altered rapidly in response to changing business conditions. The workplace can be transformed – workers no longer need to be dependent upon a series of applications to do their work. Instead they can have a single composite application that meets all of their needs without ever having to leave to transfer to another application. Anyone who's ever been on hold with a call-center representative while they said, "Can you hold for a second, that's in a different application," knows how valuable this can be. Call centers know it as well – they know to the penny just how much a single second's delay costs them. Even before an enterprise completes its transformation to a fully deployed SOA environment, the benefits of interoperability and increased agility grow dramatically with every application that is decomposed into services in an effect similar to the very familiar Network Effect. The more services deployed, the more valuable they become. Usually, this is attributed to the fact that SOA makes things simpler, easier to do.

But the other view is the familiar devil-in-the-details argument. While not exactly contrarian, there is also a group who insist that SOA introduces increased complexity and requires greater attention to details. It's hard to argue with this when you try to create composite services and need to introduce concepts like security and transactionality to organizations that never consciously had to consider them, because they were always embedded in the application and were dictated by whatever the application vendor or in-house developers decided was the best approach. SOA requires an increased awareness on the part of the IT organization, and greater responsibility from the business side with respect to understanding the full impact of their decisions. Not that that's a bad thing, because this increased self-awareness helps a business to understand itself and adapt to changing conditions. SOA has value, but it's not as easy as it sounds at first.

How do we resolve these different views on SOA, or do we even need to? This is where the dichotomy comes into play – both views are correct. What's more, not only are they both correct, but they are both necessary. The key understanding is that SOA is a business paradigm shift, not a technology one. The true goal of an SOA is to make it easier to affect business changes and make business decisions. From that perspective the ability to work as a service distinct from an application provides strategic value. The ability to create composite services and manage them from a business perspective is a competitive advantage in today's marketplace. In tomorrow's, it will be a requirement.

That doesn't mean that a business shift makes things easier on IT. Nor does SOA. IT knows full well that while a single vendor system often has its issues, dealing with a federated heterogeneous computing environment often is more least-common denominator than best of breed. With multiple standards bodies and numerous standards and versions, the SOA landscape is cluttered and complex, and requires skilled practitioners to successfully navigate the muddy waters of a service environment. They also know it's a better place to be than the one they are leaving, which had the same problems, but no solutions, even if the solution in this case increases the complexity of their jobs.

And that's the dichotomy of SOA. It is both easier and harder, more complex and simpler. It's all a matter of perspective. ■

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

# How Much SOA?

## Where you end up depends on where you start

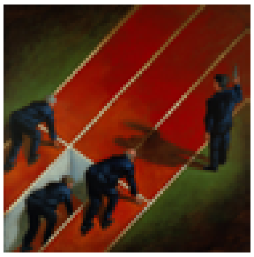
WRITTEN BY **AJIT SAGAR**

Companies that decide to invest in SOA sometimes end up going to extremes – too little or too much. Too little happens when some stakeholder latches onto the buzzword and wants to get the benefits promised. However, the environment may be too conservative to invest in the infrastructure and planning required to service-orient existing applications. In this case, an analysis concludes that business as usual is doing just fine, and that there's no need to introduce fancy technologies and platforms. A few minor tweaks to the existing infrastructure are considered sufficient to get on the SOA bandwagon.

In the other case, stakeholders buy into the entire vision of SOA. The IT department is looking desperately for a revamp to continue to survive and prove value, and the company as a whole decides to service-orient lock, stock, and barrel. Major investments are made, five-year plans are drawn up, and the first deliverable is more than a year out. Often, in this situation, business stakeholders end up losing patience, cutting funding, and things roll back to square one.

At one of our current clients, the firm decided to invest in sufficient planning upfront to grow their existing business and achieve business agility via SOA adoption. Often a funding problem is like a tax problem – it's good to have the problem, but it's hard to address it. In our client's case, the business was making money – hence it had extra to hand out to IT to innovate, improve, and grow the business portfolio. The good news was that IT was well funded. The bad news was that it had to figure out what to do with the money. The answer – create an SOA blueprint and plan of execution. And then march to it.

So what do you do if you want to venture out into the world of service orientation, but aren't well versed in the technologies involved? You hire external consultants to set your house in order. Well, that's exactly what happened. Business stakeholders brought in external help to draw up the architecture blueprint. A grandiose vision was delivered and signed off on over a period of over a year. When the time to implement came, the client realized that the change was too big, implementation teams weren't in place, and furthermore, the IT department hasn't really bought into the vision, because it didn't fully understand it. Besides, major pieces of the puzzle were incomplete, because the landscape was too big to be addressed at once. Currently I'm in a situation of rationalizing this vision (created by architects no longer consulting with the company), with a client team that doesn't "believe in SOA."



The main reason companies find themselves in such a situation is because people don't grasp that there are various degrees of SOA adoption and there are often several parts to get there. First you have to decide on which approach applies to you – top-down or bottom-up. Is your infrastructure and team organization mature enough that you can identify and define services in business processes that can then be handed out for independent analysis and development? Or do you need to build basic services that will aggregate to composite infrastructure services, which can then be consumed by calling applications? Do you need to build new applications/interfaces, or do you need to address the fundamental issue of making data available for consumption? Do you need to focus on fundamental services, intermediary services, or process-oriented services? If the existing environment is analyzed properly, the appropriate road to SOA adoption can be defined.

One of the main things to accept is that you will reach a degree of SOA maturity and not everyone reaches the same level in the first iteration. Finally, the vision has to be accepted universally by the business and IT stakeholders. ■

***About the Author***

*Ajit Sagar is a principal architect with Infosys Technologies, Ltd., a global consulting and IT services company. Ajit has been working with Java since 1997, and has more than 15 years experience in the IT industry. During this tenure, he's been a programmer, lead architect, director of engineering, and product manager for companies from 15 to 25,000 people in size. Ajit has served as JDJ's J2EE editor, was the founding editor of XML Journal, and has been a frequent speaker at SYS-CON's Web Services Edge series of conferences, JavaOne, and international conference. He has published more than 125 articles.*

*ajitsagar@sys-con.com*

SOA WebServices JOURNAL

www.WebServices.SYS-CON.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 Director**
Robyn Forma robyn@sys-con.com

**Advertising Manager**
Megan Mussa megan@sys-con.com

**Associate Sales Managers**
Kerry Mealia kerry@sys-con.com
Lauren Orsi lauren@sys-con.com

**SYS-CON EVENTS**

**Associate Event Manager**
Lauren Orsi lauren@sys-con.com

**CUSTOMER RELATIONS**

**CIrculation Service Coordinator**
Edna Earle Russell edna@sys-con.com

**SYS-CON.COM**

**VP information systems**
Robert Diamond robert@sys-con.com

**Web Designers**
Stephen Kilmurray stephen@sys-con.com
Wayne Uffleman wayne@sys-con.com

**ACCOUNTING**

**Financial Analyst**
Joan LaRose joan@sys-con.com

**Accounts Payable**
Betty White betty@sys-con.com

**Accounts Receivable**
Gail Naples gailn@sys-con.com

**SUBSCRIPTIONS**
SUBSCRIBE@SYS-CON.COM
1-201-802-3012 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.

SYS-CON MEDIA www.SYS-CON.com

MAKE YOUR ^SOA^ SECURITY MOVES WISELY...

**XWALL**
WEB SERVICES FIREWALL

**XRAY**
WEB SERVICES DIAGNOSTICS

**VULCON**
VULNERABILITY CONTAINMENT SERVICE

**SENTRY**™
SOA SECURITY GATEWAY

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

## FORUM SYSTEMS ENTERPRISE SOA SECURITY SOLUTIONS:

- TRUSTED SOA MIDDLEWARE
- WEB SERVICES SECURITY
- XML ACCELERATION

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

FORUMSYSTEMS™

THE LEADER IN WEB SERVICES & SOA SECURITY

# Extending Identity Management Solutions into a SOA

## Managing and applying policies and controls to Service Oriented Architectures

WRITTEN BY **ROBIN MARTHERUS AND WILLIAM BATHURST**

Companies are under tremendous pressure to meet the complex business requirements found in their IT infrastructures.

For example, they need to expose their applications to external trading partners, comply with government regulations such as Sarbanes-Oxley, integrate merged companies or their own complex application environments. One common solution path that helps solve these problems is to adopt a Service Oriented Architecture (SOA), which allows companies to be more agile in meeting their business needs.

When a company begins to expose their business processes in a SOA, then they will need to ask how they will control access to their services. These interactions can be complex, since SOAs can be composed of many composite applications. These interactions can also be dynamic, where business processes can be re-routed, or modified quickly. There are a number of questions that need to be answered in this type of environment:

- How can access to these services be managed in such a dynamic environment?
- Can access management be centralized?
- Can existing identity management solutions be used to manage access to SOA services?
- Even though identity management access control was built for Web-based applications, is it powerful enough to secure services or business processes?

This article introduces a solution for managing and applying policies and controls to Service Oriented Architectures. First it looks at the need for policies and why they need to be centralized, next it will define the access and identity management problem, and then finally discuss possible architectures that can used for access policy management.

## Access and Identity Management

Identity management infrastructures are used today to address access control and policy management for Web applications. Expanding them to SOAs, where policies will be applied to business processes instead of URLs, will require more sophistication and intelligence from identity management solutions.

Before considering the new requirements demanded of identity management, let's review the current state of identity management. The following definition comes from the Internet-based Wikipedia, which does an excellent job of summing up identity management:

*"Identity Management (IM) is an integrated system of business processes, policies, and technologies that enable organizations to facilitate and control their users' access to critical online applications and resources — while protecting confidential personal and business information from unauthorized users. It represents a category of interrelated solutions that are employed to administer user authentication, access rights, access restrictions, account profiles, passwords, and other attributes supportive of users' roles/profiles on one or more applications or systems."*

Identity management is a mature technology that provides standard features such as delegated administration, user provisioning, policy management, and access control. The security challenges for Web-based applications are very similar to those in the SOA world. They require both authentication and authorization policies, and each has its own policy store. SOA and Web application policies are created and managed with different tools, and the protocols, methods, and session-handling mechanisms of Web-based and Web Service applications differ. The ability to create, manage, and apply policies across both technologies requires advanced identity management.

Let's now drill down into the policy management component of identity management and see how it can be expanded to control access to SOAs. The following sections of this article define policy and describe a system in which policies can be applied broadly and generically across Web Services and applications.

## Policies

Policy means different things to different people. Here are some common definitions:

- Rules of practice and procedure
- An established course of action that must be followed
- The set of rules that govern the interaction between a subject and an object
- A written principle or rule to guide decision-making
- A governing principle pertaining to goals, objectives, and/or activities

For purposes of this article, let's consider a policy to be a set of rules that govern the interactions between entities. An entity could be a person, device, service, and so on. This definition encompasses current buzzword technologies (such as identity management and Web Services management) and is general enough to apply to a broader range of technologies. Let's also define policy management. It can be considered the act of creating, modifying, monitoring, and enforcing policies.

Policies have been in use for some time in the Web server single sign-on world for protecting specific URLs by letting an administrator determine who can access them and under what conditions. Policies of this type, usually called authorization policies, are tightly integrated into identity management architectures in the Web server single-sign-on context. Authorization policies have made identity management one of the essential components of any IT infrastructure.

Policies are also used as integral parts of Web Service management. In this context, they're used to describe the flow of information between a Web Service client and a service. They dictate the format of a request and a response, how they are to be signed and encrypted, and so on. Authorization policies can also be included in Web Services policies.

At a higher level, an organization can have a set of business policies. These kinds of policies generally apply across organizations and describe technically abstract rules. For example, an organization's IT department might dictate that all passwords must be encrypted on the network. Such a policy doesn't describe how the passwords are to be encrypted or even where they're used. It's a very general statement of a rule.

## Generalized Entity Management

Using policies to govern the interaction between entities has to take into account that each entity type often has its own policy management. For example, there are a slew of Web Service management products, with their own specialized set of ways for managing entities. They include quality-of-service monitoring to verify adherence to service-level agreements. For example, a Web Service may have a policy stating that if response times are greater than agreed on, some clients may be redirected to an alternative Web Service. Web-based applications also have their own set of policies that control their interactions with users.

Efforts underway to combine the features of different types of entity management products into generalized entity management products would bring provisioning and delegated administration to Web Service management. There's no reason to have multiple products that manage different types of entities.

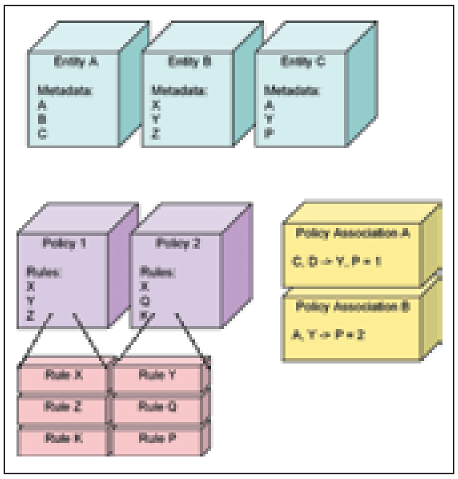


**Figure 1:** Policy association

Any good identity management system can be used to manage entities other than people, but the products aren't designed to manage anything else. It's not hard to imagine that an application can have its own identity when it attempts to interact with a Web Service. An application's identity is very similar to a person's. An application can authenticate itself, and it has attributes such as its location and whether it's a batch process or interacts with users. However, many standard identity attributes, such as "manager," "phone number," and "e-mail address," are specific to people and inapplicable to applications.

## Generalized Policy Management

Just as the management of entities is in the process of being generalized, there's a need to generalize the underlying policy management technology to allow standardized management of existing policy types as well as future ones. At the same time, an infrastructure is necessary that can meet the specific vertical needs of each policy type.

Building an infrastructure that can administer and enforce various policy types is complex. Different policy types are usually administered and enforced by different organizations in an enterprise, managing them has required different applications, and different standards have evolved that focus exclusively on one policy type.

Policies used to be associated with a specific entity. Take, for example, an authorization policy for a Web site or a specific resource (the entity being acted on) on that site. Any person using a browser (the entity doing the acting) to access those resources had to be authorized in accordance with the authorization policy before gaining access. That authorization policy was designed specifically for those resources on that Web site. More advanced authorization engines allowed the same policy to be used for a larger set of resources on multiple Web servers.

Another example might be a process policy assigned to a specific Web Service (the entity being acted on) designating a set of steps required before and after a SOAP request coming from a SOAP client (the acting entity) can be handled.

The ideal way of managing policies would be to enable policies to be managed the same way entities are managed. Policy management will be able to take advantage of the same rich feature sets that identity management has enjoyed for some time now. Policies will be able to go through approval-based workflows, for example. Imagine that you're modifying a policy that specifies which entities can access a Web Service containing sensitive information and that your corporate security office needs to review the modified policy before it can go live. When you make the change in the policy management system and save it, it will automatically create a ticket for the security office for review and approval. The change doesn't go live until after it's approved. This is similar to the approval process built into many identity management products.

Policies can then be treated independently of specific entities, meaning that they can be used and reused for different entity types. For example, you might create a policy stating that some entities can access some resources between 8:00 a.m. and 5:00 p.m. from machines outside the firewall. Then you can associate groups of users and groups of resources with that policy at any time.

## Associating Policies with Entities

An association of policies with entities can be based on the entities' attributes and capabilities. Consider Figure 1

Each of the boxes in the figure represents a managed object. The entities can be users, services, devices and the like. Policies can be created to govern the interactions between these entities. Policies are made up of a set of rules, which are independent of the policies and can be assigned to be part of many policies. Policies are then associated with entities, or groups of entities, based on the entities' metadata.

Policy Association A associates Policy 1 with any entity with Metadata C or D when it interacts with any entity with Metadata Y or P.

The benefit of the combination of the dynamic nature of association and delegated administration is that corporate policies can be defined and associated at the highest level and also require adherence at a lower level. For example, a corporation might have a corporate policy that says, "All passwords must be sent over SSL." A policy defining this requirement can be created, along with a dynamic association, to force all passwords to be sent over SSL. This association wouldn't be reversible by delegated administrators.

Another concept to borrow from identity management is that of advanced groups. For example, identity management leverages the power of dynamic and nested groups. Expanding the use of traditional identity management groups beyond groups of users to include collections of policies, rules, and even associations can easily lead to an expansion of traditional "roles." Traditional roles are generally associated with authorization policies (as defined in role-based access control [RBAC]), but generalized policy management can also mean generalized roles.

All types of entities can act in a role, not just for authorization policies but also to determine which steps to take as part of a process or a company policy.

So what should an expanded policy management system look like?

## Architecture for Policy Management

Policy frameworks have three main components as shown in Figure 2:

1. **A policy server:** the central authoritative policy distributor
2. **A policy manage:** the GUI application that allows the management (creation, validation, monitoring) of policies
3. **And a policy enforcer:** the distributed policy enforcement points, such as gateways and agents

Before an entity can interact with another entity, it must first know what policies govern the interaction. Policy enforcers are part of each entity. For example, Web Services run in an application server, which should have a policy enforcement agent running as part of its process. This policy enforcement agent, which is the policy enforcer, gets policies for the Web Services it controls.

There are two ways an enforcement point can get its relevant policies:

1. **Pull:** The policy enforcement agent queries the policy server for the policy expressions that govern interactions associated with the entity it's assigned to, and the policy server returns a policy document containing the policy expressions associated with a specific interaction.
2. **Push:** The policy server pushes a policy document containing all the policies that are associated with an entity to the policy enforcer for that entity.

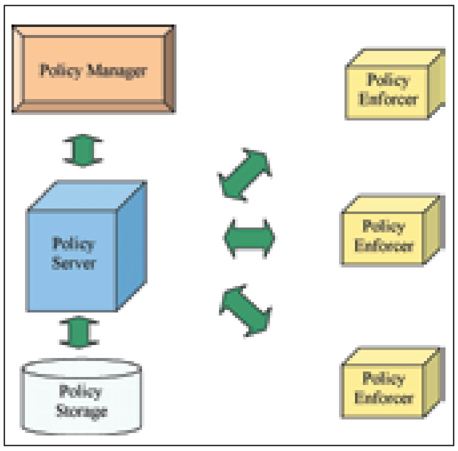


**Figure 2:** Policy management architecture

Because of different requirements for different policy enforcers, a generalized policy server must support both the push and pull models of distributing policy documents. In fact, a single interaction between two entities may require both pushing and pulling policy information. For example, Entity A wants to interact with Entity B. The policy server may have pushed Entity B's policies ahead of time. Before Entity A can interact with Entity B, Entity A may need to know some aspects of the policies governing the interaction. Entity A may query Entity B for the relevant policies, or it may query or pull the policy information from the policy server.

No current standard is sufficient to provide the flexibility necessary to express all types of policies. WS-Policy is widely used to describe Web Services policies. Authorization policies are often described by another standard called XACML. WS-Policy by itself can't describe authorization policies nor can XACML describe Web Services policies. It's unclear if it will be necessary to develop a so-called "Über" policy language capable of describing general policies.

The policy server, combined with an entity management server, can be used as an authoritative registry for entities, their capabilities, and their policies. It's essentially a Universal Description, Discovery, and Integration (UDDI) server on steroids.

Because policies can be very complex and may be created at different levels by different people, a policy server has to be able to resolve conflicting policies. Rules of precedence should be part of the policy manager application.

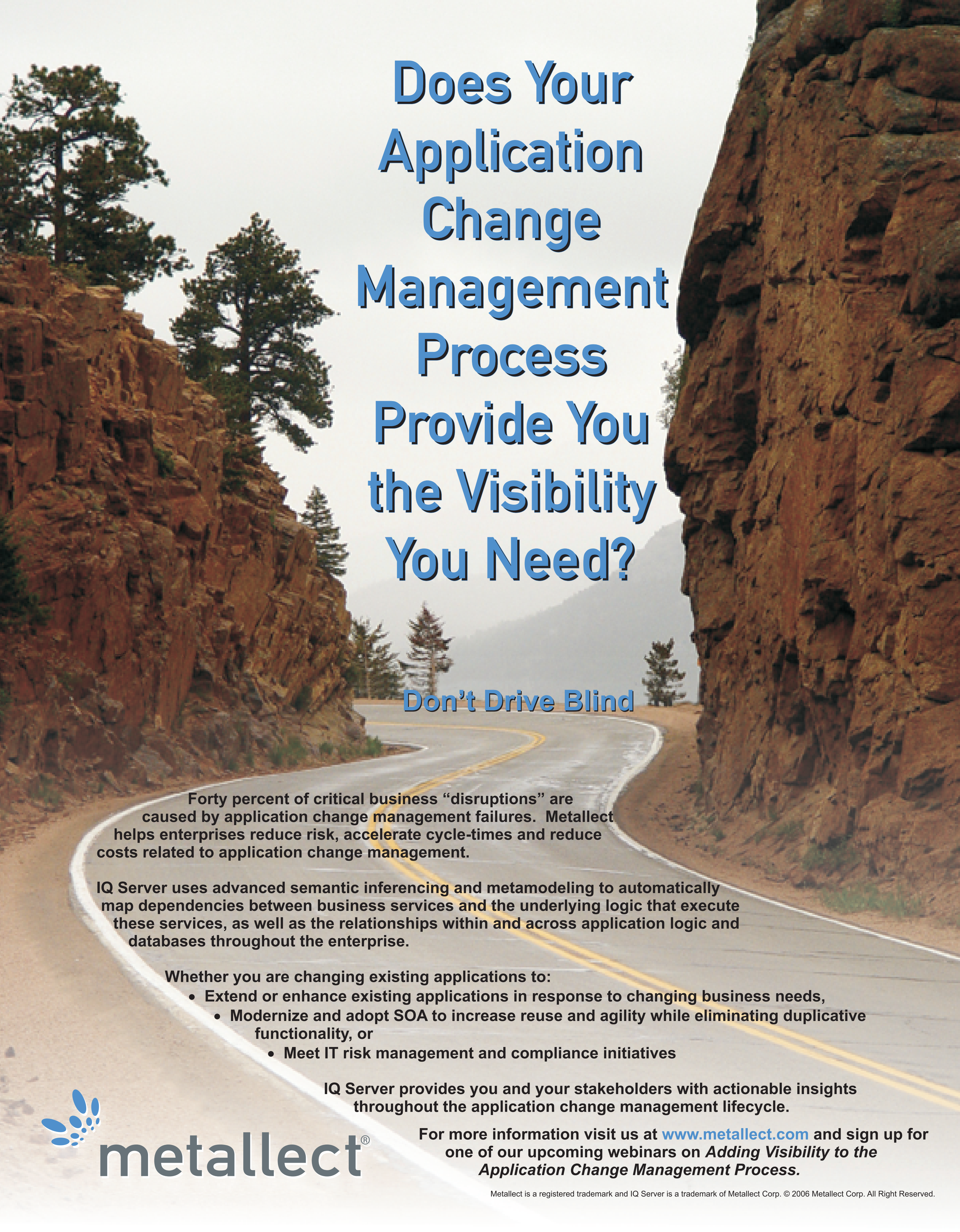
## Conclusion

Identity management is evolving to satisfy the need for more generalized entity management. It must be able to address the various types of entities found in corporate infrastructures, such as persons, services, and devices. With the focus on policy, businesses have to be able to control how access is managed across all their applications easily and consistently whether they're Web-based applications or Web Services. This will provide IT with a flexible approach to managing access and applying policies across application and SOA environments. ■

***About the Authors***

*Robin Martherus is a consulting member of technical staff in the Security and Identity Management Group, a part of Oracle's Fusion Middleware. Robin was previously with Oblix as a senior developer.*

*William Bathurst is a senior product manager for platform security at Oracle. He's part of the Oracle Fusion Middleware product management team. As a seasoned eight-year veteran of Oracle, he enjoys pushing the limits to improve identity management and security to meet customer's business needs.*



## Does Your Application Change Management Process Provide You the Visibility You Need?

**Don't Drive Blind**

**Forty percent of critical business "disruptions" are caused by application change management failures. Metallect helps enterprises reduce risk, accelerate cycle-times and reduce costs related to application change management.**

**IQ Server uses advanced semantic inferencing and metamodeling to automatically map dependencies between business services and the underlying logic that execute these services, as well as the relationships within and across application logic and databases throughout the enterprise.**

**Whether you are changing existing applications to:**

- **Extend or enhance existing applications in response to changing business needs,**
- **Modernize and adopt SOA to increase reuse and agility while eliminating duplicative functionality, or**
- **Meet IT risk management and compliance initiatives**

**IQ Server provides you and your stakeholders with actionable insights throughout the application change management lifecycle.**

metallect®

**For more information visit us at www.metallect.com and sign up for one of our upcoming webinars on *Adding Visibility to the Application Change Management Process.***

# Service Components

## The Quickest Path to a SOA

WRITTEN BY **ATUL SAINI**

The emergence of the Enterprise Service Bus (ESB) over the past two years has spurred the deployment of componentized applications based on a Service Oriented Architecture (SOA). SOA enables the development of business systems and processes with loosely coupled components (often called services) facilitating business agility. Much of the industry's focus has been on the architecture of the underlying ESB infrastructure that supports an SOA.

Standards adoption has made it easier to learn new tools, but basic ESB infrastructure improvements haven't significantly reduced the effort involved in deploying and managing new business processes. That's because an ESB is simply a platform that unifies the advantages of multiple previous generations of middleware. This article illustrates that a comprehensive model for service components – the application-level modules wired together over an ESB to support SOA – facilitates rapid deployment of componentized, extensible, agile business processes.

### What Are Service Components?

A service component is an independent software application that executes a specific business-level function. For instance, while operations such as "update database table" have a technical meaning, a service component always executes a business-level function. As an example of a service component, consider an application that updates a general ledger in an accounting system or product information in a supply chain system.

This article explores core attributes that a service component must support to enable rapid deployment of an SOA. Since these are mostly independent of the underlying infrastructure over which the modules are deployed, service components can – with basic support from each target platform – be deployed over an ESB, an application server or even a raw Message Oriented Middleware (MOM) platform to create an enterprise SOA.

### Key Service Module Characteristics

SOA deployment normally involves orchestration of multiple service components to solve a particular problem. You can reduce the time it takes to deploy an SOA if your service components support the key characteristics below.

A service component:

- ***Can be self-describing:*** A service component is a self-sufficient (self-contained) bundle of software, which can be exchanged between multiple entities and reused with no external dependencies. Service components typically expose interfaces in Web Services Description Language (WSDL) or XML, and can be searched and queried with standard tools, including tools that support the Universal Description Discovery and Integration (UDDI) standard among others.
- ***Can be automatically deployed (usually without manual installation):*** A service component can be remotely deployed, monitored, updated, and controlled over a highly distributed infrastructure environment from a single point of control. Auto-deploy capabilities require that all implementation dependencies (e.g., libraries or other platform-specific files and data) of the component reside in the component when the component is remotely deployed.
- ***Can be developed in multiple languages (C/C++/Java/.NET):*** A service component allows the development of interfaces in any supported language native to a platform without having to create or use language translation code. Since a service component is completely modular, the language used to implement the underlying logic isn't relevant to the exposed interface. So a single distributed SOA process can involve interactions between service components developed in different programming languages. Support for multiple languages is especially attractive for deploying an enterprise SOA since few enterprises have homogeneous environments.
- ***Is configurable, using a contextual user interface (typically specific to each service component):*** Easy configurability for each service component used in a business process implementation is vital for rapid SOA composition and deployment. A service component lets you bundle a customizable configuration interface (specific to each such component) with the component. This gives SOA designers maximum flexibility in deploying SOA processes, while enabling the underlying platform to handle the entire component lifecycle from discovery to configuration, deployment, and runtime management.
- ***Exposes changing, dynamic service definitions on a configuration (usually with input and output schemas that can change based on the configuration):*** For maximum flexibility, a service component must let the invocation interfaces of the component be dynamically generated based on the user's configuration. Introspection of the configuration parameters

of the service should automatically lead to the generation of custom input and output interfaces. The ability to associate different input and output schemas (i.e., interfaces based on configuration) supports the development of a reusable service component that can "adapt" the invocation interface based on requirements of a particular instance of the module. The generated interfaces can be obtained in WSDL, allowing the invocation of the component as a Web Service among other options.

- ***Has pluggable synchronous and asynchronous invocation interfaces:*** An important requirement for flexibility in process design is to ensure that components can be invoked using both synchronous and asynchronous interfaces. This means the interfaces of the component are completely abstracted from the business logic that the component executes. This separation provides the ability to reuse the same service component in different contexts. For instance, a database update can be triggered asynchronously each time mail arrives in a specified mailbox; in a different context, the database update can be invoked synchronously by a Web portal during order processing. Detaching an invocation interface from the body of a service component lets a single technology framework support both Event Driven Architecture (EDA) and request/reply semantics with seamless interoperability – an important goal for any SOA platform.
- ***Has "external" transport bindings, allowing transparent invocation over multiple transports:*** Most enterprise environments have multiple middleware transports in deployment, often with significant investments in technology, including raw TCP sockets, Common Object Request Broker Architecture (CORBA), HTTP, SOAP, Java Messaging Service (JMS), MQSeries, and TIBCO/Rv. External transport bindings let you use a service component with the most optimal transport (or an already available transport) in a specific deployment without having to use transport adapters. External transport bindings permit maximum flexibility in development since different service components in a single business process can use separate transports. The ability to switch transports "on-demand" protects legacy investments and enables optimal performance.
- ***Can be recursively grouped as sub-components in a larger composite component:*** Service components let you create larger components (and processes) that represent larger reusable blocks of functionality specific to a given problem. This recursive grouping capability simplifies the maintenance of large processes (due to compartmentalization), allowing separate portions of the process to be developed and tested independently, potentially by different teams of developers, before the production environment is updated.
- ***Supports multiple deployment options (as a standalone executable embedded inside a container, etc.):*** Flexibility of deployment requires a service component to support multiple deployment options, including deployment as a standalone executable or embedding in a Java 2 Enterprise Edition (J2EE), Microsoft .NET, or Web container. This enables reuse of existing infrastructure investments without adding complexity or managing multiple independent deployments. Multiple deployment options are especially useful for small deployments where independent hardware and software management isn't warranted.
- ***Can be monitored and managed at runtime using standard and remote consoles:*** Agility of business process composition requires service components (which usually run strategic processes for a business) to be individually monitored and managed, with the ability to alert users of potential problems in real-time. The best systems enable component integration with third-party tools such as HP Openview, IBM Tivoli, etc., reducing reliance on redundant management and monitoring systems and providing proactive notifications of potential trouble even before it surfaces.

## Conclusion

With service components that support the attributes listed in this article, it's feasible to deploy flexible business processes over a wide variety of platforms. Even purpose-built "SOA platforms" can, at best, support only core infrastructure-level features such as fault-tolerance, re-routing of messages at a transport level, quality-of-service, scalability, and performance. In and of itself, the infrastructure can do little to ease the creation of flexible processes since speed of deployment and composition is largely orthogonal to the infrastructure. For maximum agility, a process has to be composed of modular service components knit together over multiple transports using multiple invocation methods, whether synchronous or asynchronous.

### *Business:*

- Service components represent the industry's first "Lego block" concept, enabling composition of networked processes by relatively non-technical business analysts.
- Applications based on service components are flexible and can be changed without stopping processes, enabling a real-time response to changing conditions.
- Service component-based processes enable reuse of existing infrastructure software assets, reducing the overall cost of deploying new business processes.

### *Technology:*

- Infrastructure platforms such as the ESB or even the new SOA platforms are basically like previous generations of middleware; they don't, by themselves, speed up SOA deployment.
- Service components drive an engineering discipline that forces developers to modularize their solutions, making them more maintainable.
- Service components support both event-driven and request/reply semantics, enabling modular solutions that are easy to change in response to business requirements.

***About the Author***

*Atul Saini is CEO and CTO of Fiorano Software Inc. Under his leadership, Fiorano has emerged as a recognized leader in the standards-based integration middleware business.*

*atul@fiorano.com*

# SOA and Data Integration... The Realities

## The marriage of data integration and SOA could end up in divorce

WRITTEN BY **DAVID S. LINTHICUM**

First, the history. Data integration is the name the vendors have adopted to replace the ETL (Extract Translate Load), data cleansing, and data warehousing tools of days gone by.

These tools actually pre-date the notion of EAI, and were really the first sets of technology designed to deal with data and the use of that data for decision support (business intelligence now). They would extract large amount of data from a single or several operational data stores, clean the data, roll up the data, and put it in another data store, typically the data mart or data warehouse for analysis. From there somebody would "mine" the data to extract relevant information, such as productivity over time, sales effectiveness over years, profit by division, you get the idea. Very powerful notion for its time, and very powerful technology today.

When I was first working on the notion of EAI, I used these tools, and thought they had value. However, I also understood that the value of integration was really about the movement of information in real-time, system-to-system, in patterns that resemble actual business processing (sales to inventory to accounting, etc.), and while there was some business intelligence there it was really in the domain of real-time monitoring (BAM). Thus, you really had two different threads of technology born...ETL (now data integration) and EAI (now integration or SOA).

Enter SOA, and the hype around it, and everyone looking to link to it. All data integration vendors, and EAI vendors for that matter, are repositioning, retooling, and remarketing their technology as a "SOA solution." So, where is the actual fit for data integration?

Unlike SOA, which can support real-time data movement, data integration (typically) provides adequate business information (data replication) without up-to-the-minute access of information. In many cases, the data is weeks, even months, old, and the data mart or data warehouse is updated through antiquated batch, extract-aggregate-and-load, processes. Indeed this is the way integration is done today, using a data integration product or in most cases no product at all. When I was doing research for my last book, for instance, I found that FTP was still the primary form of data integration.

### Evolving Away, and to Data

Things are changing, fortunately. SOA, and the technology that comes with it, lets data warehouse architects and developers move information...no matter where it comes from or where it's going...as quickly as they want to move it. As a result, it's not unheard of to have all participating databases and services in a SOA solution getting new data constantly, thus providing more value to those using the source and target systems existing in the SOA...including those who use them as a data warehouse or data mart.

Therefore, the rise of SOA will also lead to the rise of real-time data warehouse solutions, or could replace the notion of data warehousing altogether. For instance, we could put the data behind services (data services or abstract data services), with users leveraging up-to-the-minute information to make better business decisions using BI tools or BAM, or just snapping them into composite applications.

As mentioned, as time goes on data integration products may not be needed as SOA architects craft services that abstract both operational and aggregated data, in some cases leveraging aggregated data without having to replicate and change the data, but doing it through abstraction layers. This approach, if possible for your domain, is much less expensive and less complex. In essence, the SOA becomes the place to leverage services that can deal with the data layer(s) through many types of abstraction services, services that can be mixed and matched in composite to create solution instances for the SOA. SOA's value is in bringing all of these things together as a platform for business solutions.

### Needs Coupling... Okay, Some Coupling

For a technology to truly be a SOA technology, or so I argue, it have to support the

*— continued on page 17*

# 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

# SOA and Integration Testing: The End-to-End View

## Challenges and benefits of the end-to-end testing solution

WRITTEN BY **CHRIS BENEDETTO**

We've seen a dramatic rise in the use of SOA and integration to provide better business process visibility and agility to organizations. The ease and low cost of assembling new systems together makes SOA an efficient and valuable business asset.

But the challenge remains: With so many moving parts in the SOA environment, and a multitude of heterogeneous systems to expose as services, how can businesses ensure all those crucial parts are working properly?

End-to-end SOA and integration testing provides the necessary insight to see what's happening inside the SOA environment and validate that services, messages, interfaces, and business processes are performing properly. End-to-end testing identifies elusive defects and corrects them — before the defect causes serious workflow interruptions.

To understand the full scope of end-to-end testing, businesses should be aware of the challenges of an SOA environment, why only end-to-end testing will work in this space, and the return on investment in end-to-end testing.

### SOA Challenges

In SOA environments, systems and applications depend on one another to complete a business process or service (see Figure 1). But the numerous points of exchange create numerous opportunities for defects. To find and correct these defects, SOA and integration testing must address five core challenges of the SOA space:

- Defects in the SOA space are extremely difficult to diagnosis because the data in messages is buried in transport protocols that are inaccessible to the typical tester and system administrator. As a result, these defects usually aren't seen until the full system can be tested at the very end of the project, after potential problems have multiplied throughout the system and are more costly to fix.
- XML content and messages that are transmitted back and forth in an SOA environment often encapsulate the actual substance of the message, plus the function that it's supposed to perform. These messages are often written in SOAP (Simple Object Access Protocol), and may contain many customized formats and fields. An error can occur anywhere in the large number of fields, creating an enormous set of permutations and error points and making it extremely hard to have an effective testing solution around just the Web Services.
- Web Services and SOA demand very explicit and predictable inputs and outputs. This largely hinges on the accuracy of the application programming. For example, in Microsoft Excel each cell must be formatted to accept the proper data. Cells for dollar figures must accept dollar signs; cells for decimals must accept decimal points, etc. Unless this is explicitly expressed at the API level, errors in message communication can abound.
- The SOA testing model isn't just about unit testing. When businesses build an SOA or integration initiative, they're pulling and modifying data from dozens of different systems. Many systems provide only a confirmation message that a process, such as an update, has occurred. There's no guarantee that the data that came from System x, that's now in System y, is accurate or has been put in the right place. Again this opens the door for error if the update is affecting multiple systems and isn't executing correctly.
- Increased business process agility is the overwhelming advantage of moving to SOA. As SOA becomes more prevalent, SOA governance standards will become more prevalent too. However, because SOAs change frequently, it will be difficult to develop and adhere to governance standards and provide high-quality services without adopting a test-driven SOA approach.

### Traditional versus SOA Testing

To overcome these challenges, most developers and quality assurance professionals have resorted to using traditional GUI testing tools. However, GUI testing is screen-based and doesn't address all of the deeply embedded moving parts that SOA and integration testing do. There are three key differences between the testing styles that make GUI-based testing ineffective for SOAs:

- Traditional testing is focused on the front-end. It involves checking aspects of the application that can be readily seen such as proper performance levels and the correct working order of all connections and interfaces.

  SOA and integration testing isn't focused on the GUI. It's focused behind the screen on the middle tier including application servers, enterprise services buses, legacy assets, and connection points between systems. SOA and integration testing is a headless, almost stateless, environment with very low human visibility or interaction.

- Traditional testing is more code-centric and application-specific. That means the developer writes code for a specific application and tests it.

  SOA and integration testing, on the other hand, is more assembly-centric and workflow-centric. Developers assemble components from existing applications and turn them into services, or they access components through a service repository and string them together. Developers need not only to test all the individual components, they need to test while the application is in the assembly phase and check that the workflows and business processes are properly connected.
- In traditional testing environments, the development team can test for defects in the system at the desktop level. Since GUI-based systems aren't overly dependant on one another, defects can be found more easily, contained, isolated, and repaired.

In SOA and integration testing environments, systems are distributed, highly dependent on one another, deployed on heterogeneous platforms, and often not even available. Defects in one application or message can cause a cascade of failures system-wide.

## The End-to-End Answer

Traditional testing's front-end application-centric approach makes it ineffective in the SOA environment. SOA requires end-to-end testing to see deep inside the SOA, check for defects, and correct them before they interrupt critical business activities. The end-to-end SOA and integration testing market delivers four critical components that do this:

- ***Increased coverage.*** SOA and integration environments are made up of many components: Web Services, enterprise service buses, legacy assets, databases, files, and numerous transport protocols that move messages and orchestrate services.

  What's needed is a testing tool to address these components and the layers of complexity because they contain a tremendous amount of logic that falls outside the domain of traditional application testing, such as the dollar sign and

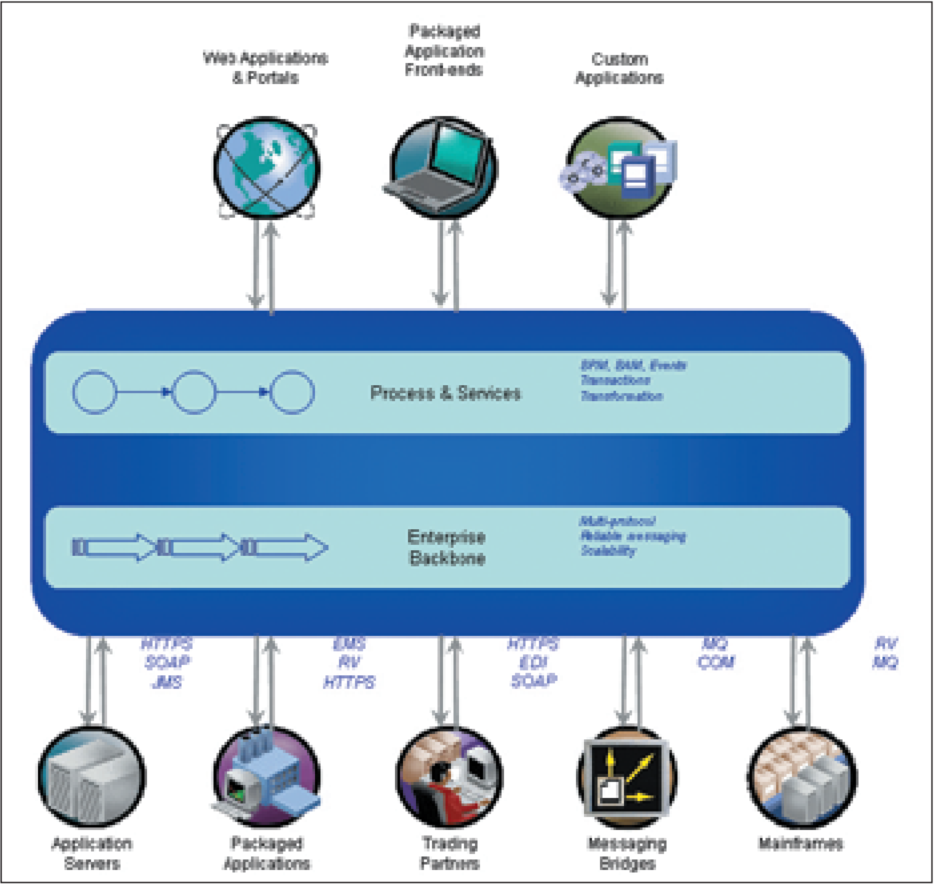


**Figure 1:**

## SOA and Data Integration...

*— continued from page 14*

notion of coupling, as well as cohesion, and not just one or the other. This is where some data integration products fall down. Coupling, in the context of application integration and SOA, is the binding of applications together so that they are dependent on each other, sharing the same services, methods, interfaces, and perhaps data. This is the core notion of SOA where the applications are bound by shared services, versus the simple exchange of information (using services or not).

Of course, the degree of coupling that occurs is really dependent on the SOA architect, and how she or he binds source and target systems together. In some instances systems are tightly coupled, meaning they're dependent on each other. In other instances, they are loosely coupled, meaning that they're more independent. It doesn't matter if you're doing this through Web Services or other mechanisms; you're typically going to have to make these architectural tradeoffs within the notion of coupling.

There are, of course, more pros and cons of coupling that should be considered in the context of the problem you're looking to solve. On the pros side you have the ability to bind systems by sharing behavior, and bound data, versus simply sharing information. This provides the integration solution set with the ability to share services that could be redundant to the integrated systems, thus reducing development costs. This is the reason we leverage SOAs.

Then there's the ability to tightly couple processes as well as shared behavior. This means that process integration engines, layered on top of SOA solutions, have more skill at binding actual behavior (functions, methods, services) versus just simply moving information from place to place.

The problem is that many data integration solution are more about information/data than about sharing services, so they're hard fit for many SOAs. ESBs have a similar issue, but not as obvious. As a result, the marriage between data integration and SOA could end up in divorce if coupling is a requirement. Again, generally speaking. 🟧

***About the Author***

*David S. Linthicum is the president and CEO of BRIDGEWERX, and the author of several books on application integration and service-oriented architecture, and the host of the SOA Expert Podcast.*

*david@bridgewerx.com*

decimal point example mentioned earlier. That example illustrates how deeply embedded logic can cripple a composite application that has been built from SOA or integration components. A traditional application testing tool wouldn't find these defects, but SOA and integration testing tools will.

- ***Test automation.*** SOA's assembly-based approach strongly favors the use of automation and test-driven development techniques. That means developers can automatically test at each stage of an application's development, or with every sprint release.

  This way the application is "evolved" as opposed to built and tested. The only way to constantly test is through an automated testing environment that supports agile practices and tools. Traditional testing tools don't have this automated capability.
- ***Process visibility.*** SOA testing helps QA teams and developers inspect Web Services, business process, and messages across transport protocols at many different levels. Developers can look at an actual message that moves from system A to system B, and correlate that message to a larger business process, such as making a deposit at the bank, and make sure that the larger process ultimately commits and executes as planned.

  Unlike traditional testing, SOA testing lets developers see each step of a process and the end result. This way SOA testing makes it easy to design test scenarios for complex business processes.
- ***Reuse.*** As SOA and integration environments grow larger and larger, and test case volume grows, communications between development, operations, and QA personnel can get skewed. To administer such large integration efforts and validate quality, businesses need a tool that lets developers, system administrators, and QA teams share test cases between them.

  SOA testing tools can do this by storing each test case and libraries of test suites, and allowing different application development personnel to access them. For instance, if a QA employee finds an error, he can quickly reference the specific test case and consult the developer to address the issue.

## End-to-End ROI

End-to-end testing streamlines the entire process of discovery, diagnosis, and validation for SOA and integration environments in several quantifiable ways:

- Increased test coverage through rapid root cause identification. End-to-end testing provides the ability to identify and measure problems quickly, and generate specific reports and metrics related to SOA and integration quality.
- Cost savings. The loosely coupled yet highly connected applications and services in SOAs mean the impact of a single defect will be magnified throughout an organization. Finding those defects beforehand saves the cost of going back and fixing the problem after production, which can be as high as 10 times the original cost of production.
- Rework reduction. For each patch release put out, there's an initial cost of at least $65,000 to $70,000 for that patch. Multiply that by the hundreds of defects out there and the hundreds of patches that are needed and the cost adds up quickly.
- Reduced staff involvement and increased turnaround time. Automated end-to-end testing allows testing to happen at the click of a mouse, which can positively impact release frequency and quality as well as testing turnaround time. This increase in work efficiency can also decrease staffing costs.

### The Solstice Integra Suite

Solstice Integra Suite is a prime example of an end-to-end integration and SOA testing suite that shows businesses what's really happening deep inside their integration and SOA infrastructure. It provides end-to-end testing and validation by providing businesses with visibility across multiple protocols and multiple vendor architectures.

Integra works by automating the testing and troubleshooting of businesses' integration and SOA deployments before they go live, saving the time and cost of after-the-fact debugging.

By supporting industry-standard, vendor, and file database protocols, and integrating with Mercury, JUnit, and ANT, Integra limits risk and quality issues in large integration projects. Everyone involved in the project, including developers, testers, project managers, QA professionals, and offshore coordinators, uses Integra to:

- Gain visibility into message structure and content, and compare the content and path of messages
- Simulate unavailable systems
- Automate unit, functional, and regression testing
- Customize test scenarios such as parameterized input, validation of updates and replies, and multiple transport options
- Create a central shared repository of test cases

More information on Solstice Integra Suite can be found at http://www.SolsticeSoftware.com/Products.aspx

## Summary

The SOA and integration testing world is a world of "behind the screens" testing where defects in the systems are some of the most elusive to find — and most expensive to correct. One defect can cause a domino effect of poor quality across business processes and supporting systems.

The most effective way to validate SOA business processes is through a continuous end-to-end testing process that inspects each layer of the SOA and integration infrastructure as it's developed, run, and maintained. Further, the ability to see deep inside the SOA and integration environments provides an ROI that's vital to a streamlined and cost-effective business. ■

### *References*

*For more information about end-to-end integration and SOA testing, visit:*
- *http://www.SolsticeSoftware.com*
- *http://www.SOALink.com*
- *http://www.IntegrationConsortium.org*

***About the Author***
*Chris Benedetto is vice-president of marketing for Solstice Software, a leader in automated, end-to-end integration and SOA testing. He has nearly 20 years of experience in software sales, marketing, and development for leading software companies including Hewlett-Packard.*
*CBenedetto@SolsticeSoftware.com*

# Sound Architecture Requires Proper Planning

WEB AGE SOLUTIONS - YOUR TRAINING PARTNER FOR SOA

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

## Custom training for complementary SOA technologies

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

## Custom training plans for virtually every job role

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

## Consulting services for all phases of SOA migration

IBM Premier Business Partner

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

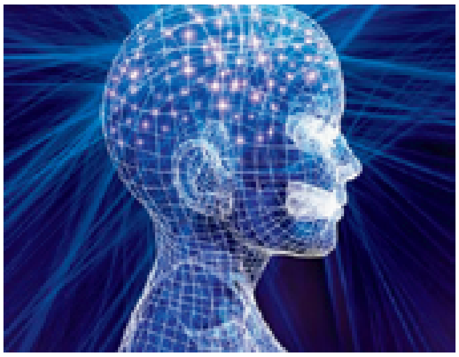
www.webagesolutions.com - 877.517.6540 - soa@webagesolutions.com

WEB AGE solutions inc.

# Fiorano SOA 2006 Platform

## A honey of a product

REVIEWED BY **WARREN HAMPTON**



**SOA, EDA, BCM, ESB and BPEL...More than IT Catch Phrases?**

I recently had the chance to evaluate the next-generation Fiorano SOA Platform 2006 suite from Fiorano Software, Inc. As an architect and developer who's worked with previous versions of the kit over the last three years in addition to several competitor offerings, I looked forward to sitting down with Fiorano's latest release.

For those unfamiliar with the product, it's a feature-rich SOA/BPM development, deployment, and administration suite built on the company's J2EE ESB technology. It relies on standards-based services, shared component modeling, peer-to-peer communications, and high-performance event-driven messaging (pub/sub and queuing) across a distributed network. This latest release brings a standalone BPEL Editor, Composite Components, pre-built JCA-compliant adapters, a Business Development Component Kit and Shared Resource Pools among other enhancements (see Figure 1).

With intuitive interfaces, pre-built components, and data adapters, business analysts can build business process workflow applications with little or no programming. These components aren't graphical representations for diagram purposes; they are fully functioning feature-rich services that can be added to the application with drag-and-drop ease and are configured using simple parameters.

This is not to say that developers can't build complex coarse-grained custom services to leverage the full power of the suite. The extensive toolset has allowed our developers to concentrate on extending and refining richer applications to meet business requirements in a fraction of the time of competitors' offerings while encouraging code re-use, SOA design, and solidifying patterns and practices. Another key point is the platform's low overhead, full scalability, unassisted fail-over, and flexibility to run on a single server or across a highly distributed WAN environment or load-balancing cluster with ease.

## Getting Started

First note the decision to rename the suite from Fiorano ESB to Fiorano SOA. This makes perfect sense. I've always maintained that previous versions were more than a toolset for building ESB-based solutions. Although Fiorano originated in high-performance message queuing, the platform has matured into a powerful, scalable service-oriented development platform on top of the proven messaging abilities. Earlier versions allowed for rapidly building robust workflows and quick integrations across disparate systems and relied on intuitive administration interfaces for views into daily development and production processes. I was eager to see what the latest iteration would add.

The installation is simple, a few point-and-clicks is all you need to get the full Enterprise version installed and ready to roll. Although the suite is intuitive I'd recommend browsing the Startup and the Platform Concepts guides before beginning the install to familiarize yourself with the primary components and the depth of the product. You can find extensive documentation and code samples at http://devzone.fiorano.com/devzone/dev_zone.jsp.

An improvement over previous versions is the built-in ability to launch both the servers and individual processes as Windows Services. This allows for auto-restarting services and applications as well as monitoring the services using commonly available network monitoring tools. I would, however, have liked to see this as part of the initial installation. The rules-based security is easily understood, set up, and maintained but also robust and flexible enough to pass stringent security standards. Integration with LDAP services is also an option.

Once installed you'll find the following:

- **Fiorano ESB Server 2006:** A web Services-capable middleware platform
- **FioranoMQ Server 2006:** Peer-to-peer JMS messaging platform
- **Fiorano BPEL Server 2006:** A distributed BPEL processing orchestration engine.
- **Fiorano Business Components and Adapters 2006:** Ready-to-use JCA-compliant components
- **Fiorano Process Orchestration Tools 2006:** Integrated development and admin toolset
- **Fiorano BPEL Editor 2006:** A standalone BPEL design, development, test, and deployment tool set

## SOA, EDA, and ESB in the Real World

Although real-world sample applications are included I wanted to see how I could improve existing workflows created in previous versions as well as solutions built with other tools to see if there were some viable gains in this "major" release to further simplify the development process and make it easier to build, track, debug, extend, and monitor these solutions. I choose more complex workflows that get XML messages from external sources via HTTPS posts or Web Services, rely on content-based XPATH queries for routing and XSLT transforms from external to internal formats, create physical archive files, update an SQL DB, and again transform messages to one of many positional or delimited file formats for delivery to legacy systems. Also included along the way are extensive error-handling alerts and simple business rules.

What I found was very encouraging. As previously mentioned the suite includes an Event Process Orchestrator to build flows. I was glad to see that this had retained its look-and-feel while the pre-built services have improved in many key areas in regards to continuity and configurable parameters. Also apparent was an improvement in speed when opening the configuration dialog screens. I was quickly able to build new loosely coupled versions of the existing applications (including non-Fiorano solutions) leveraging many of the enhancements such as improved services, data adapters, composite components, and shared resource pools to create highly efficient applications.

Several new or enhanced pre-built services and data adapters are offered, some of which include BeanShell script components, improved DB, HTTP, Web Service, and transform components as well as support for adding iWay JCA-compliant services to the flows (www.iwaysoftware.com). These, added on top of the existing extensive palate of services, protocols, and adapters, (60+) went a long way to letting me build richer enterprise-level solutions with little or no custom programming. Not that you're limited in this area; improvements have been made making it easier than ever to build custom components that can be added to the existing palette using Java, C, C++, and C# as required.

One of the main improvements is the enhanced ability to build composite components that are repeated many times in day-to-day workflows and integrations. Easily and effectively I re-created large pieces of a complex workflow that can be called on from multiple applications. This is a great time-saver and helps you adhere to code re-use principles and enforce standard handling of repeated processes. Another welcomed enhancement is the improvements to what was arguably the best mapping tool available. FioranoSOA 2006 now retains existing mappings when adding to the underlying schema. This is a real time-saver when you consider the implications of re-mapping an extensive transformation from scratch to add an additional tag to a complex schema.

## Business Process Meets Application Workflow...

One of the most striking things I always found with the platform was its ability to bridge the divide between business process and application workflow modeling. This is even truer today using a combination of the BPEL Studio, the Event Process Orchestrator, and an extensive palette of standards-based, pre-built functional services; everyone from business analysts to developers and administrators can conceptualize workflows, build applications, test, and monitor the full development process from a common interface without extensive knowledge of the underlying services.

The ease with which you can create and deploy working solutions must be seen to be appreciated. Many times I have built a working solution before someone's eyes and deployed it only to be asked, "Okay, so now what are the steps required to build, code, and deploy the application?". By immediately running the application and showing the messages flowing is it clear a fully functioning flow can be created almost as quickly as you visualize it. Compared to competitor offerings FioranoSOA excels in this area.

Production applications can even be extended, debugged, or corrected without downtime to the application, something unique to the platform and eye-opening to those accustomed to late-night builds, pushes, and testing. Application profiles help control versioning between development phases and push-to-production environments. The platform challenges the traditional development process methodology, saving significant time and money when a production issue has to be corrected. But it still fits perfectly into the standard process and eases many of these tasks compared to alternative methods.

The speed with which you can do change requests to a business process is where the application and toolsets really shine. I have faced major production issues after a deployment in which the ESB platform had no direct affect, acting largely as a transport/transformation layer. However, due to the power and flexibility of the platform I was able to correct the issue by applying business rules that altered the messages in the production application in less than one hour, whereas the legacy system sending the data in question would have taken four to six hours of development, testing, and a late-night build to resolve the issue by the next business day. Starting and stopping services at runtime, components like the Feeder Service that can send messages to the orchestration or the display service that shows you the output from any service port help to speed debugging and re-queuing from any point in the process. The ability to capture messages via event interceptors on the routes between services empowers you to queue messages, edit services, and analyze log files without disrupting other services in the flow.

The cost savings in these scenarios is hard to quantify since it's very far-reaching in a high-volume transaction-based business that directly affects revenues based on these messages; needlesstosay it's significant. The savings starts at the development and testing stage where the cost savings created by the speed with which adjustments can be made carries all the way through ongoing maintenance of the platform and solutions in production. At each stage significant gains are realized when compared to competitive offerings or traditional coding methods.

This flexibility also allows for simple scaling of the platform. You can easily add more peer servers, fail-over solutions, load-balancing flows, and error handling without significant impact on your applications. You can go from test to beta to QA to production in a very short period of time and deploy these updates with confidence and minimal risk. The support included for resource sharing across mul-

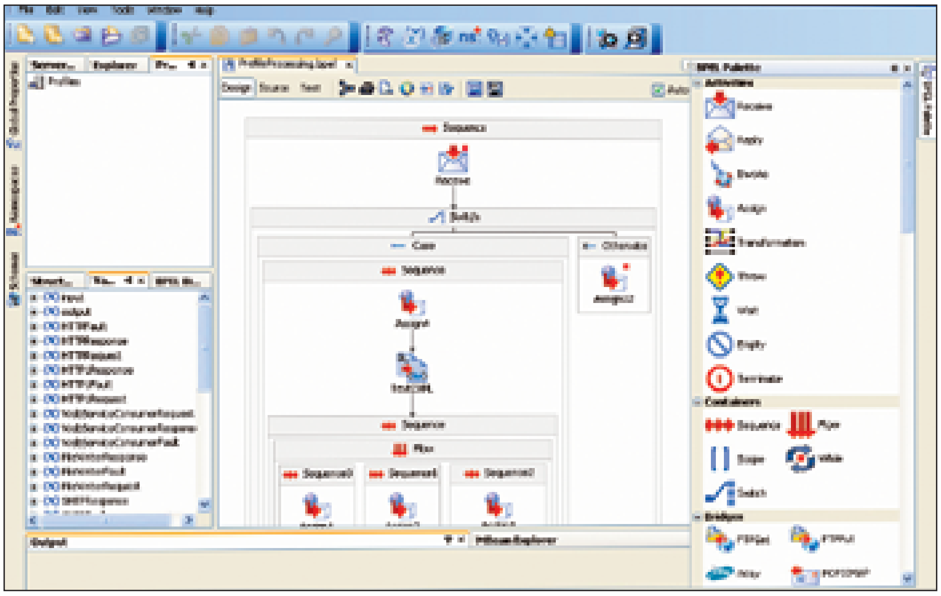


**Figure 1**

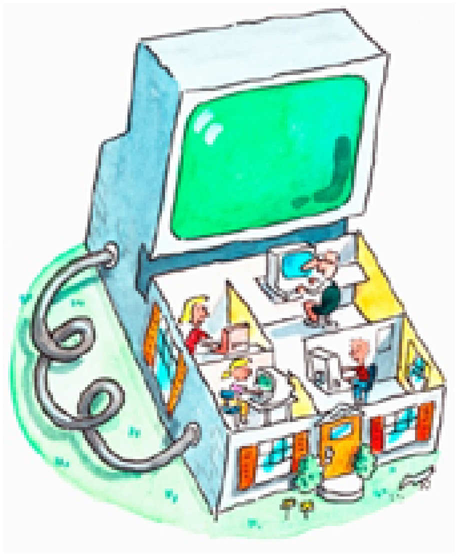
— *continued on page 49*

# Mainframe SOA Best Practice

## Contract-First Design

WRITTEN BY **ROB MORRIS**

To ensure the success of your mainframe SOA initiatives, it's important to be able to support both bottom-up and contract-first design approaches. With the former, businesses may see an opportunity to jumpstart the SOA, quickly packaging bite-sized chunks of mainframe code as Web Services, and pushing them out to the rest of the organization to do with them what they will. But experience shows that the contract-first approach – basically, Web Service design informed by business processes – is a "best practice" that will yield optimal results.



### The Myth and Pitfalls of "Instant SOA"

Let's look at the difference between the two methodologies. With the bottom-up tactic, generally adopted for "instant SOA," the mainframe developer wraps pieces of mainframe functionality as isolated Web Services, and basically throws them over the wall to be accessed by various end-user applications and systems. This sort of point-to-point delivery of unassembled building blocks puts far too much of a burden on the rest of the organization to try to understand what a company's mainframe developers already know about the mainframe's proven applications and data. As a result, the service consumer has to interact with the mainframe developers to understand the underlying mainframe functionality. This eliminates the time-conserving benefit that probably drove the initial adoption of the bottom-up approach – and puts the lie to "instant SOA."

This method puts additional burden on the service consumer to orchestrate the interactions between the Web Service and the necessary data transformations, as well as the inputs and outputs in the business application. Further, when any of the underlying mainframe Web Service components change, the consumers have to be alerted as to how and what to change in relation to their specific uses of the service. With a back-to-the-drawing-board reaction every time there's a component change, the benefit – and therefore the likelihood – of reuse is lost, and along with it, much of the advantage of implementing SOA in the first place.

Further, companies that embrace this approach to get ahead of the game in delivering mainframe SOA frequently find that it has the opposite effect. They may end up with dozens or even hundreds of mainframe Web Services that have been developed in complete isolation from the SOA decisions that are typically made by centralized groups outside of the mainframe development organization. Services delivered in advance of such planning have no opportunity to take advantage of corporate direction. What's certain about a strategic undertaking like SOA is that all the elements must arise from and align with a centralized policy, so to avoid throw-away work, it's best to understand the big picture before you pull the trigger.

### A "Top-down" Approach Is the Real Fast-Track

In the top-down approach, by contrast, business processes drive the development of the Web Service. Typically, the mainframe developer works in advance with the user who needs the service, identifying the consumer's functional and data requirements, and negotiating a contract to deliver what they need. Based on that contract then the mainframe developer maps the mainframe components to the service requirements, and proceeds to develop an integrated business service that automates all the steps and operations required to fulfill those requirements.

While it's true that this approach requires a time commitment in the form of upfront communication between the end user and the mainframe developer, it's a much more efficient, healthy, and beneficial interaction than the post-service delivery complaint-and-rescue interactions inherent in the bottom-up approach. In addition, it forges a social partnership between mainframe and non-mainframe constituents that serves as a solid foundation for building a successful SOA, ensuring that the components of the architecture are optimally designed and sized to promote maximum reuse and efficiency. It's this service optimization that leads to the fastest returns on SOA investments in terms of cost savings, operational efficiencies, reduced overhead, and strategic advantage.

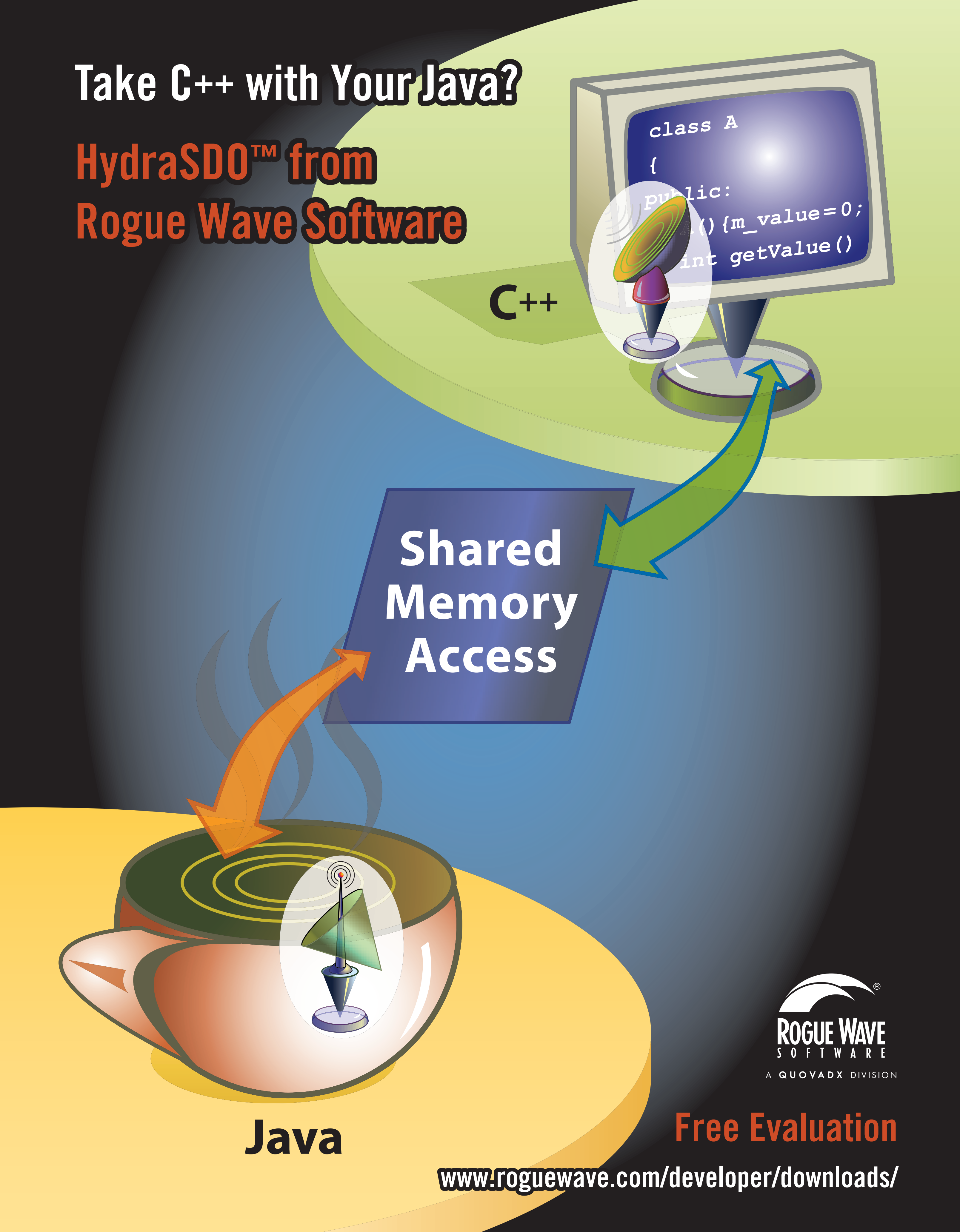
If you embrace the contract-first approach as a best practice for successful SOA implementation, there are some foundational elements that must be addressed.

### Don't Sweat the Small Stuff

As we've established, true service optimization can only come from a thorough understanding of the service definition in terms of the business process. Developing that level of understanding on the part of the mainframe developer requires social interaction with the end user. To free up the

Take C++ with Your Java?

HydraSDO™ from
Rogue Wave Software



ROGUE WAVE SOFTWARE
A QUOVADX DIVISION

Free Evaluation

www.roguewave.com/developer/downloads/

**"An SOA best practice... is a business process-oriented approach that begins with social interaction between mainframe developers and service consumers.**

time for such interaction, mainframe developers will need service development tools that enable them to automate as much of the low-level development and delivery mechanics as possible. The payback on time spent understanding user requirements and negotiating effective service delivery contracts to maximize reuse will far exceed any return on time spent mastering unfamiliar Web Service integration mechanics such as SOAP, XML, WSDL, or HTTP protocols.

## Maximize Your Assets

It's also important to leverage the mainframe developer's comprehensive knowledge of mainframe resources – what applications are available, where they are, and what they do. It may be helpful to augment or focus that knowledge by taking advantage of technologies specifically designed to assess the full range of complex mainframe applications, and isolate particular business functions for exposure as services. This kind of assessment, mapping, and realignment of existing applications to embrace Web Services and SOAs fully can greatly facilitate the service development process, and ensure that all available mainframe resources are utilized for maximum value.

## Some Assembly Required

You'll find that the business services required by users will most often span multiple operational systems, incorporate a variety of transaction types and data formats, and even invoke external Web Services. Of course, the users won't be able to express their requirements in those terms, so an important factor in the contract-first approach will be the mainframe developer's ability to extrapolate from the users' requirements the cross-system orchestration that will be required to make the service work as advertised.

To minimize the learning curve and avoid manual processing, as well as to be certain that the orchestration of multi-step, multi-operation business services is comprehensive and accurate, mainframe developers should take advantage of development and implementation tools specifically designed to facilitate and automate the deployment of mainframe-based services across systems and platforms. The ideal environment for a quick learning curve would enable developers to graphically model the inputs and outputs and multi-step processes required to implement the published service. Further, once the service is defined and published, the implementation environment should automate the business service flow, ensuring that SOAP processing and WSDL discovery can be fully leveraged without requiring mainframe developers to acquire an in-depth detailed understanding of these technologies.

## Dance with the One Who Brought You

By definition, a robust business service that leverages mainframe assets in the SOA will likely combine a variety of mainframe technologies. For example, 3,270 applications may be combined with CICS and possibly IMS transactions and data in any number of formats, as well as incorporate external Web services. And the whole thing must also be able to integrate new functionality, such as data transformation and routing, easily. The most efficient leverage of resources will exploit native data access capabilities, processing information directly in its native environment, eliminating the need for middle-tier servers, and providing the flexibility to fully leverage mainframe processing power as appropriate in the SOA.

## A Contract for SOA Success

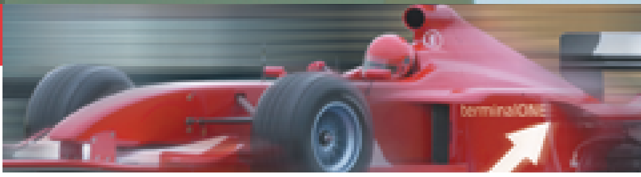
An uninformed rush to wrap and ship isolated chunks of mainframe functionality as Web Services – in a vacuum of organizational strategy and user requirements – will ultimately slow and negatively impact the success of your SOA, both in terms of customer satisfaction and reduced ROI. An SOA best practice, on the other hand, is a business process-oriented approach that begins with social interaction between mainframe developers and service consumers. Such a dialog produces a real understanding of the ultimate use and function of the service in terms of business benefits, and produces a contract between service provider and consumer that enables mainframe developers to effectively produce the complex, sophisticated business services that will ensure maximum usability and reuse profiles.

In addition, the contract-first approach opens the way for organizations to fully and strategically utilize their proven mainframe resources, including the in-depth knowledge of legacy applications, technologies, and data that resides with the mainframe developer. To realize this goal, it's important to provide tools to help the developer map and integrate these mainframe assets, to really understand what they have to draw from, how it works, and where it resides.

Finally, to free up the time required for the social interactions that enable the best practice of business-process development be sure your mainframe developers have the best tools for the job. Although it may seem counterintuitive, the more sophisticated tools that enable mainframe developers to do it the "right way" need not be any more complex to master than the simplistic tools of the "wrap-and-throw" approach. Look for tools that they can learn to use quickly and easily, such as graphical modeling tools. Minimize the time they have to spend on low-level mechanics by looking for tools that automate the process flow as much as possible, and orchestrate the execution of complex services across operational platforms without developer intervention. With the right approach and the right tools to implement it not only will the final result be a more strategically beneficial SOA implementation, getting there will actually be more efficient and cost effective. And after all, isn't that the definition of a best practice? ■

***About the Author***
*Robert Morris is senior vice president of marketing and strategy responsible for the planning, integration, and marketing of GT Software product solutions to the global market. Prior to GT Software, he held a variety of sales, marketing, and product management positions at KnowledgeWare, Forté Software, ClientSoft (now NEON systems), and Jacada. He has an extensive background in application development and integration including experience with CASE methodologies and distributed systems as well as midrange and mainframe environments.*
*rmorris@gtsoftware.com*

**_The future of EDI is B2B over IP_**



EDI sure had promise in the sixties – but its complexity, inflexibility, and the bottleneck of the VAN make it very costly today.

Simple to integrate, easy to manage, and blazingly fast, **terminalONE** is *the* B2B over IP platform. It intelligently transports, transforms, and routes all your data transactions.

We're about ebusiness adaptability: your business world changes fast – wouldn't it be nice if your data interchange adapted quickly and painlessly along with it?

terminalONE™

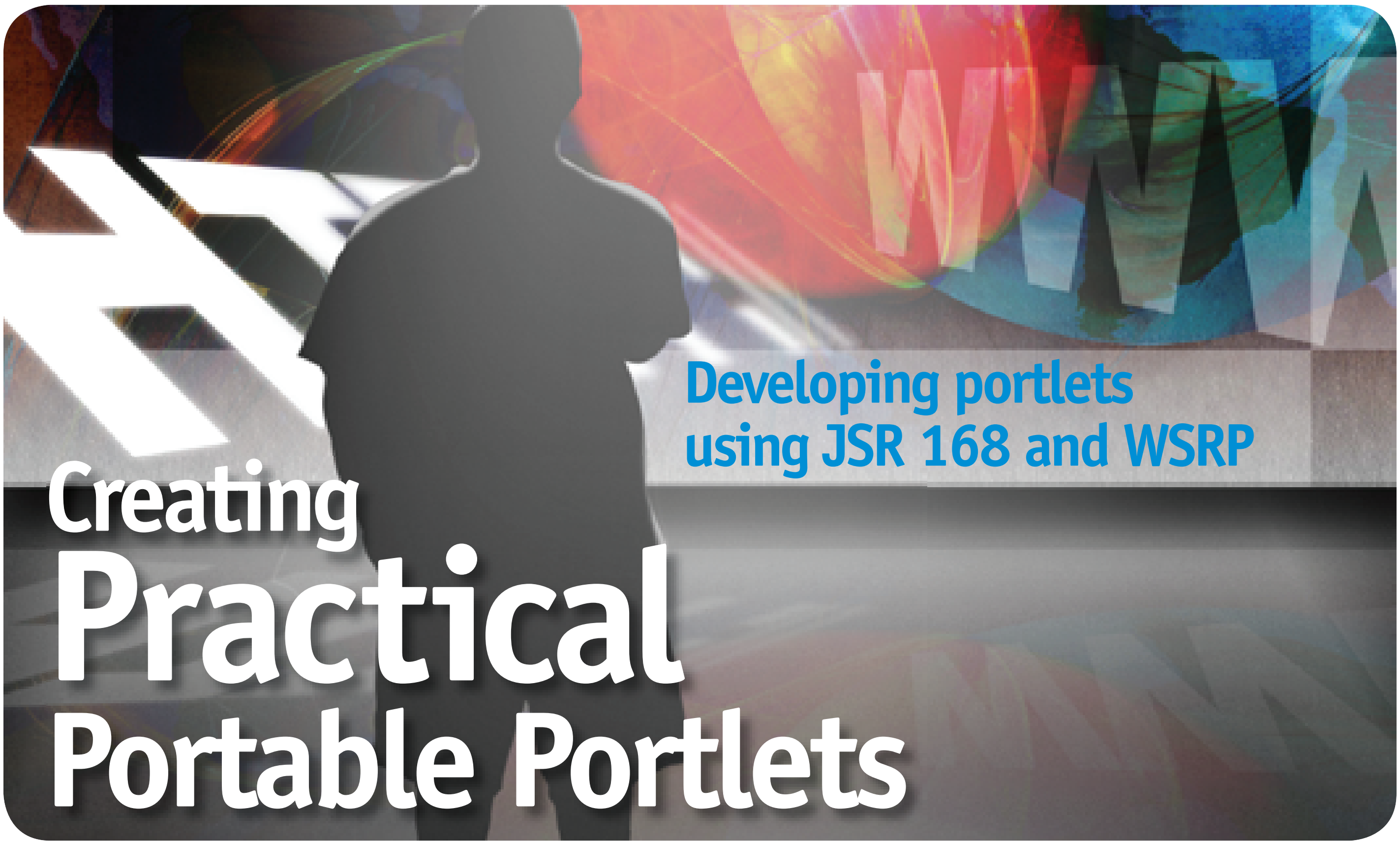
Secure, intelligent ebusiness transactions

*high-**velocity***
*high-**volume***
*high-**availability***

Take **terminalONE** for a test drive – ***today***
**www.xenos.com/VAN**

1 888 242 0695
1 905 763 4468
terminalONE@xenos.com

xenos



# Creating Practical Portable Portlets

## Developing portlets using JSR 168 and WSRP

WRITTEN BY **SABBU ALLAMARAJU AND ALEX TOUSSAINT**

JSR 168 has changed the playing field for portal development, letting vendors (and especially ISVs) develop portlets that various portals can consume. Likewise, WSRP has provided a standard so portals can consume portlets that reside remotely from the consuming portal. But questions remain. How real is the interoperability between portlets across different portal vendors? How does WSRP relate to other development patterns such as Struts and JSF? When should you use WSRP as opposed to JSR 168? As a developer developing for a portal customer, how do you know where to start?

This article will discuss two approaches to deploying portlets: the use of Java Specification Request (JSR) 168, which addresses the characteristics and specifications for a Java portlet, and the specification for Web Services for Remote Portlets (WSRP) from OASIS, one of the industry organizations defining Web standards.

Both of these standards seek to define portlets that are independent of the portal that they may be tasked to run in, allowing for the portability and interchangeability of these portal objects. The specifications are for both the developers of the portlet itself as well as the developers of the portals in which they run. In both cases, the standards define the areas of presentation, aggregation, security, and portlet lifecycle.

## The Role of Portlets

Portlets are small objects (in our case, Java objects) that provide specific services running in a portal system. Web portals are a kind of content management system, letting registered users access password-protected information from a number of different sources. The information owners who, in most cases, aren't the portal owners update the information in the portal, relieving the portal owners of that onerous task. And the user controls all this through a set of "preferences screens" so there's no need to learn complex programming to design the portal.

New sources of information are appearing daily with an unbelievable range of content, enough to satisfy even the most selective users. The ability to include diverse content into a single viewable portal is dependent on both the portal developers and the content developers complying with a stringent set of portal and portlet standards. The most relevant standards from OASIS and the Java Community Process are critical to the portability and usability of these portlets and will be the basis of the discussion here.

## Portlet Type Selection

As an application developer you should examine the different types of portlets that are available in portal systems and decide which type is best suited to the task at hand. For example, Java page flow portlets are particularly useful if you're interfacing with Java Controls, leveraging a Struts-based infrastructure, and delivering rich navigation elements. A JSP portlet would be a good choice if you're converting an existing JSP page into a portlet. If you're concerned with portability across multiple portlet containers then you may want to use JSR 168-compliant Java portlets.

Below is a matrix to help you decide which implementation to use when building portlets:

| Portlet Types | Pros | Cons |
|---|---|---|
| **JSP or HTML-based portlets** | Simple to implement and deploy<br><br>Provides basic functionality without a lot of complexity | Business logic and presentation layer can get combined in JSPs<br><br>Not well suited to advanced portlet navigation |
| **JSR 168-based portlets** | Accommodates portability for portlets across platforms<br><br>Doesn't require portal server-specific JSP tags<br><br>Behavior is similar to a servlet | Doesn't leverage advanced portlet features<br><br>Requires a deeper understanding of the J2EE programming model |
| **Java Page Flow-based portlets** | Lets you separate the user interface code from navigation control and other business logic<br><br>Both simple and advanced portlet navigation can be modeled<br><br>Lets you quickly leverage Java Controls, Web Services, and business processes<br><br>Provides a visual environment to build rich applications based on Struts | Advanced Pageflow features are unnecessary for static or simple one-view portlets |

## Java Portlet Environment

A portlet is a Java technology-based Web component, managed by a portlet container that processes requests and generates dynamic content. Portals use portlets as pluggable user interface components for a presentation layer for information systems. A portlet container provides portlets with the required runtime environment, managing their lifecycle and the persistent storage of portlet preferences.

While portlets are different than servlets, they share many concepts. Each is based on Java technology-based Web components, runs in a specialized container to manage their operations and lifecycles, and generates dynamic content for the user. Portlets differ from servlets in only generating markup fragments, not complete documents. They aren't bound to a URL and have more defined request handling, action requests, and render requests than servlets.

Portlets also have extra functionality that's not inherent to servlets. Portlets can access and store the persistent configuration and customization data specified by the user, have access to the user's profile information, and show hyperlinks on the display ultimately rendered.

A portlet container is, in general, an extension of the servlet container. The Portlet API v1.0 is based on the Java 2 Platform, Enterprise Edition 1.3. Portlet containers and portlets meet the requirements for executing in a J2EE environment as described in the J2EE specification.

The Portlet container must use the same classloader that the servlet container uses for the Web application resources to load the portlets and related resources in the portlet application. The portlet container is responsible for informing portlets of user roles, but the portlet container doesn't deal with user authentication.

## Making Portlets Remote

Web Services for Remote Portlets (WSRP) lets you decouple your portlet applications from portals. This decoupling can offer significant benefits for managing large portal deployments. Instead of bundling all your portlets with the portal in a single application, you can choose to deploy your portlets in individual portlet applications and let the portal consume those portlets using WSRP. For most large portal development projects, this decoupling eases team development, upgrades, and administration.

The best way to understand WSRP is to compare it to HTTP. The most typical HTTP application is viewing and interacting with a remote UI (for example, Web applications) via Web browsers. Using HTTP, browsers can talk to remote HTTP servers to get markup (for example, HTML), and post data (for example, by submitting a form). WSRP is a similar protocol between two applications, one application (the Consumer) acting as a client of another application (the Producer) for getting UI markup and submitting user actions. The Producer hosts the UI and Consumers use the WSRP protocol to aggregate the UI and interact with it.

Unlike browsers, Consumers are more sophisticated because they can aggregate several UI components into a single page, and they offer features like personalization, customization, and security. Consumers also deal with markup fragments, not complete documents. Consumers get these markup fragments from different Producers and combine them into a single page by applying Consumer-specific page layouts, styles, and so on.

The WSRP protocol defines a set of Web Services that WSRP Producers implement. WSRP Consumers can send messages to these Web Services to view and interact with the UI. The WSRP protocol translates the typical browser-server interaction protocol into a protocol that's suitable for applications (Consumers) to act as clients for applications (Producers) that host the UI.

It's important to understand that WSRP Web Services are synchronous and UI-oriented. Unlike business logic-oriented or data-oriented Web Services, UI-oriented Web Services offer coarser-grained application reuse and are more resilient to change.

Typically, portal systems will include both the Producer and Consumer components. The Portal Producer is a container that can host portlets. This is where application code (page flows, backing files, other portlet classes, controls, EJBs, and so on), user interfaces (JSPs and other resources), and the data used by the portlets reside. The Producer is designed so that you can convert any existing Web application into a WSRP Producer with minimal changes at deployment. Once a Web application is enabled as a Producer, it can start to offer portlets available in the Web application via WSRP interfaces.

To consume a remote portlet, the user typically starts with the location of the WSDL of the Producer, adds the Producer metadata to the Consumer, and then creates a remote portlet (also known as a proxy portlet). When such a portlet is added to a portal or a desktop, the WSRP protocol is typically used to present the portlet to portal users.

## The Design of Systems Using Remote Portlets

To illustrate how a WSRP Consumer can interact with remote portlets, let's start with a simple page flow portlet.

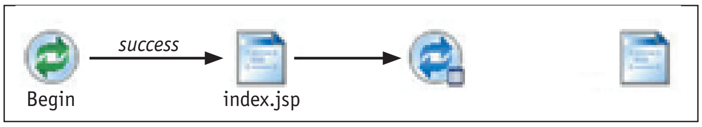


**Figure 1:** A search page flow

This page flow collects user input in a form, does a search based on the user input, and displays the search results. Here's a form presented by the index.jsp page:

### *Example 1 index.jsp*

```
<netui:form action="search" method="post">
 <table>
  <tr valign="top">
   <td>First Name:</td>
   <td><netui:textBox dataSource="{actionForm.firstName}" /></td>
  </tr>
  <tr valign="top">
   <td>Last Name:</td>
   <td><netui:textBox dataSource="{actionForm.lastName}" /></td>
  </tr>
 </table>
 <netui:button value="search" type="submit" />
</netui:form>
```

Let's assume that you created this page flow at /Search/SearchController.jpf in the Producer Web application, and created a .portlet file for this page flow. Here's a sample portlet file:

### *Example 2 Sample portlet file*

```
<portal:root xmlns:netuix="http://www.bea.com/servers/netuix/xsd/controls/ne-
tuix/1.0.0"
  xmlns:portal="http://www.bea.com/servers/netuix/xsd/portal/support/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
  "http://www.bea.com/servers/netuix/xsd/portal/support/1.0.0 portal-support-
1_0_0.xsd">
  <netuix:portlet definitionLabel="search" title="Search">
    <netuix:titlebar>
      <netuix:maximize/>
      <netuix:minimize/>
    </netuix:titlebar>
    <netuix:content>
      <netuix:pageflowContent contentUri="/Search/SearchController.jpf"/>
    </netuix:content>
  </netuix:portlet>
</portal:root>
```

In the WSRP protocol, Consumers use what is called a portletHandle to refer to a portlet on a Producer. The Producer is responsible for assigning this value. When a portal Producer exposes any portlet to Consumers, it uses the portlet's definitionLabel as the portletHandle.

After creating the remote portlet, you can change its attributes like caching, definitionLabel, title, error page, and so on. You can also add a backing file to remote portlets on the Consumer side.

## Displaying a Remote Portlet

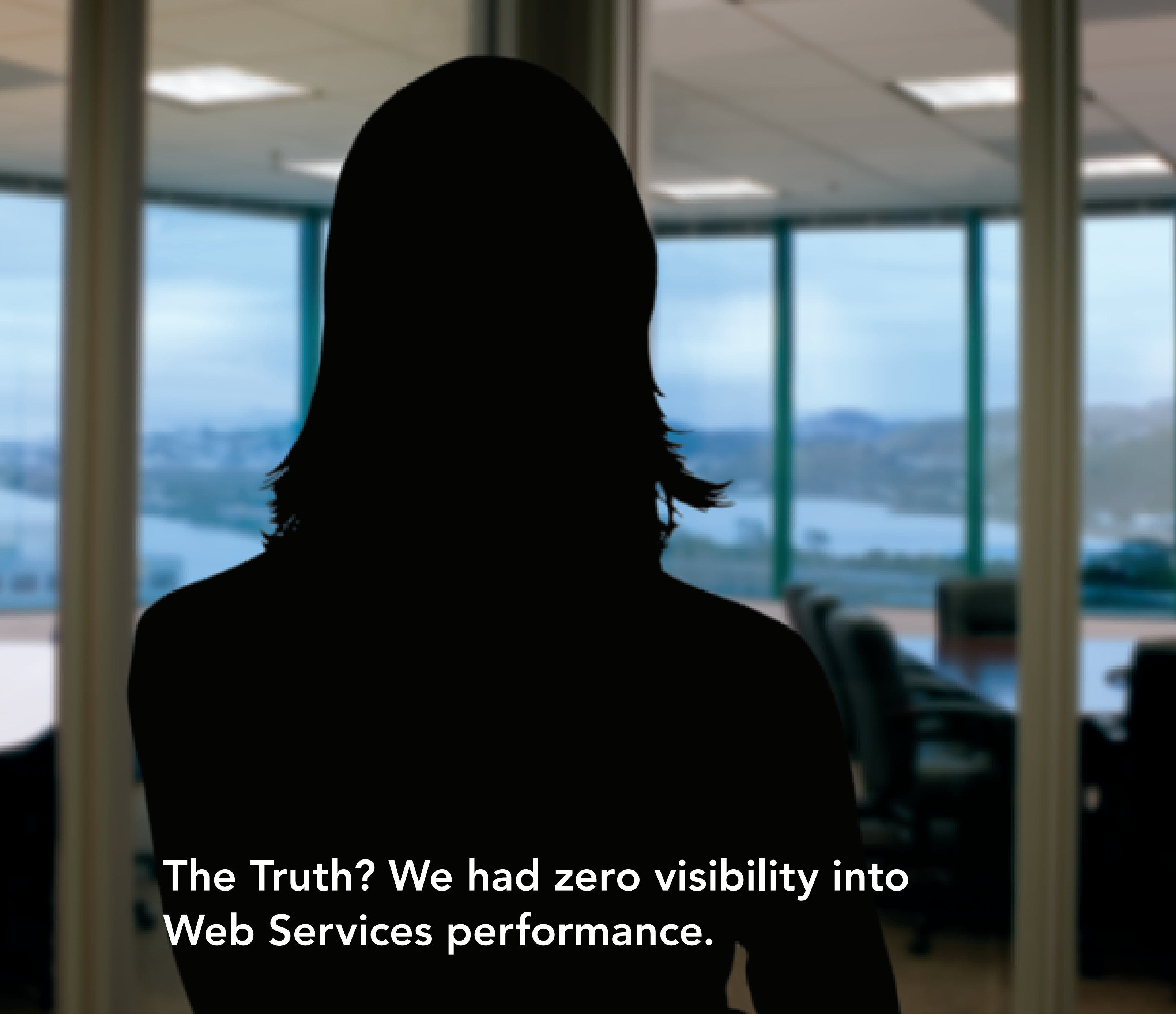
The portal uses WSRP to get the markup (in this case, the HTML generated for this portlet) from the Producer. When the portal finds a remote portlet in a page, it does the following:

- Sends a getMarkup message to the Producer and gets a response from the Producer
- Collects the markup from the Producer's response
- Aggregates the markup into the portal page

If the remote portlet has a backing file, the portal will call its methods in the appropriate order. Here's a typical getMarkup request that a Consumer sends to a Producer:

### *Example 3 getMarkup request*

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <urn:getMarkup xmlns:urn="urn:oasis:names:tc:wsrp:v1:types">
         <urn:registrationContext xsi:nil="true"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
         <urn:portletContext>
            <urn:portletHandle>search</urn:portletHandle>
         </urn:portletContext>
         <urn:runtimeContext>
            <urn:userAuthentication>wsrp:none</urn:userAuthentication>
            <urn:portletInstanceKey>search_1</urn:portletInstanceKey>
            <urn:namespacePrefix>search_1</urn:namespacePrefix>
            <urn:templates>
               <!-- Snip -->
            </urn:templates>
         <urn:userContext xsi:nil="true"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
         <urn:markupParams>
            <urn:secureClientCommunication>false</urn:secureClientCommunication>
            <urn:locales>en-US</urn:locales>
            <urn:mimeTypes>text/html</urn:mimeTypes>
            <urn:mimeTypes>*/*</urn:mimeTypes>
            <urn:mode>wsrp:view</urn:mode>
            <urn:windowState>wsrp:normal</urn:windowState>
            <urn:clientData>
               <urn:userAgent>
               Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1;
                  .NET CLR 1.1.4322; FDM)
               </urn:userAgent>
```


The Truth? We had zero visibility into Web Services performance.

**Wily can handle the truth.** We've helped hundreds of organizations around the world manage critical Web Services and proactively identify problems before customers are affected. Using a single solution, you gain control over both your application infrastructure and the customer experience. To learn more about how Wily can help you achieve customer success with Web Services, visit **truth.wilytech.com**.

**Get Wily.™**

Enterprise Application Management

wily™ technology

```
            </urn:clientData>
            <urn:markupCharacterSets>UTF-8</urn:markupCharacterSets>
        </urn:markupParams>
    </urn:getMarkup>
</soapenv:Body>
```

Let's look at the lines highlighted in bold in the above request:

- As mentioned above, the Consumer supplies the portletHandle assigned by the Producer.
- The portletInstanceKey and namespacePrefix are based on the value of the instanceLabel of the remote portlet. This is assigned by the user creating the remote portlet.
- The value of the locales element is based on the language requested by the browser and corresponds to the Accept-Language HTTP request header received from the user's browser. If the browser sends multiple values for this header, the portal Consumer sends all those values in the same order to the Producer.
- The values of the mimeTypes element is based on the content type set on the portal response, followed by the values of the Accept HTTP request header received from the user's browser.
- The mode and windowState elements correspond to the window mode and window state of the remote portlet.

***Note:*** *In the WSRP protocol, the Consumer keeps track of the mode and window state of the portlet.*

Now let's look at what the portal Producer must do when it gets a getMarkup request from a Consumer:

- Based on the portletHandle, the Producer identifies the corresponding portlet.
- Based on the value of the mode element, the Producer identifies that this request should invoke the begin action of the page flow specified by the pageFlowContent element in the portlet file.
- The Producer invokes the begin action of the page flow and includes the beginning page (which is index.jsp in this example).
- The Producer then collects the response and creates the SOAP response message.

Here's a sample response from the Producer.

### *Example 4 Sample response*

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Body>
        <urn:getMarkupResponse xmlns:urn="urn:oasis:names:tc:wsrp:v1:types">
            <urn:markupContext>
                <urn:mimeType>text/html; charset=UTF-8</urn:mimeType>
                <urn:markupString><![CDATA[
 <form name="searchForm" action="http://localhost:7001/consumer/test.portal?_
nfpb=true
 &_windowLabel=search_1_1&_pageLabel=test_page_2&wsrp-urlType=blockingAction&wsrp-
url=
 &wsrp-requiresRewrite=&wsrp-navigationalState=&wsrp-interactionState=
 _action%3D%252FSearch%252Fsearch&wsrp-mode=&wsrp-windowState=" method="post">
     <table>
         <tr valign="top">
             <td>First Name:</td>
             <td><input type="text" name="search_1{actionForm.firstName}"
value=""></td>
         </tr>
         <tr valign="top">
             <td>Last Name:</td>
             <td><input type="text" name="search_1{actionForm.lastName}"
value=""></td>
         </tr>
     </table>
     <br/>
     <input type="submit" value="search">
</form>]]></urn:markupString>
                <urn:locale>en-US</urn:locale>
                <urn:requiresUrlRewriting>false</urn:requiresUrlRewriting>
            </urn:markupContext>
            <urn:sessionContext>
                <urn:sessionID>
                B9Ml78JJyZNrMbzKnPxfyXZj511LL420BfKZGmLssNG02DbSJm3y!-1979539005
                </urn:sessionID>
                <urn:expires>3600</urn:expires>

            </urn:sessionContext>
        </urn:getMarkupResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Let's look again at the lines highlighted in bold in this response. If the portlet's markup is larger than 8k, you can improve performance by configuring the Producer to return the markup as a MIME attachment.

- The mimeType element indicates that the Producer rendered the portlet to generate text/html markup with UTF-8 character encoding. The portlet could influence this value using the JSP page directive for Content-Type, or the setContentType() method on the HttpServletResponse.
- The markupString element includes the response generated by the portlet. In our example, this is the HTML generated from the index.jsp page of the search page flow. To invoke portlets, the Producer uses custom request/response wrappers. These wrappers mimic a Web container environment although the incoming request is a SOAP request and not an HTTP request.
- The form's action element refers to the Consumer's portal. This is done by using the URL templates sent by the Consumer.
- The Producer rewrote the names of the form's input controls using the Consumer-supplied namespacePrefix element. This was done to ensure that the names don't collide with the name attributes of other HTML elements present in the portal's page and those present in the action target of the HTML form.
- The sessionContext element includes the ID of the HTTP session created by the Producer or the portlet. In our example, the page flow runtime created an HTTP session while executing the page flow's begin action. We'll discuss session management in more detail later in this article.

After getting a response to the getMarkup request, the portal Consumer extracts the markupString and writes it to the portal's HTTP response, making the portlet visible to the end user.

It's important to realize how accessing request parameters affects portability. When you deploy this portlet as a local portlet, the portlet can access the request parameters from the portal's request and request attributes set by other portlets on the same page. If you implement a portlet to depend on such request parameters and attributes, the portlet may not function correctly in a WSRP environment. Note that in a WSRP environment, the portlet is remote, and the HTTP request received by the portlet (on the Producer) isn't the same as the one received by the portal (on the Consumer).

If your use cases depend on such parameters/attributes, you can use the Custom Data Transport APIs to send that data explicitly from the Consumer to the Producer.

## Submitting a Form

Now let's see what happens when a user fills out and submits the form in the search portlet. Since the portlet is remote, the Consumer does the following to communicate the user's request to the Producer:

- The Consumer extracts all parameters from the incoming HTTP request. These parameters include all the query parameters and form parameters.
- The Consumer sends a performBlockingInteraction request to the Producer.

The interactionParams element encapsulates the user's request for the portlet and contains the following:

- A portletStateChange element indicates whether the portlet or the Producer can change the portlet's persistent state. In some portal Producers, this value indicates whether the portlet is allowed to change the portlet's preferences. I'll discuss using portlets preferences with remote portlets in greater detail later in this article.
- The interactionState element contains the state of the user interaction. For page flow portlets, the portal Producer uses this element to encapsulate the page flow action that must be executed. If you refer back to the getMarkupResponse message, you'll notice that the portal Producer inserted this data in the form's action URL.
- The formParameters elements include the form parameters submitted by the user.

When the portal Producer gets this request, the producer takes the following steps:

- Based on the portletHandle, the Producer identifies the corresponding portlet.
- The Producer decodes the interactionState element to find out the page flow action to be executed. In this case, it's the search action.
- The Producer creates request/response wrappers again to mimic a Web container environment and invokes the page flow's action. The Page Flow can now access the form parameters from this HTTP request.
- After invoking the page flow's action, the Producer gets the value of the next JSP to be displayed. In this case, it's the results.jsp.

At this time, as the WSRP protocol outlines, the Producer has two choices:

- Create a performBlockingInteractionResponse and return.
- Invoke the portlet's results.jsp and include the markup in the performBlockingResponse.

In the latter case, the Producer includes a MarkupContext element with the portlet's markup.

It's important to understand these two choices because portlets deployed in a local portal don't necessarily see the difference. To understand why WSRP specifies two requests with regard to getMarkup and performBlockingInteraction and what these requests mean to portlets, let's look into the above performBlockingInteractionResponse.

This response has a navigationalState element whose value includes the state of the portlet after processing the performBlockingInteraction request with the user's input. The actual value of this element varies from Producer to Producer. For page flow portlets, the portal Producer uses this element to include the action that was executed, the form parameters that were used for the action, and the JSP that must be shown to the user.

Using this navigationalState, the Producer can render the portlet at any time in the future. For example, let's assume that the portal page has another portlet. After viewing the search results, the user can start working with the second portlet. While the user is working with the second portlet, as long as that second portlet isn't maximized, the user expects that portal to display the search results in the search portlet. To present the search results, the Consumer must be able to ask the Producer to render the portlet with the same content. That is, the Producer must be able to render a portlet without an interaction any number of times and at any time. This is "interaction-free rendering" and portlets must be prepared for it.

To allow interaction-free rendering, WSRP defines the following protocol:

- Consumers use the performBlockingInteraction request to send user interactions to the Producer. The Producer includes the current state of the portlet as navigationalState in the response.
- Consumers use the getMarkup request to ask the Producer to return a portlet's markup. If the Producer previously returned navigationalState, the Consumer can supply it with the getMarkup request, so that the Producer can render the portlet with that state.

The WSRP 1.0 specification calls this model the two-step protocol. In the current example, this two-step protocol translates to the following:

- The Consumer sends a getMarkup request to the Producer without the navigationalState. The Producer then renders the index.jsp of the search portlet.
- When the user submits a form, the Consumer sends a performBlockingInteraction request to the Producer. In response, the Producer returns the next page to be displayed as navigationalState.
- Whenever the Consumer wants to display the portlet, the Consumer sends a getMarkup request with the current navigationalState. Using this state, the Producer renders the results.jsp of the page flow portlet.

**"The ability to include diverse content into a single viewable portal is dependent on both the portal developers and the content developers complying with a stringent set of portal and portlet standards.**

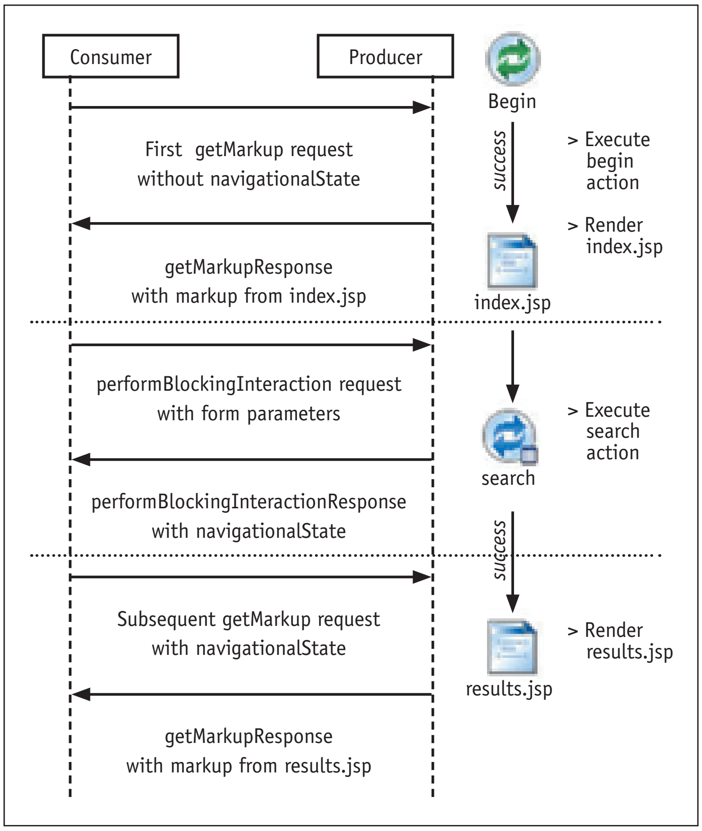

Consumer
Producer
Begin
First getMarkup request without navigationalState
success
> Execute begin action
> Render index.jsp
getMarkupResponse with markup from index.jsp
index.jsp
performBlockingInteraction request with form parameters
> Execute search action
search
performBlockingInteractionResponse with navigationalState
success
Subsequent getMarkup request with navigationalState
> Render results.jsp
results.jsp
getMarkupResponse with markup from results.jsp

**Figure 2:** Interaction-free rendering

Of these, the third step can happen any number of times without any user interaction.

The following sequence diagram summarizes this rendering model.

This diagram shows the request/response messages from the Consumer to the Produce and how the portal Producer invokes the search portlet. You might wonder why the Consumer should send another getMarkup request to the Producer to get the portlet's markup. As an optimization, WSRP lets Producers return markup in a performBlockingInteractionResponse message, so Consumers avoid the extra roundtrip. It's up to the Consumer to decide whether to use the markup that was returned or send a new getMarkup request. The portal Consumer and Producer support this optimization.

This interaction-free rendering isn't reserved only for remote portlets. The portal uses the same approach for local portlets as well. If you're familiar with the Java Portlet API, you'll notice that the javax.portlet.Portlet interface reflects this model. The render parameters in the Java Portlet API correspond to the navigational-State.

What does this interaction-free rendering means for portlets?

- The Consumer may invoke the portlet at anytime, and the portlet must be prepared to render its content when invoked by the Consumer.
- The Consumer may not render the portlet immediately after executing an action. So, the portlet can't assume that the action processing and subsequent rendering occur in the same request. Accordingly, if you add request attributes during action processing, those request attributes may not be available for your portlets during rendering.

Page flow runtime makes some exceptions for request attributes. When you make a request attribute during an action, the page flow runtime stores the attribute in the user's HTTP session and will make it available to the page flow while including the next page flow JSP. The same is true for Struts portlets but not for other kinds of portlets and the backing files used on those portlets.

The WSRP protocol puts certain restrictions on what can be done during a getMarkup request.

- During a getMarkup request, the Producer isn't allowed to change the portlet's persistent state. And in some cases this means that the portlet can't change its preferences. If you consider the fact that every portal page refresh could cause a getMarkup request, you'll quickly notice that changing portlet preferences every time a portlet is rendered can cause undesirable database changes. WSRP lets Producers change portlet's persistent state (that is, the portlet's preferences) under certain circumstances discussed below.
- During a getMarkup request, the Producer isn't allowed to change the portlet's mode or window state. This means that portlets can't use portal APIs to change a portlet's mode or window state. WSRP lets Producers make these changes during performBlockingInteraction.

These restrictions aren't specific to WSRP. The Java Portlet API has similar restrictions, which portals should enforce for all portlets whether local or remote.

## Summary

The goal of this article was to provide insight into some key concepts and implementation details of WSRP support in a portal system. When building portlets consider following the guidelines listed here. They will ensure that portlets will be more WSRP-friendly.

This is by no means an exhaustive coverage of WSRP or its implementation. Also, given the scope of this article there are a number of topics that must be left to individual research or a future article. The topics not covered here include:

- Session handling
- Clustering Producers
- The use of backing files
- Accessing portal APIs
- Portlet preferences and registration

*Please refer to the references below to learn more about WSRP.*

## *References*

- *WSRP 1.0 Specification*
- *WSRP 1.0 Primer. To get the latest version of this Primer, visit the WSRP Technical Committee home page.*
- *Java Portlet API (JSR168)*
- *Web Services Security*
- *Portlet Preferences, article*

"Everyone wants faster, better, cheaper. We thought they might appreciate smarter, too." Entering the integration market we had three advantages: we knew the costs and hassles of traditional integration solutions were limiting their adoption, we saw that a standards-based service oriented architecture would solve these problems, and we had the world's most scalable enterprise messaging server, SonicMQ®, as a core technology. We combined SonicMQ's performance and security with Web services, XML transformation, intelligent routing and a new distributed deployment and management infrastructure to develop the world's first Enterprise Services Bus, Sonic ESB™. With it businesses can easily integrate existing and future applications to create unprecedented business agility, and they can start today knowing they can scale to meet tomorrow's needs. We call it incremental integration. It's smarter. It's also faster, better and cheaper.

sonic
SOFTWARE®
CONNECT EVERYTHING. ACHIEVE ANYTHING.™

Gordon Van Huizen, Sonic Software

www.sonicsoftware.com

# Best Practices for Building SOA Applications

## Seven Steps to SOA Adoption
## *Part One: Publish and Orchestrate*

WRITTEN BY **DAVE SHAFFER**

Service Oriented Architecture (SOA) facilitates the development of applications as modular business services that can be easily integrated, secured, and administered. Benefits of an SOA approach include more-rapid development, decreased maintenance and change management costs, and improved business visibility. However, achieving these benefits isn't automatic — although many early adopters of SOA have been able to realize its promise fully, others have struggled to find the best architecture and design patterns for this approach.

The SOA model is about asynchronous, loosely coupled, stateless interactions through the use of standard component interfaces and architectures. However, it's often not obvious how this approach should be combined with traditional development practices and patterns such as model-view-controller (MVC) and synchronous and transactional Java or C/C++/C# coding. Likewise, the area of testing is one in which the flexibility of a loosely coupled architecture introduces new complexities (See Figure 1).

This article is the first of a two-part series that will outline the best practices and pitfalls that are starting to emerge for SOA, based on real-world customer implementation experiences. With this series, we hope to enable organizations to consider a few of these concerns earlier in the design process, thereby building on the successes of their predecessors and avoiding some of the mistakes.

## Challenges to SOA Adoption

Adopting a SOA involves more than just technology. Organizational issues play a major factor in the success of SOA initiatives. These factors include retraining, business and IT decision-making processes, governance and security. These are always issues whenever a new technology or architecture emerges; however, SOA has unique characteristics that amplify some of these complexities. For example, security has always been important, but a distributed SOA means that more information will be passing over a network as compared with a tightly coupled architecture.

It also means that teams and departments may become more interdependent on each other. If a group is building a risk analysis engine for a development project, that cost is going to be budgeted in that project. But the extra effort of making it a generic service, and publishing, maintaining, and securing it so that other applications can make effective use of it will incur additional development cost and must be explicitly supported by the organization as a whole. Incentives and oversight/governance need to be put in place at the broadest levels possible for a SOA to be more than just an implementation approach for individual projects.

At a purely technical level, developers must approach SOA projects with a different mindset than they have for tightly coupled implementations. Learning to identify the appropriate level of granularity for a service, determining what should be coded in languages like BPEL versus what should be in Java, and becoming comfortable with a new set of design patterns will take some time. Organizations embarking on their first SOA implementations should plan for a certain amount of exploration and refactoring in their schedules. Also, just because SOA and these new standards provide IT with

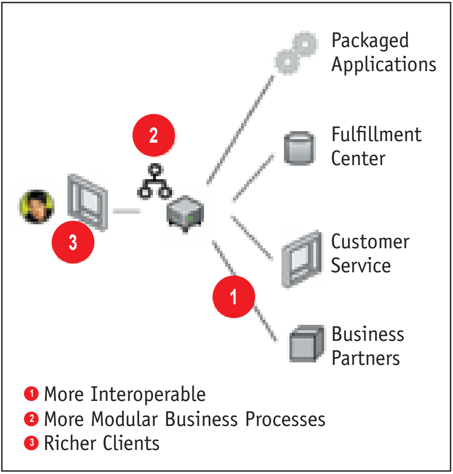


**Figure 1:** Service Oriented Architecture

a shiny new hammer doesn't mean that all projects are nails. We've often seen new tools, such as Web Services, asynchronous interfaces, and process languages like BPEL, over-generalized and used for problems they weren't suited for. For example, while we are strong proponents of BPEL, there are things that it isn't appropriate for, such as UI orchestration and highly computational business logic.

Three areas where we've seen companies encounter difficulties when adopting SOA are interoperability, testing, and performance. The nature of the service-oriented world is that it makes interoperability more difficult and more important than it is with traditional three-tier architectures. A major value of a loosely coupled standard service interface model like WSDL is that clients of a service don't need to know what technology is used to implement the service (and vice versa). But this makes for a combinational explosion when testing for interoperability. Similarly, testing SOA applications that interact with many external services is a significant challenge. Troubleshooting and performance tuning are also more complex due to the many different layers that may be involved in a single process or application.

However, all is not lost. If you haven't been scared away from SOA yet, we'll explain how some of the best practices that are starting to emerge for SOA adoption can help address these challenges. These best practices come from working closely with many customers in their initial SOA implementations and have been learned as much from problems and errors ("worst practices") as from successes.

## Seven Steps to SOA Adoption

We see the following as the key steps to effective SOA adoption:

1) Create a portfolio of services
2) Define connectivity and messaging interfaces
3) Process orchestration, workflow, and rules
4) Rich user interfaces
5) Business activity monitoring
6) Security and management
7) Performance and scalability

We recommend that you not approach these steps in a sequential order. For example, it's very risky to consider performance only at the end of a project, at which point design decisions that affect scalability can be very difficult to change. The best way to use these steps is to apply them all to a thin slice of a project and then iterate through rapidly expanding prototypes as additional functionality is added.

In this article we'll describe the first three steps. Next month we'll look at the second set of steps in detail as well as some "worst practices" that vividly illustrate some of the potholes to avoid on the road to SOA.

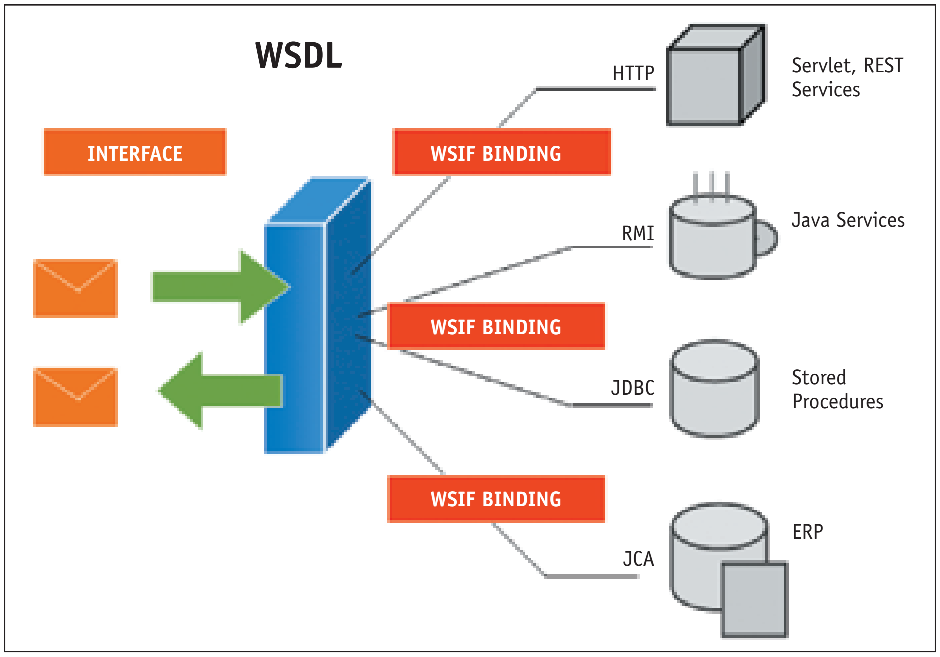


**Figure 2:** WSDL and WSIF bindings

## Portfolio of Services

It's important to lock down the interfaces to services and backend systems early in the development process. WSDL and XML schema will be your best friends when interfaces remain relatively stable and your worst enemies if they're constantly changing. Although a service-oriented approach provides the flexibility and agility that makes applications easier to change, the simplest changes will be when only the implementation of a service changes, while the interface to the service remains constant. Conversely, when the interface itself changes, the impact of that change will propagate broadly (and perhaps unpredictably) throughout your infrastructure. As a result, using coarse-grained, document-based interfaces for services will provide the most flexibility.

As mentioned, interoperability is both critical and non-trivial in a service-oriented world. We find that a governance policy requiring that services be WS-I Basic Profile–compliant (http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html) is a best practice. For example, key Basic Profile requirements include avoiding RPC encoding and SOAP-encoded arrays that seriously restrict interoperability.

Finally, a UDDI registry and taxonomy for organizing services is important; however, the first step is to lock down the interfaces, regardless of whether they're kept in a registry. But over time, as organizations create more services, a common directory becomes increasingly more important.

## Connectivity and Messaging

The next step is to determine what protocols will be used for connecting to services. SOAP is the canonical Web Services protocol, but it's not supported yet by many current backend systems and may not be appropriate for all services. Other options include packaged adapters and flexible binding frameworks such as the Web Services Invocation Framework (WSIF), an Open Source framework owned by Apache (http://ws.apache.org/wsif/).

WSIF enables service interfaces to be described by a standard WSDL and then leverages a binding for the native protocol of the backend system. This provides a "best of both worlds" approach with the loosely coupled architecture of Web Services but also the performance and transactionality of native system protocols.

WSIF bindings are now available for many protocols including Java, EJB, XML over HTTP (for a REST-style interface),

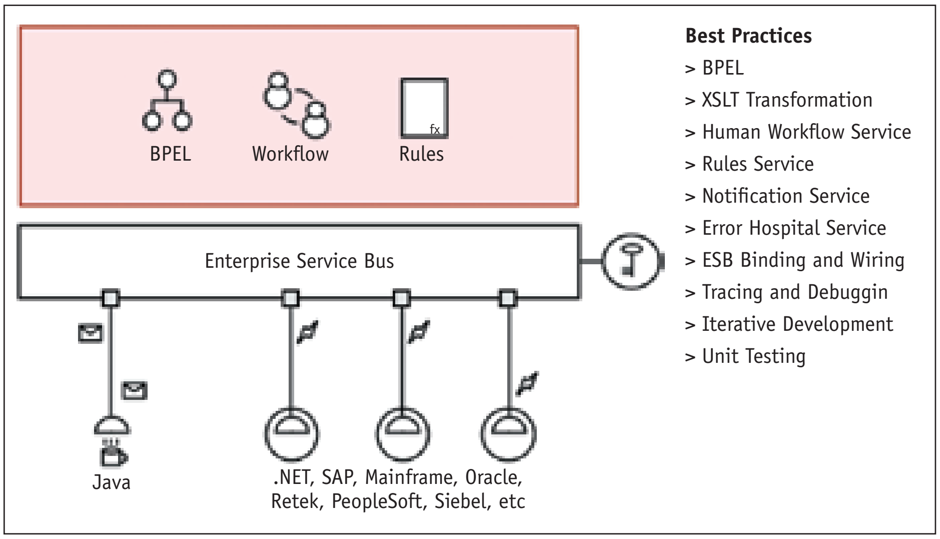


**Figure 3:** Process orchestration, workflow, and rules

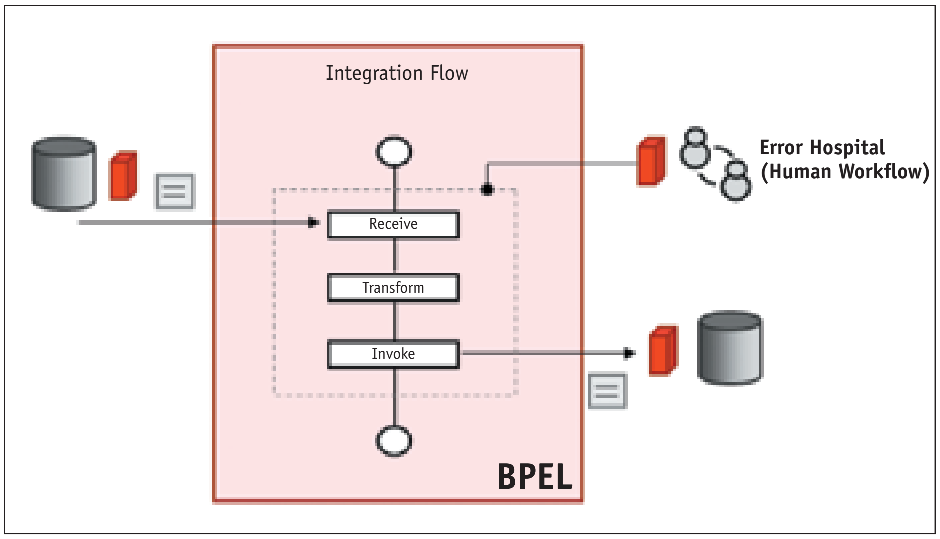


**Figure 4:** Error hospital pattern

JMS, and JCA. As with most design choices, there are tradeoffs involved in this approach. A WSDL with SOAP binding will incur a performance penalty and lose any transaction monitoring capabilities that may be supported natively by the backend system, but it provides maximum interoperability. Non-SOAP WSIF bindings can improve performance and provide two-phase commit transaction support, but the tradeoff is the loss of some interoperability. Tools like Oracle Application Server and Axis from Apache can support WSDLs with both SOAP and native protocol WSIF bindings, so the interface can be as flexible as possible. We anticipate that in the future, the WSIF approach will evolve into a next-generation implementation with the service component architecture (SCA) (http://otn.oracle.com/tech/webservices/standards/sca), but the concept remains the same.

Key factors to take into account when selecting connectivity protocols include requirements for performance, transactionality, scalability, quality of service, and interoperability. In the area of performance and scalability, it's critical that the final requirements be known and tested for as early in the design process as possible, because it's often only with a "performance POC" that the right decisions can be made early enough in the design and architecture process to avoid costly redesign later.

Service virtualization and logical naming become particularly important when a service's location may change or when levels of service availability, scalability, and security have to be changeable without modifying clients. In these cases, two approaches are possible — either use a dynamic registry lookup or a Web Services management technology as described later in this article. These approaches and options are typically supported by Enterprise Service Buses (ESBs) from the major technology vendors.

## Process Orchestration, Workflow, and Rules

Once you know what service interfaces and protocols you will use, the next step is to define the composite applications and business processes around those services. Here we believe very strongly in the Business Process Execution Language (BPEL) standard from OASIS. BPEL lets business processes be defined in a standard format, and it has gained tremendous market traction among both vendors and end users over the past several years. There are now thousands of mission-critical BPEL processes in production across hundreds of enterprises.

But there's life beyond BPEL. From its few remaining detractors, much has been made of the fact that BPEL doesn't include human workflow "support." However, BPEL doesn't say anything about any specific services at all, whether human workflow or systems. But it has very rich asynchronous service support. So we believe a clean approach for technology vendors and customers – while the standards expand to cover this area – is to use an external human workflow service that can be orchestrated by 100% standard BPEL processes. This makes human tasks and manual steps in a process look like any other asynchronous service and lets the BPEL processes remain standard while combining rich workflow capabilities with system integration.

Besides workflow, we recommend defining frequently changing business logic in external business rules. This can be done through any of a number of rules engines on the market today and incorporated into your BPEL processes and J2EE or .NET applications via Java or Web Services interfaces. Most process engines also include a tight coupling to rules engines and those that don't now probably will in the future. But even without specific vendor support, the effort to define an external business rule and call out to it will be well worth it when the rule has to change post-deployment. A loosely coupled interface provides critical flexibility to implement change without going through an entire develop-and-deploy cycle that can be both complex and risky in many production environments.

Another area where flexibility is critical is exception handling. We've seen many requirements for processes that need maximum performance but also rich exception handling. These can be conflicting goals because maximum performance is often achieved through simple processes with synchronous, tightly coupled interfaces, but rich exception handling may include asynchronous requirements and human workflow. A design pattern we've found useful for addressing these competing needs is the "error hospital." This pattern provides rich exception handling as a service. For maximum performance the processes are designed as short-lived synchronous processes under the assumption that all will go well. When issues are encountered, messages are published to an error hospital process (this could be a JMS queue or a BPEL process) that can then provide sophisticated exception management, perhaps with humans involved when needed.

The last best practice in the area of process orchestration that we want to discuss is testing. As mentioned, testing is both critically important and particularly complex for long-running, asynchronous processes that orchestrate many external services. Testing a process means not only simulating all the services involved, but also simulating all possible return values from all the services, including exceptions, timeouts, retries, and so on. We've seen organizations that spend three times as much time writing test cases as writing code for SOA projects — and these are the projects that have a good handle on a testing approach. We recommend leveraging a rich testing framework that can do more than just a black box simulation for things like BPEL processes. At Oracle, we've worked for several years on such a testing framework that we're now bundling with our BPEL Process Manager product and independent vendors like Parasoft are now coming out with testing tools that have built-in support for SOA standards like BPEL and WS-Addressing.

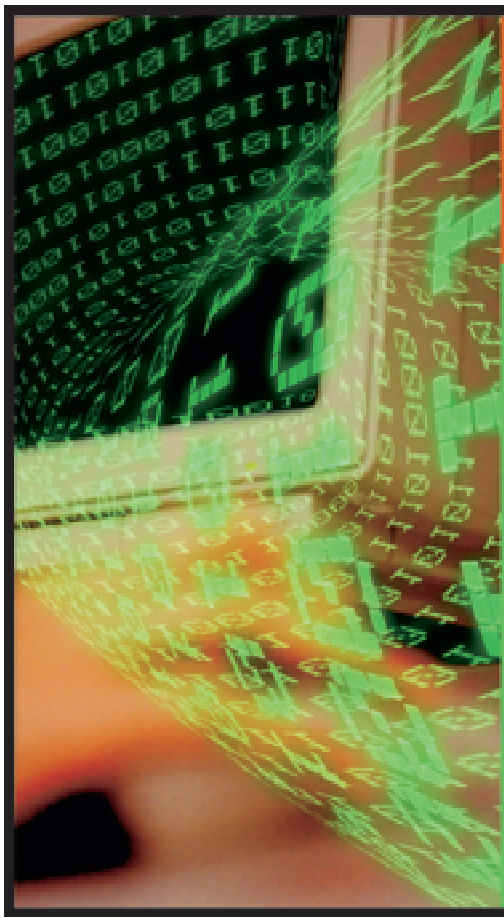
## Summary

Next month we'll look in detail at the best practices for the final four steps: rich user interfaces, business activity monitoring, security and management, and performance and scalability. In the meantime, we hope that the first part of this series has given you sufficient food for thought, if your organization is considering adopting SOA, about how to achieve its promise while minimizing risk and how to leverage the best practices that are emerging. These are non-trivial tasks and even a two-part article such as this won't get very deep below the surface. However, we hope that these topics will spur an ongoing discussion in the IT community about how to best achieve the promise of SOA. ■

**_About the Author_**

*Dave Shaffer is Sr. Director of Product Management at Oracle for the Oracle SOA Suite, overseeing BPEL, BAM, ESB and other products. Prior to Oracle, he has held consulting, product management and software development roles at a wide-range of technology companies including Collaxa, Apple Computer, NeXT Software and Integrated Computer Solutions.*

*david.shaffer@oracle.com*



# Oracle Business Activity Monitor

Building operational dashboards, and monitoring and alerting applications delivered via the Web

WRITTEN BY **BRIAN BARBASH**

Services Oriented Architectures (SOAs) and business collaboration technologies and platforms, often enabled by Web Services and orchestration constructs like BPEL, can be a tremendous business benefit. SOAs can provide the flexibility in enterprises to adapt to rapidly changing business conditions. Collaboration platforms enable tighter integration between trading partners both at a data and process level. When successfully implemented, these approaches and tools can garner significant improvements in efficiency, communication, service levels, and ultimately profits. However, quantifying the benefit can prove challenging.

As services proliferate and orchestration platforms automate increasing numbers of processes, manual oversight of low-level details becomes impractical and unnecessary. So identifying and addressing the issues require a more analytics-based approach than in the past.

In response to these challenges, many of the SOA and business collaboration platform vendors deliver some kind of management and analysis component providing insight into the transactional activities. Oracle is no exception with its Business Activity Monitor (BAM) solution. Oracle BAM is a platform for building operational dashboards, and monitoring and alerting applications delivered via the Web.

## BAM Architecture & Integration Capabilities

BAM is essentially a third-party listener to business communications both in and between services and applications. It's designed to collect low-level transactional data and deliver aggregate information to corporate operational dashboards. It may also be configured to send alerts of business exceptions to various audiences and devices.

Figure 1, from the Oracle BAMP documentation, illustrates the architecture of the system. Its components include:

- **Enterprise Link:** Provides an ETL and adaptor mechanism for BAM to get data from information sources including flat files, XML data, databases, and messaging queues
- **Active Data Cache:** The main store of active business information for BAM that monitors information, creates events in response to data changes, and publishes the events to the reporting and event engine
- **Report Cache:** Provides the working store for reporting information ultimately displayed to the user
- **Event Engine:** Monitors changes in data based on user-defined rules and executes the appropriate actions
- **Reports Server:** Transforms the raw data contained in the report cache to the display format defined by the users

The Enterprise Link and the Active Data Cache both provide integration touch points for BAM. The Active Data Cache exposes APIs in Java, C#, and Web Services allowing applications to insert, update, and delete data in the cache. The Enterprise Link component is used to get messages from JMS topics and transform them to the Active Data Cache using data flows configured in the system. Enterprise Link also provides the capacity to use data transforms to merge and correlate data from multiple sources into a single structure.

## Building Reports and Dashboards

***Development*** – The ActiveStudio component of the BAM platform serves as the IDE for creating reports, dashboards, and alerts for users. This is a Web-based UI that, along with all UI components of BAM, is designed to work with Internet Explorer.

Building reports in the IDE is a straightforward step-by-step process. After selecting the layout of the report, developers have access to several different data visualization tools including:

- **Lists:** Tabular displays of information, updating in real-time, that can be configured to display data at either a transactional or aggregate level
- **Charts:** Several charting options are presented including bar charts, line charts, and pie charts among other formats
- **Contextual displays:** These elements provide performance informa-

tion at a single glance and include objects such as arrows and gauges
- **Pivot/Crosstab/Excel objects:** Summary objects that allow for data manipulation either directly in an embedded Excel spreadsheet or similar objects
- **External objects:** OracleBAM can integrate external Web pages to the reports providing the flexibility to add additional context outside of the application boundaries to information reported

Each visualization object extracts information from data objects that have been defined in the BAM. As seen in the sample report created in Figure 2, the lower half of the screen shows the data configuration area for the line chart. Developers select the field by which to group transactional information and the values to display in the chart. Oracle BAM provides a choice of standard aggregate functions including Sum, Average, Minimum, Maximum, Count, Count Distinct, Calculation, and Percent of Total. The system is data type-aware so that only functions appropriate to the chart values are made available (e.g., Count and Count Distinct only for non-numeric chart values).

Once the data is defined for the visualization object, the data can be filtered to isolate only the information desired. The system also allows for filters to be based on prompts providing users the flexibility of changing information when viewing the reports online. Reports can also be filtered to display only a list of a configurable number of the top results of a given metric.

***Drilling and Report Linking*** – One of the key analytical elements of Oracle BAM is the ability to cross-link reports and drill through data. Using the example report above, one of the metrics is a pie chart that shows the breakdown of valid versus invalid orders. This pie chart can be configured so users can drill across to another report that details valid and invalid orders by company. This can be done through the following steps:

1. Create a detail report with a parameter for filtering information; in this case the filter will be the order status
2. Set the pie chart's drilling target to the newly created detail report
3. Map the order status of the pie chart to the parameter of the newly created detail report

Once configured, users can double-click a region of the pie chart, replacing it with the newly created report detailing the orders with the status of the pie chart region selected.

***Alerts*** – Service levels are a key part of an SOA. The concept of service levels can be extended to business metrics to help define the financial performance of processes in an organization. BAM can be configured to send alerts that optionally contain report instances to key personnel based on service-level thresholds. Thresholds can be defined against any of the data available to the reports. Messages may be sent via e-mail, the Active Messenger component of BAM, or based on how individual BAM users have configured their alert preferences.

To manage the performance implications of inspecting data passing through the system, constraints can be placed on the alerts to minimize the impact. Some capabilities include restricting the time and/or day on which alerts are sent and setting a minimum time between successive rule evaluations.

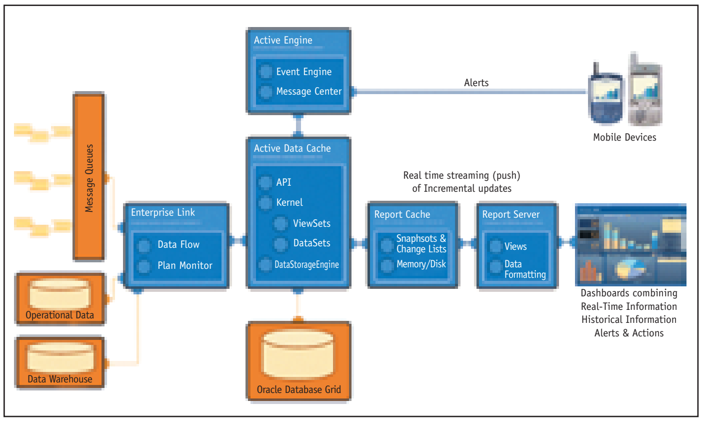


**Figure 1:** Oracle's BAM architecture

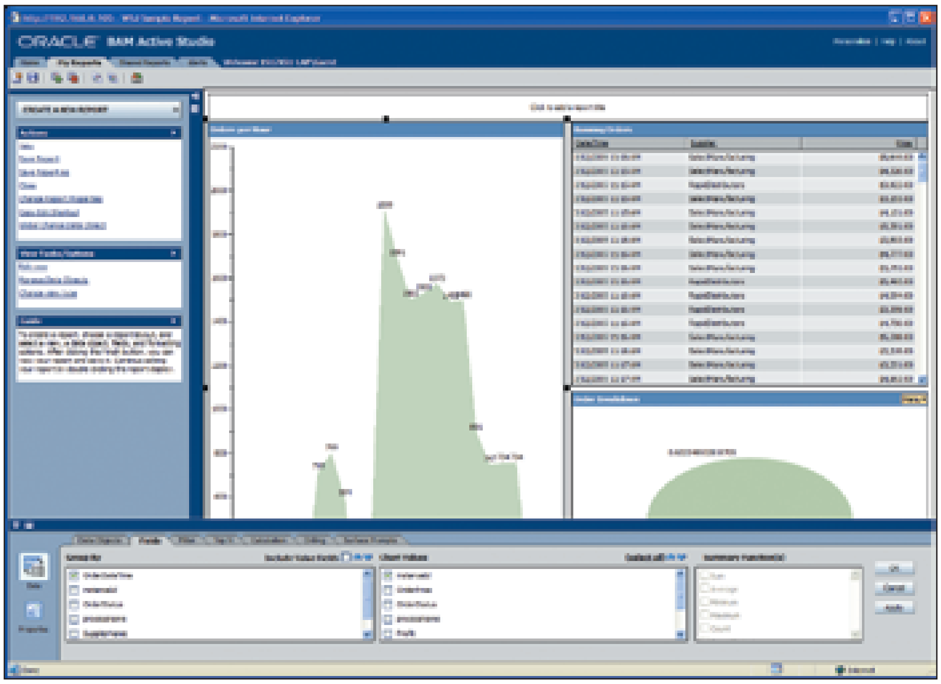


**Figure 2:** ActiveStudio IDE

## Summary

Modern SOAs and trading partner collaboration technologies present unique challenges with respect to business performance management. Oracle BAM provides an analytics-based solution to measuring and monitoring these systems. A Web-enabled real-time reporting and dashboard solution, BAM integrates into transactional processes providing insight into performance. For SOA management and performance analysis, Oracle BAM is a solid solution.

***About the Author***

*Brian R. Barbash is the product review editor for Web Services Journal. He is a senior consultant and technical architect for Envision Consulting, a unit of IMS Health, providing management consulting and systems integration that focuses on contracting, pricing, and account management in the pharmaceutical industry.*

*bbarbash@sys-con.com*

# Rich Internet Applications: AJAX,

www.AjaxWorldExpo.com

AjaxWorld™ CONFERENCE & EXPO

## SANTA CLARA SILICON VALLEY

**SYS-CON Events is proud to announce the first-ever AjaxWorld Conference & Expo 2006!**

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

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

### CALL FOR PAPERS NOW OPEN!

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

# Flash, Web 2.0 and Beyond...

REGISTER TODAY AND SAVE!

## October 3-4, 2006

**Santa Clara Convention Center**
Hyatt Regency Silicon Valley
Santa Clara, CA

**To Register**
Call 201-802-3020 or
Visit www.AjaxWorldExpo.com

## May 7-8, 2007

First International AjaxWorld Europe
**Amsterdam, Netherlands**

***"Over the two information-packed days, delegates will receive four days' worth of education!"***

**Early Bird***
**(Register Before August 31, 2006)** ........................................ $1,495**
See website or call for group discounts

**Special Discounts***
**(Register a Second Person)** ........................................ $1,395**
See website or call for group discounts

**(5 Delegates from same Company)** ........................................ $1,295/ea.**
See website or call for group discounts

**On-Demand Online Access**
**(Any Event)** ........................................ $695

*Golden Pass access includes Breakfast, Lunch and Coffee Breaks, Conference T-Shirt, Collectible Lap-Top Bag and Complete On-Demand Archives of sessions in 7 DVDs!

**OFFER SUBJECT TO CHANGE WITHOUT NOTICE, PLEASE SEE WEBSITE FOR UP-TO-DATE PRICING

**"It Was The Best AJAX Education Opportunity Anywhere in the World!"** —John Hamilton

## Topics Include...

**Themes:**
> *Improving Web-based Customer*
> *Interaction*
> *AJAX for the Enterprise*
> *RIA Best Practices*
> *Web 2.0 – Why Does It Matter?*
> *Emerging Standards*
> *Open Source RIA Libraries*
> *Leveraging Streaming Video*

**Technologies:**
> *AJAX*
> *The Flash Platform*
> *The Flex 2 Framework & Flex Builder 2*
> *Microsoft's approaches: ASP.NET, Atlas, XAML with Avalon*
> *JackBe, openLaszlo*
> *JavaServer Faces and AJAX*
> *Nexaweb*
> *TIBCO General Interface*

**Verticals:**
> *Education*
> *Transport*
> *Retail*
> *Entertainment*
> *Financial Sector*
> *Homeland Security*

**GROUP DISCOUNTS AVAILABLE:**
— 5 Delegates from Same Company — for only $995 (each)
— Register a Second Person — for only $1195

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

## Receive FREE WebCast Archives of Entire Conference!

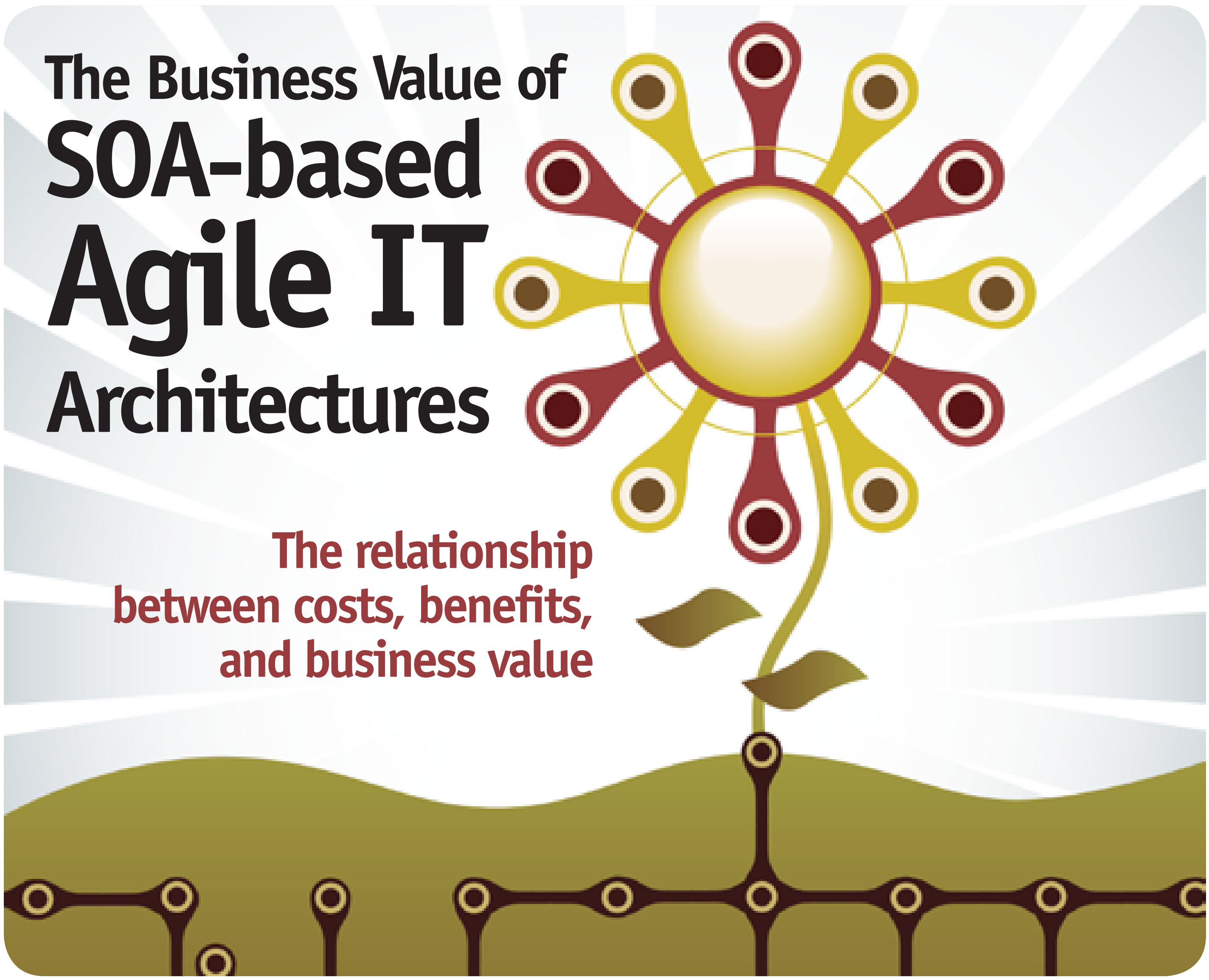
The best news for this year's conference delegates is that your "Golden Pass" registration now gives you full access to all conference sessions. We will mail you the complete content from all the conference sessions in seven convenient DVDs after the live event takes place.

**This on-demand archives set is sold separately for $995**

SYS-CON EVENTS
**For more great events visit www.EVENTS.SYS-CON.com**
VISIT WWW.AJAXWORLDEXPO.COM FOR THE MOST COMPLETE UP-TO-DATE INFORMATION

# The Business Value of SOA-based Agile IT Architectures

**The relationship between costs, benefits, and business value**



WRITTEN BY **JAI GANESH, NIRANJAN IYENGAR, AND SRIRAM ANAND**

Agile IT systems are systems that are malleable enough to address business uncertainties. Such systems can effectively respond to internal and external stimuli in a very short period of time. Flexible IT systems imply that the IT architecture underlying them is itself flexible and lends itself to incorporating changes in a dynamic fashion. Architectural approaches such as Service Oriented Architecture (SOA) are transforming the way IT systems are designed by bringing in a high degree of reuse and loose coupling of applications.

IT architectures include the technology, strategies, plans, and principles that guide an organization's new technology investments as well as manage the existing technology investments.

The objective behind having a good IT architecture is so an organization can operate with a high degree of flexibility while keeping the cost for the technology investments within justifiable limits.

IT architectures are derived from the business architecture and once the architecture is defined, it supports the business. Organizational flexibility is a key requirement supported by IT architectures. Organizational flexibility and agility is emerging as a key constituent of sustainable competitive advantage. The business processes supporting organizational activities have to be flexible enough to enable organizational flexibility. Business processes are dependent on the underlying IT systems for execution. These IT systems and architectures should be flexible enough to deal with uncertainties in an unstable environment. They have to respond rapidly to internal and external stimuli. Agility refers to an organization's ability to thrive in a continuously changing business environment. A flexible enterprise is qualified by the cost and the speed of response to changes in business models, business opportunities, and market conditions.

## The Business Value of SOA-based IT Architectures

The new enterprise architectural approach termed Service Oriented Architecture (SOA) envisages delivering IT functionality as services over a distributed network. The idea behind SOA is to treat business and IT functionality being delivered as a set of services, wherein the services are self-contained and independent of the context or state of other services. SOA-based systems can be seen as a collection of services having well-defined interfaces. In technical terms, SOA is an approach to loosely coupled, protocol independent, standards-based distributed computing where the IT resources are available on a network as services. The key idea differentiating SOA from earlier models of distributed computing lies in the notion of a service as the least common denominator. Here, a service refers to a piece of functionality, which is defined by a strict contract via a well-defined interface that is independent of any underlying implementation platform of the service. This kind of interface is loosely coupled in that a service is a standalone entity with no tight coupling to the underlying environment or to the other services. These characteristics are key to delivering the required flexibility for enterprises by adoption of SOA.

Service Oriented Architecture has three key constituents: service provider, service requestor, and service registry and three fundamental operations: publish, find, and bind. A service provider makes the service available and publishes the contract that describes its interface. It then registers the service with a service broker. A service requestor queries the service broker and finds a compatible service. The service broker gives the service requestor directions on where to find the service and its service contract. The service requestor uses the contract to bind the client to the service. The connection between these entities is loosely coupled offering the maximum decoupling between any two entities. For the loose coupling it is mandatory that the services be described in ample detail to be self-describing and self-contained. These factors lead to the natural question of how to measure the business value of SOA based IT architectures, which has a combination of unique characteristics, prominent among which include re-use, flexibility, and open standards.

## Cost and Benefit Dimensions

Our primary research revealed six cost dimensions and five benefit dimensions for the posited relationship between the costs, benefits, and business value. (See Figure 1.) The cost dimensions are related to the efforts required for an enterprise-wide IT effort such as SOA, while the benefits are potential rewards of the SOA deployment.

The cost dimension is comprised of variables such as *Organizational, Business Process, Technology, Servicification, Integration* and *People*. The benefit dimension is comprised of operational measures such as Re-use, Flexibility, Integration, Open Standards and People. Table 1 gives the definitions of these dimensions.

On the basis of our research and interviews with industry experts,

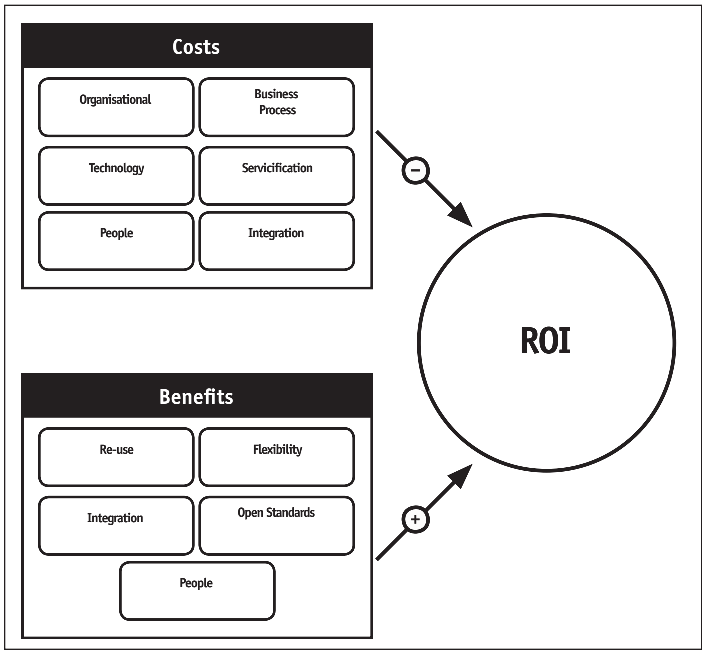


**Figure 1:** Posited relationship

| Dimension | Variable | Explanation |
|---|---|---|
| Cost | Organizational | Costs involved in obtaining organization-wide buy-in, identifying business metrics, business contexts, etc. |
| | Business Process | Costs involved in use case analysis, process modeling, etc. |
| | Technology | Costs involved in identifying architectural pain points, mapping systems to processes, developing future state architecture, application portfolio analysis, etc. |
| | Servicification | Costs involved in service identification, developing new services, service governance, etc. |
| | Integration | Costs involved in interface development and maintenance. |
| | People | Learning costs, training, etc. |
| Benefit | Re-use | Benefits from re-use of business models, business processes, applications, infrastructure, etc. |
| | Flexibility | Benefits from support for emerging scenarios, cost, time, and ease of introducing/modifying business processes, applications, IT infrastructure, etc. |
| | Integration | Reduced complexity and ease of integration (due to standard interfaces and skill-set requirements). |
| | Open Standards | Benefits from open standards. |
| | People | Benefits from IT personnel efficiency and standardized employee skill levels (products, tools, etc.). |

**Table 1:** Dimensions and Variables

we have identified the following propositions regarding the cost dimension:

**P1** Organisational, business process, technology, servicification, integration, and people costs are the most important cost variables influencing SOA architecture decisions

- **$P_{1A}$** Among organisational costs, *understanding business context* is the most important cost variable influencing SOA architecture decisions
- **$P_{1B}$** Under Business Process analysis costs, *identifying business processes that can be shared* is the most important cost variable influencing SOA architecture decisions
- **$P_{1C}$** Among technology costs, *application portfolio analysis* is the most important cost factor influencing SOA architecture decisions
- **$P_{1D}$** Among servicification costs, *determining the optimal level-of-service granularity* is the most important cost variable influencing SOA architecture decisions
- **$P_{1E}$** Among integration costs, *interface development* is the most important cost variable influencing SOA architecture decisions
- **$P_{1F}$** Among people costs, *ensuring business continuity* is the most important cost variable influencing SOA architecture decisions

The following were the research propositions regarding the benefit dimension:

**P2** Re-use, flexibility, people, integration, open standard benefits are the most important benefit variables influencing SOA architecture decisions

- **$P_{2A}$** Among re-use benefits, *business process re-use* is the most important benefit variable influencing SOA architecture decisions
- **$P_{2B}$** Among flexibility benefits, *time to market* is the most important benefit variable influencing SOA architecture decisions
- **$P_{2C}$** Among people benefits, *development of a standardized employee skill set* is the most important benefit variable influencing SOA architecture decisions
- **$P_{2D}$** Among integration benefits, *inter-organizational collaboration* is the most important benefit variable influencing SOA architecture decisions
- **$P_{2E}$** Among benefits from open standards, *platform and technology independence* is the most important benefit variable influencing SOA architecture decisions

## Conclusion

In this research, we have studied the relationship between IT architectures and business value. By breaking down the cost constructs into variables such as organizational, business process, technology, servicification, integration, people and benefit constructs into variables such as re-use, flexibility, integration, open standards, and people, we attempted to find the most important cost and benefit variables that have to be considered while measuring the business value emerging out of SOA-based IT architectures. Our study reveals that the most important cost variable in SOA-based architecture implementations is the identification of new business services. This is followed by variables such as understanding business models and strategies, organizational buy-in, service realization through interface development, identifying existing services, and identifying the optimal level of service granularity.

The most important benefit is derived from low vendor lock-in (allowing enterprises to replace customised applications and products). This is followed by variables such as platform and technology independence (leading to greater collaboration and integration, with less spending on middleware infrastructure), time to market (which enables enterprises to come out with products/services in a shorter timeframe), adaptability to emerging business scenarios (which facilitates the enterprise to realize faster growth by speedier and responsiveness to market conditions), benefits due to a reduction in application redundancy (emerging from rationalization of applications), and re-use of business processes, applications, and infrastructure (through the potential reuse of services and infrastructure), standardized employee skill sets (as the SOA interfaces is abstracted from the implementation), benefits from loose coupling (as SOA enables systems to be assembled and disassembled easily as they have less dependency on other systems). Finally there are benefits from reduced cost and time of introducing new business processes, reduced cost and time of modifying existing business processes, reduced cost and time of introducing new applications, reduced cost and time of modifying existing applications, reduced cost and time of introducing new IT infrastructure, and reduced cost and time of modifying existing IT infrastructure.

Our research implies that the IT investment decision makers need to have a comprehensive understanding of the cost and benefit variables adopted in the study to arrive at the business value measure to justify SOA-based IT architectures. Understanding the role of SOA-based architectural approaches in achieving flexibility across the three dimensions can guide appropriate investments in flexible IT systems and strategic IT planning. Studying the influence of IT architectures on organizational flexibility poses a number of challenges. In understanding the impact of IT architectures on organizational flexibility, we have studied business processes, IT applications, and IT infrastructure as the three dimensions to capture organizational flexibility. But organizational flexibility could be influenced by a number of other factors.

***About the Authors***

*Dr. Jai Ganesh is a Senior Research Associate with the technology research division of Infosys Technologies Limited. He obtained his PhD in information systems from the Indian Institute of Management Bangalore (IIMB) in 2003 and holds an MBA degree in corporate strategy and marketing. His research focuses on Web services, IT strategy and adaptive enterprises.*
*jai_ganesh01@infosys.com*

*Niranjan Iyengar is a technical architect and Infosys Technologies Limited.*
*niranjan.iyengar@authors.sys-con.com*

*Dr. Sriram Anand is a principal researcher at Infosys Technologies, Bangalore. Prior to joining Infosys he worked in IT consulting as well as product engineering in the US for over 12 years. His interests include enterprise architecture, service-oriented architecture, and legacy integration and software engineering methodologies. Dr. Anand is experienced in designing enterprise architectural strategy for leading U.S. companies in the financial services, retail, and pharmaceutical domains. He holds a Bachelor?s degree from IIT-Madras with a PhD from SUNY-Buffalo, USA.*
*sriram_anand@infosys.com*

# Extracting UML from Legacy Applications

## Modeling existing business processes to extend their value

WRITTEN BY **MIKE OARA AND RICHARD SOLEY**

The operations of many large organizations rest on large applications that are characterized as "legacy." To increase flexibility or reduce costs businesses are looking to modernize these applications, for instance, via renovation, introducing an SOA architecture, or even re-implementing in a new environment. No matter which approach is taken, it's important to salvage as much knowledge and logic as possible from the legacy application. Unless the application's function is obsolete recovering functional knowledge (what does the application do?) and structural knowledge (how does it do it?) can accelerate the modernization effort.

A parallel can be drawn with renovating a building, since modernization can involve gradual changes to the building's internal structure, say, larger doors, or complete demolition and reconstruction. In both cases blueprints of the buildings are required. These blueprints form a shared basis of knowledge between the architect and the developer that's necessary for planning and execution. Just as blueprints are necessary to determine how a building can be adapted to suit a new need, they are also necessary to determine how to adjust an application. However, since legacy applications have, by their nature, been developed over long periods of time and, in most cases, by many people such blueprints don't exist. They have to be recovered to start the modernization process.

Many legacy analysis tools have emerged to address this situation. They attempt to reveal the internal and implicit architecture and business function of a legacy application in a fashion understandable for people. The discovery and then the expression of the business function at an abstract level that discards incidental technical details is a difficult task. However, this is precisely what is necessary since modernization projects are interested in re-implementing business functions, not particular technical solutions.

The tension between expressing business functions in a comprehensible fashion and expressing them with enough detail and precision has to be addressed. This requires a language in which business knowledge can be practicably expressed.

### UML as a Solution

UML has emerged as the most successful non-proprietary object modeling and specification language. UML includes a standardized graphical notation that can be used to create an abstract model of a system, that is the UML model. UML can precisely and understandably describe an application at a high level of abstraction, hiding the implementation details to reveal the actual business functions. Furthermore, as a formal language, UML has a well-defined syntax that makes it suitable for both forward and reverse engineering. In forward engineering, a UML model could be used to generate actual code (e.g., Java code). Most commercially available UML tools are capable of forward engineering (in various degrees), while possessing some reverse engineering features. The most common reverse engineering activity involves extracting class diagrams from Java code. However, reverse engineering is limited so far to modern programming languages, while lacking similar capabilities for older technologies like COBOL.

### The Benefits of UML vis-à-vis Legacy

The ability to reverse engineer legacy applications to UML would offer substantial benefits:

- **Preservation of Knowledge at an Abstract Level**
  In many instances it may be necessary to re-implement an application in a new and modern technical environment. However, because the application represents years of organizational learning it contains functionality that must be preserved. The information must be described at a high enough level of abstraction to avoid unnecessary technical details. UML offers the right level of abstraction so that the author can retain as much detail as necessary for future implementation.

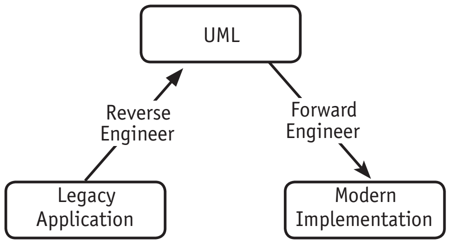
- **Communication Mechanism**
  Distance, language, and cultural issues can impair the communication of information about business functions between the legacy maintenance team and the re-implementation team. UML offers developers a common language that promotes precision and comprehensibility.

- **Vendor Independence**
  The universal acceptance and adoption of UML also offers the advantage of

choosing from a large number of UML tool vendors. One team can select Rational Rose to create the models while an outsourced re-implementation team can use Borland to view and reuse the same models. Users can even switch between tools and still preserve the knowledge encapsulated in the models.

- **Forward Engineering**

While the challenge to be discussed is reverse engineering legacy applications into UML, we should recall that many UML tools offer forward engineering capabilities.



Thus UML can be the intermediate stage through which an application rewrite may pass. The process would thus be:

### *"People Reuse"*

The standard method of acquiring business process knowledge through user interviews involves a major commitment of time and new resources. The results may be incomplete and error-prone. Alternatively, extracting knowledge directly from the current application circumvents these concerns while continuing to rely on the resources currently involved in application maintenance.

### *Limitations*

There's no silver bullet for reverse engineering legacy to UML. In fact, one may notice that some concepts simply don't match. Furthermore, some important information about the use of the legacy application isn't captured in the code itself, making automatic extraction impossible. For instance:

## Actors

In UML, an *actor* is a user of the system; "user" can mean a human user, a machine, or even another system that interacts with the system from outside its boundaries. Because of this definition in most cases information about actors can't be found in the code.

## Requirements

While the current system may implement business requirements, they may not appear explicitly in the code, but rather in the form of fulfilled requirements.

## Navigation & Sequence

Certain sequences of operations may not be explicitly specified in the application. So, while a user knows that to open a new account, he or she must perform activities A, B, and C in this precise order, the application may allow other paths that aren't meaningful from a business perspective.

We can therefore recognize that any UML description of a legacy application can't be achieved through completely automatic reverse engineering. While a legacy analysis tool may expose the artifacts of the application, only a human can assemble them into meaningful UML diagrams.

## A Balanced Approach

We have shown that a totally automated approach isn't feasible. At the other extreme, a completely manual approach has two primary disadvantages:

### *Economics*

Over time applications tend to be modified to such a degree that neither the initial plans, nor the current documentation reflects the reality of the application. Knowledge must be acquired from the code itself, but to manually review a multimillion-line application would be far too burdensome financially to be a realistic option.

### *Completeness*

As a legacy application is modified and enhanced over the years users often lose a complete understanding of how the application functions. For example, in a pension system the rules for computing the pension can be spread through numerous government and corporate policy documents. This knowledge is already in the code, which is more complete, precise, and concise than what would come from user interviews. Moreover, the application stakeholders are likely to insist that nothing is lost from the current functionality.

The best balance between fully manual and fully automatic can be called "tool assisted." In this approach, a software tool may be able to:

1. **Parse legacy code and show its information in a convenient manner.** Convenience is key since the selected tool would have to filter out a great deal of unnecessary detail while assembling the application information that is actually needed.
2. **Allow the user to select relevant legacy artifacts and quickly derive UML entities.** For example, the tool may show a list of COBOL structures that, when clicked on, create a class with the data members derived from the fields of the structure.
3. **Let the user quickly assemble UML diagrams based on derived entities.** Further to point 2, once two classes are created the user should be able to indicate a relationship that would appear in a class diagram.
4. **Export data in standard XMI notation.** Doing so would ensure that the user would end up not only with just attractive diagrams but useful models that can be refined by UML tools and used for the forward generation of the code.

These features may involve various degrees of automation. The automation appears wherever the extraction or deviation is clear and algorithmic. However, there are remaining steps that require human intervention to give meaning to the resultant models.

## What UML Diagrams Can Be Extracted?

We have now identified the approaches that yield the maximum benefit and their drawbacks. So let's look at specific information that can be extracted from the legacy application. These possibilities should be thought of as a starting point since more automation will likely arise as UML extraction tools increase in sophistication.

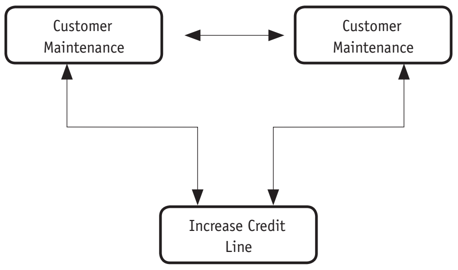
## Use-Case Diagrams

Use cases correspond to major areas of the application in which the user can accomplish defined business goals. The cascading hierarchy of menus can represent such major functional areas. The 'Maintain Account' menu screen may lead to 'Add Account,' 'Delete Account,' or 'Freeze Account,' each of which can be a use case of its own. As legacy analysis tools are aware of these screens as well as the transitions between them, the existing information can be used to extract use-case diagrams.

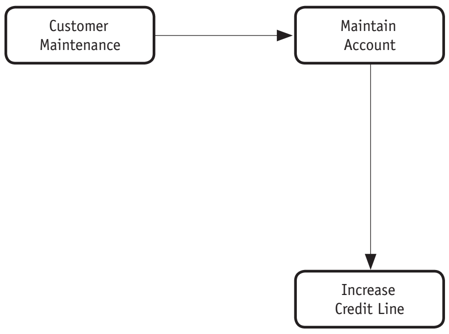
If automation is pursued, the task isn't so simple. A legacy analysis tool may be able to locate the screens and transitions between them. This would result in a graph in which the screens are nodes and the transitions are edges. A pure technical analysis would offer no indication of what the starting point for the navigation is and what the difference between entering a major area of the application and returning to the initial point is. For example, we may detect that there's a transition from 'Maintain Account' to 'Increase Line of Credit,' but from a purely technical viewpoint, the navigation may as well go in the reverse direction. In most cases, if the user can go from Screen A to Screen B then he can go from Screen B to Screen A. If each of the two screens indicates an important use case then what is the relation between them? Which one is using or including the other?

As a result, we have the interesting problem of taking a non-directional graph and transforming it into a directional one. One possible solution is to let the user designate a starting point, which is usually the application's main menu screen. An algorithm may be developed by which the program will traverse the graph starting with the designated main menu and navigate the edges without ever returning to a node (screen) that was visited before on the path from the main menu. This will result in a sub-graph of the screen navigation that's directional and indicates the normal navigation of the user who starts at the first main menu screen.

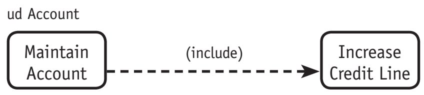
As an example, a total navigation graph that's automatically extracted from the code may look like this:



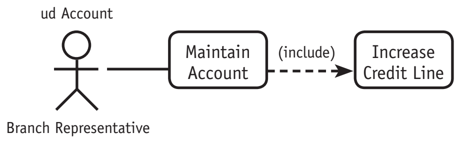
In this graph it's not clear if 'Increase Credit Line' includes 'Customer Maintenance' or the other way around. However, if 'Customer Maintenance' is designated as the starting point, the navigation graph may be reduced to:



Now it's clear that a use case called "Maintain Account" would include or use the use case "Increase Credit Line" and we can derive a use case diagram:

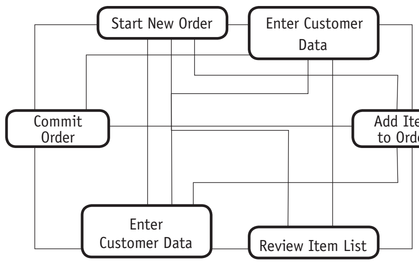


As stated earlier, the actors may be hard to detect from the code, so additional information may be needed to complete the picture. The manually adjusted diagram will show:

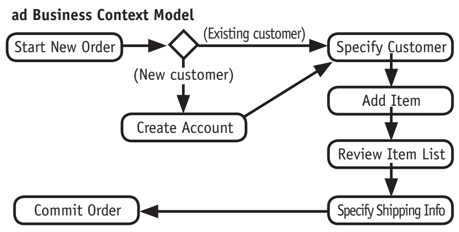


## Activity Diagrams

To do a certain task, the user of the legacy application must navigate a particular path through various screens while doing some specific operations on each. Legacy analysis tools are aware of the screens and all possible paths through them, and this information may be used to generate activity diagrams. The user of the UML extraction tool would then have to select particular paths that reflect particular user tasks. Now we have a problem similar to that for use cases. The screen flows of the legacy application may allow all possible transitions between the screens, but only some particular ones would make sense in the context of an activity diagram. A technical analysis of the application may indicate the following possible transitions:



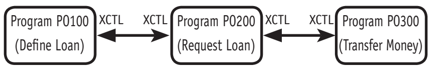
A tool-assisted extraction of an activity diagram can allow the user to point to a starting point and then select a particular navigation path, which reflects how the creation of a new order is done in real life. Furthermore, the user may also indicate some decision points. The final activity diagram would look like this:



## State Diagrams

The states of a particular object can appear more or less explicitly in the COBOL code. A so-called 88-level field could in itself list all the possible states. Looking at all possible values that a variable can have may help collect other states. Furthermore, even some of the conditions that lead to a state transition can be discovered.

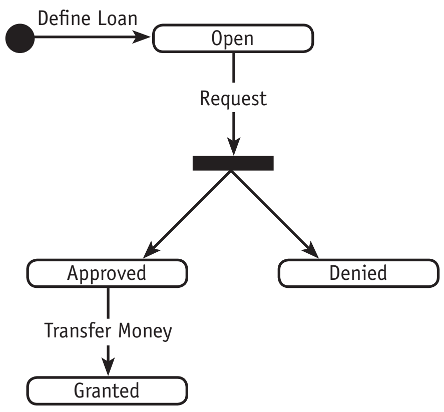
Legacy analysis tools can generally discover data flows (e.g., value moved to a field) and program connections (i.e., calls, links, exclusive control, etc.). Suppose, for example, that the legacy tool discovers the following:


Program P0100 (Define Loan)
XCTL
XCTL
Program P0200 (Request Loan)
XCTL
XCTL
Program P0300 (Transfer Money)

Data flow analysis done for the LOAN-STATUS field may discover the following statements:

| | |
|---|---|
| **P0100** | MOVE "OPEN" TO LOAN-STATUS |
| **P0200** | IF LATE_PAYMENTS = 0 THEN<br>MOVE "APPROVED" TO LOAN-STATUS<br>ELSE<br>MOVE "DENIED" TO LOAN-STATUS |
| **P0300** | MOVE "GRANTED" TO LOAN-STATUS |

A state diagram can then be inferred that would look like:



A legacy UML extraction tool would therefore need to discover data flows (focused on fields indicating state) and program flows and then process this information and assemble it into a state transition diagram.

## Class Diagrams

The discovery of the major classes in the application requires human judgment. While there could be compelling reasons to create certain classes, in many cases the classes are created for maximum programming convenience. There are some simple heuristics (e.g., create a class for each table) that could be used as an initial step. The data members of the class can be inferred from the sub-fields of a structure, from the columns of a table, or from other artifacts. While discovery of data members is almost a trivial exercise, the discovery of methods offers a more difficult challenge.

One way to discover the methods corresponding to a certain class is to take its data members (or rather, the legacy fields from which the members were inferred) and find their use in the application code. One may discover how they are calculated and used, and then designate corresponding methods for the class. If the class is derived from some data store, it's natural to create Insert, Update, Delete, and Read methods.

## Conclusion

UML extraction from legacy is beneficial and made possible by commercially available legacy analysis tools that caption legacy application knowledge in well-organized repositories. We expect that UML extraction software will emerge in consecutive steps and lead to increased automation, coverage, and precision. By using such UML extraction tools, companies can document and retain the business knowledge embedded in their legacy applications. UML would make sharing this information and the use of forward-engineering techniques for the purpose of re-implementing the functionality in new environments possible. ■

**About the Authors**

*Mike Oara is the co-founder and CTO of Relativity Technologies. He has 23 years of experience in enterprise software and is an acknowledged expert in automation methodologies for modernization and transformation of legacy applications. He has a MS in Mathematics from the University of Bucharest in 1979 and pursued graduate studies at the Courant Institute of NYU*

mike.oara@relativity.com

*Dr. Richard Mark Soley is Chairman and CEO of the Object Management Group, Inc. (OMG). As Chairman and CEO, Dr. Soley is responsible for the vision and direction of the world's largest consortium of its type. He holds the bachelor's, master's and doctoral degrees in Computer Science and Engineering from the Massachusetts Institute of Technology.*

soley@omg.org

## Product Review — *continued from page 21*

tiple instances of components in composite applications reduces the memory load of intensive services such as DB connections across multiple components further extending scalability.

An often-overlooked strength is the power of centralized modeling, development, deployment, and administration. Through rules-based security, roles can be restricted as required allowing all disciplines to view, alter, or administer the applications, servers, real-time messages, services, and related components from a common set of tools as required. Being able to have the BA, developer, QA analyst, and administrator all look at the same visual interface with no risk to the application and work through testing, deployment, monitoring, and debugging is invaluable. The ROI continues well past the development stage of your applications. The flexibility of the platform also excels at extending the lifecycle of legacy applications by adapting to the older platforms, then allowing you to extend the legacy applications' abilities by integrating newer protocols, standards, and abilities such as HTTPS, XML Messaging, and Web Services.

## Conclusion

With this release Fiorano has firmly established a mature development platform further increasing the ability of a business analyst to create application flows and transformations and freeing highly skilled developers to build richer solutions, custom services, and loosely coupled composite services. With minimal training the ROI can begin immediately and principles like RAD (Rapid Application Development), SOA, and code re-use become reality. The cost savings begins with the first stage of Business Process Modeling and continues all the way through day-to-day administration, extending the lifecycle of existing platforms where applicable.

Also noted are the improvements in performance and throughput in several key areas, enhanced simple and distributed transaction handling, improved ANT script support, and AXIS and Eclipse integration. On the horizon a native C# version of platform and enhanced Web Service support will further extend the scope of the platform. Consuming Web Services is a breeze but there's still work to do regarding incoming Web Services. Overall I was very impressed with the new release and have already realized immediate gains in time-to-production, quicker ROI, and richer applications. ■

**About the Reviewer**

*Warren Hampton is a senior e-commerce architect with Quicken Loans/Title Source Inc.*

warren.hampton@authors.sys-con.com

## SOA WSJ Advertiser Index

| Advertising Partner | Web Site URL | Phone # | Page |
|---|---|---|---|
| ACTIVE ENDPOINTS | ACTIVEBPEL.ORG/SOA | | 4 |
| AJAXWORLD | WWW.AJAXWORLDEXPO.COM | 201-802-3022 | 40-41 |
| ALTOVA | WWW.ALTOVA.COM | 203-929-9400 | 2 |
| FIORANO | HTTP:WWW.FIORANO.COM/DOWNLOADS | | 15 |
| FORUM SYSTEMS | WWW.FORUMSYSTEMS.COM | 801-313-4400 | 7 |
| IBM | IBM.COM/TAKEBACKCONTROL/SOA | | 50-51 |
| METALLECT | WWW.METALLECT.COM | 972.801.4350 | 11 |
| PARASOFT | WWW.PARASOFT.COM/WSJMAGAZINE | 888-305-0041 (X-3501) | 52 |
| ROGUEWAVE | WWW.ROGUEWAVE.COM/DEVELOPER/DOWNLOADS | | 23 |
| SOA WEBSERVICES JOURNAL | WWW.WSJ2.COM | 1-888-303-5252 | 37 |
| SONIC | WWW.SONICSOFTWARE.COM | 1 866 GET SONIC | 33 |
| WEB AGE SOLUTIONS | WWW.WEBAGESOLUTIONS.COM | 1 877-517-654 | 19 |
| WILY | TRUTH.WILYTECH.COM | 1-800-GET-WILY | 29 |
| XENOS | WWW.XENOS.COM/VAN | 1-888-242-0695 | 25 |

_INFRASTRUCTURE LOG

_DAY 18: Came to work and found everything frozen. Icicles are everywhere. It's our processes. They're inflexible. Hard coded so we can't respond to change.

_Why did we lock ourselves in like this? Brrrr.

_DAY 19: A way out. IBM WebSphere middleware for Business Process Management. It lets us streamline business tasks and optimize performance. We can simulate and test our processes so we understand the impact they'll have, then monitor performance once they're deployed. And because it's based on a service oriented architecture, it's easy to reuse and connect existing process-based services.

_Everything's unfrozen now. Wow, it's good to feel my toes again.

Take the BPM with SOA Assessment at:
IBM.COM/**TAKEBACKCONTROL**/PROCESS

*Ensure Interoperability.*

*Validate functionality.*

*Eliminate security vulnerabilities.*

*Test performance & scalability.*

*Confirm compliance.*

*Collaborate and reuse.*

## Ensure Secure, Reliable Compliant Web Services

PARASOFT®

**SOAtest™**

As enterprises adopt Service Oriented Architectures (SOA) to deliver business critical data, ensuring the functionality and performance of Web services becomes crucial. Complex web services implementations require the means to thoroughly validate and test them to assure they are truly production ready.

Parasoft SOAtest is a comprehensive, automated tool suite for testing web services and complex Service Oriented Architecture (SOA) solutions to ensure they meet the reliability, security and performance demands of your business. SOAtest provides a total and holistic testing strategy for your SOA implementations including automated unit testing, graphical scenario testing, scriptless performance/load testing, security penetration testing, standards validation, message authentication, and more.

**If you are building serious web services, you need SOAtest. For more information regarding Parasoft SOAtest, call 888-305-0041 (x-3501).**

**Download a copy of SOAtest for a free evaluation today at www.parasoft.com/WSJmagazine**

**Parasoft SOAtest clients include: Yahoo!, Sabre Holdings, Lexis Nexis, IBM, Cisco & more.**

PARASOFT®
*We make software work.™*

**Automated Error Prevention™**