

JULY/AUG 2005 VOL.4 ISSUE:4

WLDJ™

WWW.WLDJ.COM

**THE LEADING
INDEPENDENT
MAGAZINE
FOR WEBLOGIC™
PROFESSIONALS**

MIGRATING A JBOSS EJB APP TO WEBLOGIC

...28

PLUS...

Failover and Recovery of
Enterprise Apps PART 1 ...14

SOA and SODA ...22

Developing a Service
Information Portal on the
WebLogic Platform ...34


A Perfect Fit page 4

Process Monitoring and Management page 8

Defragment Your View of the World page 40

for a Quiet Life page 40

Paths to Service-Oriented Architecture page 46

A diver is seen from behind, swimming in clear blue water. Bubbles are rising from the diver's equipment. In the foreground, the dark, curved fin of a shark is visible, pointing towards the diver. The scene is lit from above, creating a bright glow and highlighting the bubbles.

A problem isn't
a problem if it
never happens.

Increase your visibility.
Decrease your risk.
Get control.
Intersperse Manager.

Enterprise information systems are more complex – and vital – than ever. But visibility into these dynamic architectures is minimal, increasing the chances of system failure and serious risk to your company.

Intersperse Manager™ removes the risks by proactively monitoring and managing distributed and SOA applications deployed on BEA. It gives users the ability to dive deeply into system infrastructure. To quickly pinpoint and correlate the root causes of problems. To automatically or manually correct issues in real time.

Intersperse Manager gives you complete visibility – and that puts you in complete control.

Call Intersperse today for
a free demonstration

800.340.4553

 **intersperse**™

Management solutions
for the evolving enterprise.

www.intersperse.com

Keep Your Skills Ahead of the Crowd

Keeping your IT skills ahead of the crowd is not as difficult as most people fear. Staying on top of the trends may seem like a daunting task if, like most people, you assume that each new technology is a completely new invention that you must learn from the ground up. Fortunately, nothing is really all that new. Inventors typically create new technologies by studying existing technologies, then building upon them in ways that extend and improve them. 100% new technological advancements are very rare.

Inventors almost always leverage legacy technologies as they invent new ones. Why not leverage your own knowledge of those legacy technologies as you try to learn about the new inventions? To learn about new technologies as painlessly as possible, consider how each new advancement is similar to what you already know.

For example, consider Web services. Web services are a new trend, but — at a technological level — the parts of a Web service are not all that unique. Web services are based on remote procedural calls — messages sent to a server, which calls the requested function. RPCs were developed years ago, and are hardly a new concept. Really, the only "new" thing in Web services is the standard that is being used to write the application. If you break down Web services in this way, it's easy to learn about them. To continue with this process, you might next explore the payload requirements, the process for determining what function to call, and how the call works. As you can imagine, it's a lot more efficient — and interesting — to learn about a new technology based on its relation to familiar technologies than to learn about it by reading the specification cover to cover.

As always, the devil is in the details. But most details are critical only if you want to specialize in a given technology. For instance, if you want to specialize in Web services, you need to familiarize yourself with the details of Web service development. In that case, your next step would be to learn how to format the messages, how to expose Web services, and so on.

— Adam Kolawa, Ph.D.
Chairman/CEO of Parasoft

It's automated. It's fast. And it's the most versatile Web Services testing tool.



SOAPtest® enables you to deliver better Web Services in less time.

The simple fact is, no other Web service testing product can do what SOAPtest can do. From verifying functionality and performance to ensuring security and interoperability, SOAPtest automates all critical testing processes across the entire lifecycle of your Web service.

But don't take our word for it... Go to www.parasoft.com/soaptest_WSD and try it for free — no commitment, no obligation. If you like it (and we suspect you will) just let us know. We'll be more than happy to help you and your development team get up and running.

For Downloads go to www.parasoft.com/soaptest_WSD



Email: soaptest@parasoft.com Call: 888-305-0041 x1209

Features

- WSDL schema verification and compliance to standards
- Automatic test creation using WSDL and HTTP Traffic
- Data-driven testing through data sources (Excel, CSV, Database Queries, etc)
- Scenario-based testing through XML Data Bank and Test Suite Logic
- Flexible scripting with Java, JavaScript, Python
- WS-I Conformance: Basic Profile 1.0
- WS-Security, SAML, Username Token, X.509, XML Encryption, and XML Signature support
- WS-Security Interop testing emulator
- MIME Attachment support
- Asynchronous Testing: JMS, Parlay (X), SCP, and WS-Addressing support
- Windows Perfmon, SNMP, and JMX monitors
- Detailed Report generation in HTML, XML and Text formats
- Real-Time graphs and charts

Benefits

- Uniform test suites can be rolled over from unit testing to functional testing to load testing
- Prevent errors, pinpoint weaknesses, and stress test long before deployment
- Ensure the reliability, quality, security and interoperability of your Web service
- Verify data integrity and server/client functionality
- Identify server capabilities under stress and load
- Accelerate time to market

Protocol Support

- HTTP 1.0
- HTTP 1.1 w/Keep-Alive Connection
- HTTPS
- TCP/IP
- JMS

Platforms

- Windows 2000/XP
- Linux
- Solaris

Contact Info:

Parasoft Corporation
101 E. Huntington Dr., 2nd Flr.
Monrovia, CA 91016

www.parasoft.com

A Perfect Fit



By James Fenner

The first house I ever bought was built in 1936. It had style, it had character, and it had really narrow hallways and tight corners. The sofa we had bought – the one that went perfectly with all the style and character – wouldn't fit in the house. Apparently folks in 1936 had smaller furniture. Eventually I learned out how to take apart a window and was able to get the sofa into the house, but in the process, Pandora snuck out.

Behind the window frame was rot. I looked into some of the other windows and found rot and bugs. My initial reaction was that I'll just need to replace the windows, but (of course) it wasn't to be that easy. The windows, you see, had been built from scratch with the house. In fact, everything about the house was custom built. The parts of the house were all very tightly coupled with one another making replacement with componentized, functional equivalents too expensive. Though it had been well taken care of, the bottom line was that it was an old house and needed a lot of attention in order to maintain its primary function: being a living space. Ultimately we decided that the best alternative was to move into something newer that didn't need so much attention.

Many of my clients find that they're in a similar place with their software portfolio. As recently as ten years ago, they had to build systems infrastructure that wasn't then commercially available. They needed scalability, clustering, fault tolerance, high performance and integration to legacy systems and they needed to be able to add new business features at a break-neck pace. Design compromises were made, corners were cut, and the platform was not kept up-to-date. Though most have achieved great success, they now find their systems to be bloated, brittle, too highly coupled, and staggeringly expensive to maintain.

Most businesses don't have available the equivalent of buying a new house; they can't throw everything

away and start from scratch. Instead, many are choosing the Herculean task of incremental renovation, of moving toward a service-oriented architecture in which their systems and their business process operate as an enterprise, not just a collection of departments. However as they turn over the rock of SOA, most are finding a dizzying array of new plumbing and buzzwords. The question is how to achieve the benefits of SOA without again becoming plumbers and in a way that is truly flexible. The answer is to focus on delivering custom business services in a standard way on a proven, commercial platform.

WebLogic remains the best foundation for that platform. It solves many of the boring plumbing problems in a way that is stable, scalable, and highly performing. It has never been easier to build and deploy distributed systems, but the market's evolution toward SOA has introduced new plumbing problems. It could have resulted in yet another round of unfortunate custom solutions, but BEA has got our back. This month they've unleashed their new family of products, AquaLogic.

AquaLogic adds to the capabilities of WebLogic, which makes it a perfect fit for the SOA generation. With it we can continue to focus on the competitive aspects of our systems – the user interaction and the business objects and policies. AquaLogic's integrated service bus, data access, security, and registry services solve the new plumbing problems. Without taking our eye off the ball, our systems can become more decoupled, better integrated, and more secure. It even comes with its own built-in monitoring and performance measurement tools. This broader platform more easily delivers the agility – the fluidity – that our systems need to keep up with the pace of change in the enterprise.

Congratulations to BEA on their launch of AquaLogic, and here's to a future where rotten windows can just be replaced. ●

Author Bio:

Jim is a senior systems architect at CSC Consulting. He specializes in architecture and delivery of business systems and more recently in forensic software analysis and remediation.

Contact:

jfenner@gmail.com

EDITORIAL ADVISORY BOARD

Lewis Cirne, Wayne Lesley Lund,
Chris Peltz, Sean Rhody

EDITORIAL

Founding Editor
Peter Zadrozny

Editor-in-Chief
Joe Mitchko joe@sys-con.com

Managing Editor
Seta Papazian seta@sys-on.com

Editor
Nancy Valentine nancy@sys-con.com

Research Editor
Bahadir Karur, PhD bahadir@sys-con.com

PRODUCTION

Production Consultant
Jim Morgan jim@sys-con.com

Lead Designer
Abraham Addo abraham@sys-con.com

Art Director
Alex Botero alex@sys-con.com

Associate Art Directors
Louis F. Cuffari louis@sys-con.com
Richard Silverberg richards@sys-con.com
Tami Beaty tami@sys-con.com
Andrea Boden andrea@sys-con.com

CONTRIBUTORS TO THIS ISSUE

Andy Armstrong, Raffi Basmojian, Peter Halditch,
Balamurali Kothandaraman, Bahar Limaye, Alex
MacInovsky, Joe Mitchko, Eric Newcomer,
Blair Taylor, WLDJ News Desk

EDITORIAL OFFICES

SYS-CON MEDIA
135 Chestnut Ridge Rd., Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9638
WLDJ (ISSN #1535-9581)
is published bimonthly (6 times a year)
by SYS-CON Publications, Inc.

Postmaster: send address changes to:
WLDJ: SYS-CON MEDIA
135 Chestnut Ridge Rd., Montvale, NJ 07645

©COPYRIGHT

©Copyright 2005 by SYS-CON Publications, Inc. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy or any information storage and retrieval system, without written permission. For promotional reprints, contact reprint coordinator, SYS-CON Publications, Inc., reserves the right to revise, republish and authorize the readers to use the articles submitted for publication. All brand and product names used on these pages are trade names, service marks or trademarks of their respective companies. WLDJ and WLDJ.com are registered trademarks of SYS-CON Media. WebLogic is a trademark of BEA Systems. SYS-CON and WLDJ are independent of BEA Systems, Inc.

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@eposdirect.com

REPRINTS

For promotional reprints, contact reprint coordinator Dorothy Gil dorothy@sys-con.com. SYS-CON Publications, Inc., reserves the right to revise, republish and authorize its readers to use the articles submitted for publication.



Why is it that some Java guys are more relaxed than others?

These days, with everything from customer service to sales and distribution running on Java, keeping your enterprise applications available can be pretty stressful.

Unless of course, you've discovered the power of Wily. No other software offers our level of insight. Which means you'll be able to monitor transactions and collaborate with colleagues in real-time. Even more important, you'll be able to optimize performance across the entire application—end-to-end.

So put Wily to work. And keep performance problems from pushing you over the edge.

Get Wily.™

1 888 GET WILY | wilytech.com

wily
technology

©2004 Wily Technology, Inc. The Wily logo is a trademark of Wily Technology, Inc. Java is a trademark of Sun Microsystems in the U.S. and other countries.

Enterprise Service Bus



By Anbarasu Krishnaswamy

USING ANALOGY TO UNDERSTAND ESB

BEA Systems, Inc. recently announced the launch of new family of products for service infrastructure, named AquaLogic. AquaLogic Service Bus (ALSB) is BEA's implementation of the Enterprise Service Bus combined with Web services management capabilities. This article is a high-level introduction to ESB and ALSB.

In their Harvard Business Review article, Giovanni Gavetti and Jan W. Rivkin explain the power of reasoning by analogy. They explain how successful executives have tapped the power of analogy to build businesses by comparison to other successes in business history. Analogy can also be used to understand new concepts and to predict the future of the concept. In this article I use the analogy of Universal Serial Bus (USB) to understand Enterprise Service Bus.

Universal Serial Bus (USB)

A few years ago, when computer desktops were costing an arm and leg, it wasn't easy to connect any peripheral devices to the computer. Keyboards and the mouse had their own dedicated PS/2 ports. Printers were connected to the parallel ports. Mice and other devices could be connected to serial ports. Hard drives had to be installed in their own IDE slots. There was no common standard. Also, installing these devices was a very laborious process. You had to connect the devices, install the software, configure interrupts, and finally keep your fingers crossed hoping that it would work. It was also very time-consuming to troubleshoot any problems.

With the advent of Universal Serial Bus (USB), these problems started to fade away. The faster USB 2.0 followed the USB 1.0 specification. Vendors realized the promise of USB and embraced it in no time. The idea of USB was to provide standards-

based connectivity that would enable a unified approach to access devices. Any device supporting USB could easily be plugged into the desktop.

Printers, keyboards, mice, scanners, Web cams, digital video cameras, digital cameras, jump drives, CD burners, cell phones and many other devices can be hooked up to the computer and configured easier than ever. USB hubs allowed the port to be extended to connect more devices simultaneously.

The devices can be thought of as providers of specific services. For example, the jump drive provides the service of storing data, CD burners provide CD writing service, scanners provide scanning services, etc.

There are software applications and other devices that consumer these services. For example, an image capturing software would capture the images from the camera connected. NetMeeting software uses the keyboard, mouse, and cam for a videoconference.

Multiple consumers may use the same service. You can print a document from MS Word or MS Power Point. NetMeeting as well as Messenger can use the cam. This is a classic analogy to SOA where multiple consumers share the services.

Services can also be connected with each other and the flow may be routed to perform a compound task. For example, your scanner may have a printer button. Clicking this will scan and print the document in the printer. Figure 1 shows the various consumers and producers for the USB.

— continued on page 42

Author Bio:

Anbarasu Krishnaswamy is a consulting technical manager with BEA professional services. He is a six-year BEA veteran, and has been working on enterprise applications for over 12 years. He holds a Masters degree in Computer Science and is certified by BEA and Sun. In his current role, he helps customers in various vertical industries to architect enterprise applications. He specializes in architecting enterprise systems using J2EE, EAI, and SOA

Contact:

anbarasu@bea.com

PRESIDENT & CEO

Fuat Kircaali fuat@sys-con.com

VP, BUSINESS DEVELOPMENT

Grisha Davida grisha@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

Advertising Sales Manager
Dennis Leavey dennis@sys-con.com

Associate Sales Managers
Dorothy Gil dorothy@sys-con.com
Kim Hughes kim@sys-con.com

SYS-CON EVENTS

President, Events
Grisha Davida grisha@sys-con.com

National Sales Manager
Jim Hanchrow jimh@sys-con.com

CUSTOMER RELATIONS

Circulation Service Coordinators
Edna Earle Russell edna@sys-con.com
Linda Lipton linda@sys-con.com

Manager, JDJ Store
Brunilda Staropoli brunil@sys-con.com

SYS-CON.COM

VP, Information Systems
Robert Diamond robert@sys-con.com

Web Designers
Stephen Kilmurray stephen@sys-con.com
Percy Yip percy@sys-con.com
Vincent Santilli vincent@sys-con.com

Online Editor
Roger Strukhoff roger@sys-con.com

ACCOUNTING

Financial Analyst
Joan LaRose joan@sys-con.com

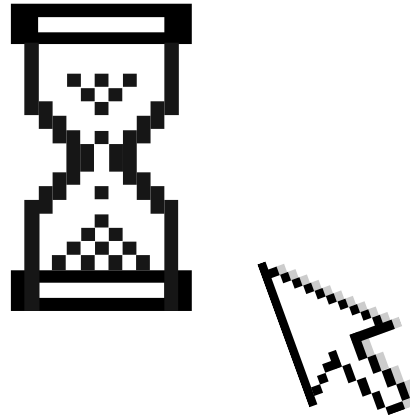
Accounts Payable
Betty White betty@sys-con.com

Credits & Accounts Receivable
Gail Naples gailn@sys-con.com

SUBSCRIPTIONS

subscribe@sys-con.com
Call 1-888-303-5282

For subscriptions and requests for bulk orders, please send your letters to Subscription Department
Cover Price: \$8.99/issue
Domestic: \$149/yr (12 issues)
Canada/Mexico: \$169/yr All other countries \$179/yr
(U.S. Banks or Money Orders)
Back Issues: \$12 U.S. (plus \$8/H)



World Wide Wait.

Don't Leave Your Customers Hanging — Ensure a Positive End User Experience with Quest Software.

They don't care where the problem is. All that matters is their experience using your site. But can you quickly detect and diagnose a problem and deploy the right expert to fix it?

Quest Software's solutions for J2EE application performance management shed light on the true end user experience and help you isolate and fix problems across all tiers. Whether a problem lies in your server cluster, database, framework or Java code, nothing gets you to resolution and optimal performance faster.

Don't just cross your fingers and hope for the best. Get more with Quest.

**Download the free white paper,
"Measuring J2EE Application Performance in Production"
at www.quest.com/wldj**

Using the WebLogic Platform to Create a Real-World Business Process Model for Order Management *Part 5*

PROCESS MONITORING AND MANAGEMENT



By Anjali Anagol-Subbarao

In the first article I gave an overview of BPM and covered the specifications in this area. I described the order change example and the steps needed to create the business process in WLI. In the second article we saw how to create a process application, specifically how to create the application orderChange. In this application I created a new process called orderChange.jpd. To start the process I added a ClientRequest received. Next we added the Web service validate config.

In the third article I added a decision point to handle the result from validate config Web service. The decision point helps in handling both the positive and negative outcome of the result from the process. Then I added to the process a database control to check the order status of the order to be changed, and last I added another decision node to handle the result from the database Control. In the last article we saw how the change order is written out to a file, and we also saw how this change order was added to an ERP-based system specifically for SAP. We also examined the code for the process.

In this article we will see how the JPD file created can be exported to a BPEL file. We will also see how the process is executed and you can see the end result on WebLogic's Test Browser. We will also see how the process is monitored in WLI and also how HP's OpenView can be used to monitor the process.

Exporting to BPEL

In the earlier article we saw how to create a business process in WLI. It creates a JPD file based on PD4J specification. WLI gives you the facility to export this file to a WS-BPEL file through the BPEL exporter.

Below is the aforementioned JPD file now exported to BPEL:

```
<process name="orderchange" xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/" xmlns:jpd="http://www.bea.com/wli/jpd" xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/" xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/" xmlns:wli="http://www.bea.com/workshop/bpel/wli" xmlns:ctrl="http://www.bea.com/workshop/bpel/ctrl" xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.openuri.org/" expressionLanguage="http://www.w3.org/TR/2003/WD-xquery-20031112/" >
```

This shows the partner information:

```
<partnerLinks>
  <partnerLink name="client" partnerLinkType="generated" myRole="provider" partnerRole="client" />
  <partnerLink name="validateConfignew" partnerLinkType="unresolved-type" />
  <partnerLink name="orderstatus1" partnerLinkType="unresolved-type" />
  <partnerLink name="ChangeorderFile" partnerLinkType="unresolved-type" />
</partnerLinks>
```

This is the start of the process:

```
<variables>
  <variable name="orderChangexsd" type="unresolved-type" />
  <variable name="fileproperties" type="com.bea.wli.control.dynamicProperties.FileControlPropertiesDocument" />
</variables>
```

OrderChangeRequest as ClientRequest:

```
<sequence>
  <receive jpd:name="orderChangeRequest"
```

Author Bio:

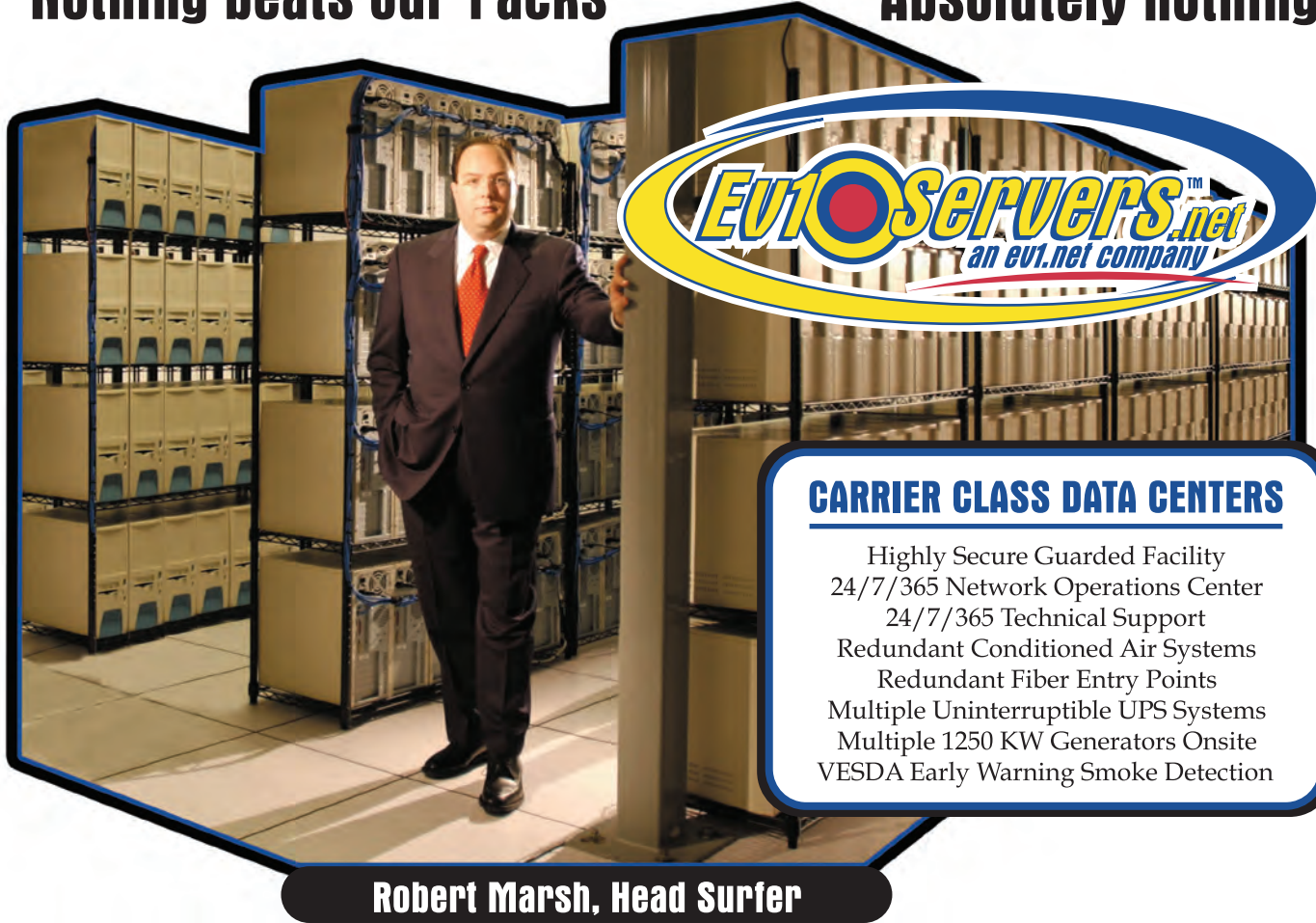
Anjali Anagol-Subbarao works in HP's IT organization as an IT architect. She has 12 years of IT experience, the last five in Web services. Her book on J2EE Web services on BEA WebLogic was published in October 2004.

Contact:

anjali.anagol-subbarao@hp.com

Nothing beats our racks

Absolutely nothing



CARRIER CLASS DATA CENTERS

- Highly Secure Guarded Facility
- 24/7/365 Network Operations Center
- 24/7/365 Technical Support
- Redundant Conditioned Air Systems
- Redundant Fiber Entry Points
- Multiple Uninterruptible UPS Systems
- Multiple 1250 KW Generators Onsite
- VESDA Early Warning Smoke Detection

Robert Marsh, Head Surfer

START YOUR OWN WEB HOSTING BUSINESS TODAY!

from
\$299*
Instant Activation!

Dedicated Server

Dual Xeon 2.4 GHz
2 GB RAM • 2 x 73 GB SCSI HD
Remote Console • Remote Reboot
2000 GB Monthly Transfer Included

Over 20,000 Servers!

1-800-504-SURF | ev1servers.net

PLESK7
RELOADED
Preferred Control Panel

IP Compliant. Price subject to change. Quantities Limited.

*Per month. Set-Up fees apply. See web site for complete details.

```
partnerLink="client" portType="clientPT"
operation="orderChangeRequest" variable="orderCh
angexsd" createInstance="yes" >
</receive>
```

JPD code is included as comments:

```
<jpd:javacode code="{
  // #START: CODE GENERATED - PROTECTED
  SECTION - you can safely add code above this
  comment in this method. #//
  // input transform
  // parameter assignment
  this.orderChangexsd = orderChangexsd;
  // #END : CODE GENERATED - PROTECTED
  SECTION - you can safely add code below this
  comment in this method. #//
}>
```

The process calling the validateConfig Web service:

```
<invoke jpd:name="validateConfig" partnerLink
="validateConfignew" portType="unresolved-type"
operation="validateConfig" >
</invoke>
```

The first decision point to see if the configuration is valid:

```
<switch jpd:name="Is configuration Valid?">
```

```
<case jpd:name="Yes" condition="data($outV
alidateConfig/ns:Status) = &quot;true&quot;">
```

The Process calling the order status database control:

```
<sequence>
  <invoke jpd:name="OrderStatus" partne
rLink="orderstatus1" portType="unresolved-type"
operation="getShipDate" >
```

This is the second decision point to find out if order is changeable:

```
</invoke>
<switch jpd:name="Is order
changeable?">
  <case jpd:name="Yes" condition="jpd:
method" jpd:method="condition">
```

```
</switch>
```

The process writes the file through file control:

```
<invoke jpd:name="write" partnerLink="C
hangeorderFile" portType="unresolved-type"
operation="write" inputValue="orderChangexsd"
outputVariable="fileproperties" >
```

```
</invoke>
</sequence>
</process>
```

Execution of Process

When you execute the process you will see the test SOAP message in the Test Browser, when you execute it you will see it will go through all of steps of the process: client request, validate config Web service, the first decision point, the order status information from the database, the second decision point, and then the end of process by writing the XML to a file. Figure 1 shows the validate config node of the processes. Figure 2 shows the last node of the process, which is writing to the file.

Process Monitoring

Management of business processes means we want to ensure that individual business process instances can be monitored to see if they finish within a stipulated time. We also want to gather metrics to improve and fine-tune our processes. You can monitor instances of your business process through the WebLogic Integration Administration Console in the WebLogic Workshop menu. Enter the <http://localhost:7001/wliconsole> URL in a Web browser. The default user name and password for the sample integration server is weblogic/weblogic. Click Process Instance Monitoring to open a page that allows you to (BEA):

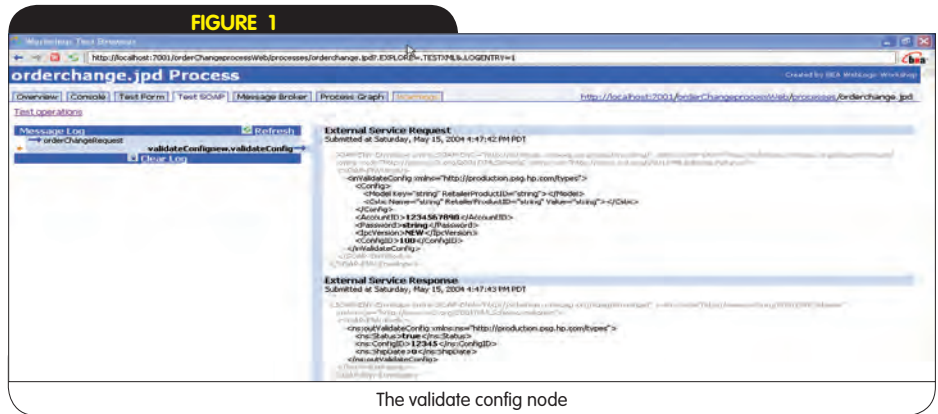
- View process instance statistics, including the number of instances in each state (running, suspended, aborted, and completed)
- View the summary or detailed status for selected instances
- Suspend, resume, or terminate selected instances

Figure 3 shows the WebLogic Integration Console.

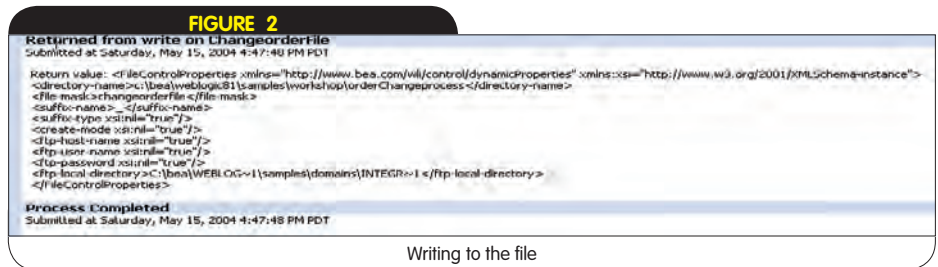
Management of Business Process Through HP Open View

As we described in the earlier paragraph for management of business processes, we need the following capabilities:

- Receiving alerts when something goes wrong (management through exceptions), such as when a business process instance is beyond the threshold level or did not complete according to the service level agreement (SLA), as well as when an error exists in part of the application.
- Having insight into current state-of-business processes with a view of business processes, the message broker, message broker



The validate config node



Writing to the file



TANGOSOL™



A Cure for Downtime.

Eliminate the single points of failure, add fault tolerance and enable transparent failover for your J2EE applications.

TANGOSOL COHERENCE™ has no single points of failure, ensuring high availability and reliability for your mission-critical J2EE applications. Application state, HTTP sessions and data caches are never at risk. Applications using Coherence keep it up without interruption, even when servers go down.

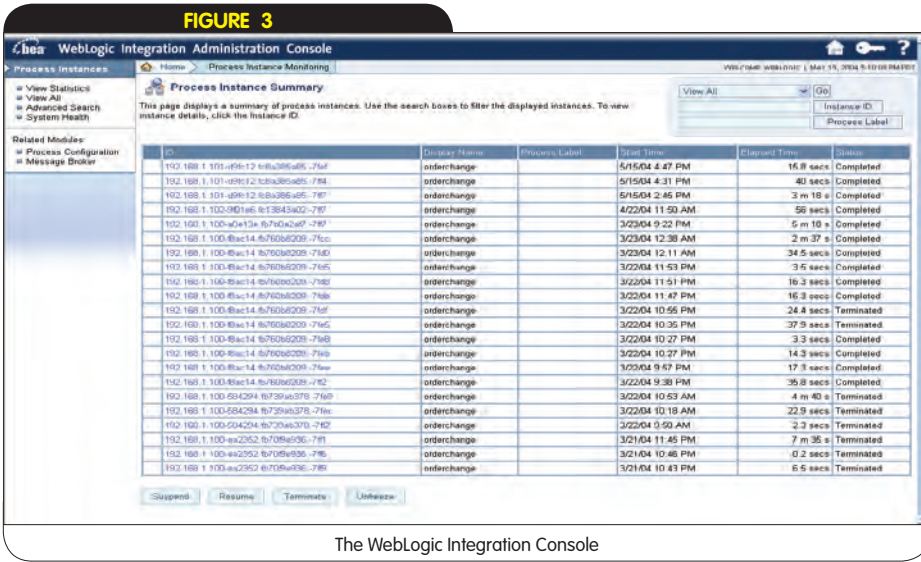
Try before you buy.

Download a free evaluation license at

www.tangosol.com

Coherence™ is pure Java software that reliably and cost-effectively scales J2EE applications across any number of servers in a cluster. Coherence provides reliable coordinated access to in-memory data that enables safe data caching and user session management for applications in a clustered environment.

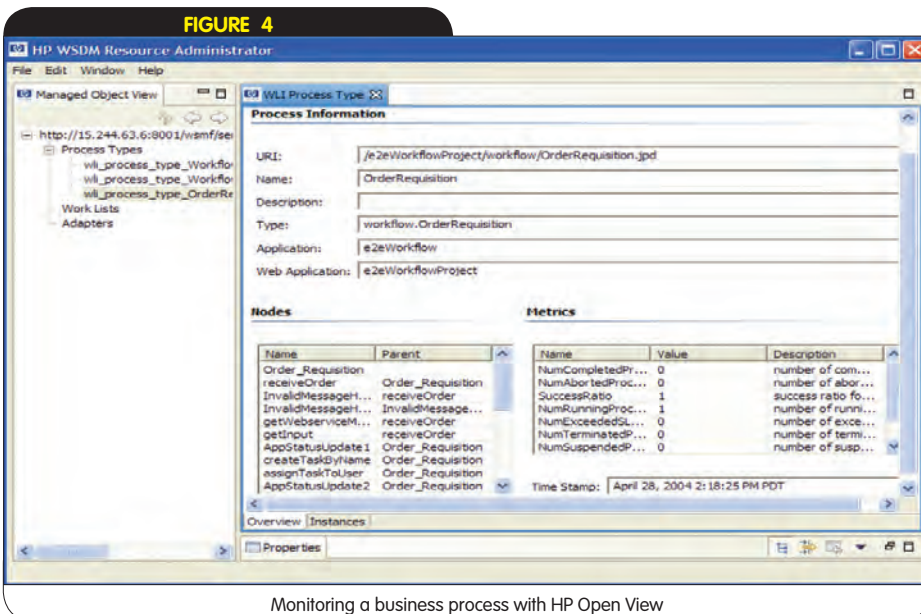
Sales +1.617.623.5782 or sales@tangosol.com
Support support@tangosol.com or <http://www.tangosol.net>
Pricing US\$1,995 to US\$4,995 per Coherence production CPU
Development licenses are offered free of charge.



lets us use the HP OpenView product family to manage WLI business processes. After installing WLI-SPI within an existing HP OpenView management system, we can use the full power of HP OpenView management to harness our business processes via tools including (DRC):

- List Business Process Types, Instances, Adapters, Event Generators, Message Channels, and the System Archiver and their relevant attributes. Also, carry out management operations on these through OV.
- Monitor performance of business processes and other relevant metrics in real time
- Monitor critical events such as failure conditions or violations of performance SLAs on Business Process execution time through notifications
- Generate historical reports on performance metrics
- The ability to customize the SPI to include application-specific monitoring and management capabilities

Figure 4 shows how a business process can be monitored with HP Open View.



channels, work list, adapters, and event generators with the ability to drill down into detailed attributes. We need statistics on instances of a given business process that answer questions such as how many have been completed, are currently running, have violated the SLA on execution time, have been terminated, have been aborted, etc.

- Carrying out certain operations on business processes or related entities such as suspend a process or resume a suspended process.
- Being able to monitor in real time critical performance metrics such as: average execution time for business processes, business process success ratio, error

count for adapters, message channels, and event generators.

- Receiving historical reports on the aforementioned performance metrics.

In the case of applications running on WLI, most of the information discussed above is already available within the WLI execution engine and is accessible to any external program via JMX MBeans. This availability, combined with programs to collect, analyze, and present the information to the right people in the right form, will allow us to manage our business processes.

The HP OpenView Smart Plug-in for WLI (WLI-SPI) is the “glue” software that

Summary

In this article we have seen how to export a jpd file to a WS-BPEL file. We also saw how to execute the business process and check the results in the test browser. We saw how the different process nodes were executed in the test browser. We talked about management of business process. Monitoring can be done in WLI through the WebLogic Integration Console. You can also use HP Open View to do the management of business processes. HP Open View uses SPIs to manage business processes in WLI through JMX Beans.

Finally, to learn more about the HP OpenView and the BEA application management solutions, visit the <http://openview.hp.com/bea> site. You will find that together, BEA and HP OpenView offer you more control and agility, resulting in a greater competitive edge.

References

- (BEA) WLI monitoring: <http://dev2dev.bea.com/>
- (DRC) Kumar, Pankaj. “Manage your business processes on BEA WebLogic Integration with the HP OpenView WLI-SPI”: <http://devresource.hp.com/drc/columns/col0407a.jsp>

A photograph of two men in an office setting. The man on the left, wearing a blue and white striped sweater, is leaning over the desk and pointing at a computer monitor. The man on the right, wearing a plaid shirt, is looking at the screen with a focused expression. The background shows office cubicles and desks.

**Create software so brilliant
it can manage itself.**

Want to spend more time developing software and less time supporting it? Spend some time discovering hp OpenView—a suite of software management tools that enable you to build manageability right into the applications and Web services you're designing.

Find out how the leading-edge functionality of hp OpenView can increase your productivity.

<http://devresource.hp.com/d2d.htm>



Failover and Recovery of Enterprise Applications – Part 1

HIGH AVAILABILITY - MOVING BEYOND CLUSTERING

By Sudhir Upadhyay

In enterprise application architecture, it is naïve to assume that none of the software/hardware components will go down. In fact, most of the IT managers and architects acknowledge this. However, a well-tested and robust recovery procedure continues to take a back seat when designing and implementing software projects. In several scenarios, administrators end up performing basic failover testing by shutting down the processes and verifying that the subsequent requests succeeded.

Although this level of testing can satisfy the failover requirements for the records, more robust failover testing needs to be performed to ensure a proper recovery if failures do occur. One of the primary reasons for the lack of robust recovery and well-tested procedures can be attributed to a high degree of reliance on the application server's in-built failover capabilities. While it is true that most of the popular application servers boost their clustering and associated high-availability features, it is equally important that system architects need to go beyond those features for an end-to-end high availability of the application. This article suggests how good planning, robust scripting, and a well-thought-out application design, along with application server facilities, can provide a highly available environment for the entire application. In the first article in a series of three, I will cover the basic high-availability options available while deploying a J2EE application on WebLogic Server.

Before diving into specifics, it is important to understand some of the commonly used terms when one talks about high-availability.

High Availability – So, what does high availability really mean? From a user's point of view, it means an application that is always available to perform the user's transactions. In other words, even though one or more of the back-end system becomes unavailable, the application may be architected in such a manner that these failures are not visible to the end user. High availability is a relative term – for some applications, there is high availability only if the site is never down – or zero downtime. For some applications it should be available, without interruptions – during the normal business operations. How and when a system is called highly available is typically defined by SLA – Service Level Agreements. Each organization may have its own system level agreements for the nonavailability of the applications during certain maintenance windows.

Transparent Failover – A highly available system does not necessarily imply that failover is always transparent. In other words, when one of the components within the application fails, the subsequent requests get routed to other available servers, thereby making it highly available. However it is possible for users to lose their sessions. There may even be a loss of data during the failure and things of that nature. In these cases, even though the system is available for subsequent requests, none of the above scenarios truly represents a transparent failover within an enterprise application.

Fault Tolerance – Fault tolerance is the system's ability to keep the system available and maintain data consistency in case of failure in one or more of its software components.

Clustering – Perhaps the most common term used in conjunction with HA, clustering is essentially a service provided by application servers in which multiple servers run as a group. The clustering service is responsible for load balancing the user requests and also for rerouting the incoming traffic to available servers should one of the servers in the cluster become unavailable.

Recovery – The ability of the system to recover

Author Bio:

Sudhir Upadhyay is currently with Architecture and Shared services at JP Morgan Chase where he is an application architect. Prior to joining JPMorgan, he was a principal consultant with BEA Professional Services where he helped customers design and implement enterprise J2EE solutions. He is a BEA Certified WebLogic Developer and a Sun Certified Java Developer.

Contact:

sudhir.x.upadhyay@jpmorgan.com

BEA's Workshop can be even more amazing for the phone.



One Application.
Web. Voice. It's Easy.

AppDev™ VXML for BEA's WebLogic™ Workshop

Delivering Web applications to phone users is as easy as clicking a mouse.



For additional information:

SandCherry, Inc.
1715 38th Street
Boulder, CO 80301

+1 (720) 562-4500 Phone
+1 (866) 383-4500 Toll Free
+1 (720) 562-4501 Fax

Info@sandcherry.com
www.sandcherry.com

Download
30 Day Free Trial
www.sandcherry.com



itself from a failure condition as well as to bring the state of the objects into their pre-failure condition.

Failure Points

In an enterprise application, there are several points of failure – both within each individual component and at integration touch points with other components and interfaces. As the number of interfaces increases, the failover and recovery procedures become increasingly complex due to interdependencies involved. Although each component within the application may have recovery procedures for itself, the problem arises when one component or a combination of components within the system fails simultaneously.

So, a first step in designing a highly available enterprise system is identifying all of the possible permutations and combinations of failure points within the system. The identification process should involve all of the components within the system – application, network, hardware, and database – each and every component involved. As one begins to create such a matrix, it becomes overly complicated as the number of components and products that are mixing increases. This article will serve to focus failover and recovery of components and applications within the WebLogic Platform.

WebLogic Platform

WebLogic Server – WebLogic Server provides a number of J2EE services that include Java Messaging Service (JMS), Transactions (JTS/JTA) and Security, Servlet Engine, EJB container, and so forth. When designing enterprise applications for high availability it is crucial that architects have a good understanding of how these services failover when one of the instances hosting them goes down. A thorough understanding of the failover and recovery procedures of these services is also critical in automating

the recovery of a downed instance or a system.

Clustering Overview – Perhaps no discussion on WLS high availability is complete without touching on WebLogic Clustering. An application that is deployed homogeneously in a cluster is generally highly available. When one of the server instances goes down, the request is routed to the other available server in the cluster. In fact, the concept of the cluster is not new to J2EE architects and WebLogic Server administrators. Readers of this article are encouraged to review the WebLogic Server clustering documentation (<http://e-docs.bea.com/wls/docs81/cluster/index.html>). Essentially, most of the services provided by WebLogic can be categorized as either clusterable (i.e., multiple instances of these run on different services on multiple servers), or nonclusterable (there may be multiple instances of these services configured, but only one is active at a given time within the cluster). More information on which services can be clustered is provided in WebLogic Server documentation. Table 1 lists the clustered and nonclustered services in WebLogic Platform 8.1.

WebLogic Platform Deployment Architecture

Before diving into specific details of the strategies used to make applications highly available, let us recap the overview of the WebLogic Platform deployment architecture from the documentation. (<http://edocs.bea.com>).

The basic administrative unit for a WebLogic Server installation is called a domain. A domain is a logically related group of WebLogic Server resources that are managed as a unit. A domain always includes at least one WebLogic Server instance called the administration server. The administration server serves as a central point of contact for server instances and system administration tools. A domain may

also include additional WebLogic Server instances called managed servers.

Administrators can configure some or all of these managed servers to be part of a WebLogic Server cluster. A cluster is a group of WebLogic Server instances that work together to provide scalability and high availability for applications. A managed server in a cluster can act as a backup for services such as JMS and JTA that are hosted on another server instance in the cluster. Applications are also deployed and managed as part of a domain.

Failover Scenario

While the details of the failover and recovery procedures would be covered in the next part of this article, a brief overview of what happens when a server instance in the cluster crashes may be useful. Figure 2 represents the possible activities that happen when a WebLogic instance goes down and all of the services are failed over to another available instance.

As indicated in the diagram, one or more of the following operations may be performed to achieve failover if one of the instances goes down.

- All of the new incoming HTTP requests will be rerouted to the other available servers in the cluster by the Web server plugins. Web server plugins detect that a server instance has faulted and dynamically update their list of available servers. The subsequent requests are then routed to the appropriate server (mostly the secondary server) from among the available servers.
- All of the new requests for EJB/RMI will be rerouted to the available servers in the cluster by the EJB/RMI client-side cluster-aware stubs. Essentially, the stubs generated by the EJB/RMI compiler are aware of all of the available servers in the cluster. When a request to a server fails, the stub intercepts the exception and, depending on the type of the exception (such as network exceptions), the stub may redirect the request to any other available servers. In the case of stateless session beans the request may be routed to any available server, while in the case of stateful EJB the stub sends the request to the secondary server, which is the location to which the primary server replicates its state.
- Internal requests for JMS are routed to other available servers in the cluster. In the case of JMS, if the destination is

TABLE 1

Object Type	Clustered	Highly Available	Failover
JSP/Servlets	Yes	Yes	Yes (with session replication turned on)
EJBs	Yes	Yes	Yes (handled via replica-aware stubs)
RMI Objects	Yes	Yes	Yes (handled via replica-aware stubs)
JMS Destinations	Yes	Yes	Yes – Producers automatically failover Consumers need a reconnect logic
JDBC	Yes	Yes (with Multi pools)	No (failover possible with Oracle RAC)
File Services	No	No	No
Timer Services	No	No	No

Clustered and nonclustered services in WebLogic Platform 8.1



CONFERENCE:
AUGUST 8 – 11, 2005

EXPO:
AUGUST 9 – 11, 2005

Moscone Center West
San Francisco, CA

WHERE **open minds** MEET > >

explore > > analyze > > gain > >

LinuxWorld Conference & Expo is the world's leading and most comprehensive event focusing on Linux and Open Source solutions. At LinuxWorld, see and learn how to best leverage the technology for your organization.

- > **Explore** your options on the exhibit hall floor, which features the world's leading hardware and software vendors.
- > **Analyze** the latest Linux and Open Source technology and discover how companies across the globe can show you how to achieve higher profits and increase productivity.
- > **Gain** knowledge about best practices and solutions by attending LinuxWorld's outstanding educational program.

It's the Linux & Open Source event you can't afford to miss!



linuxworldexpo.com

> Register Online With **Priority Code: D0109**

PLATINUM SPONSORS



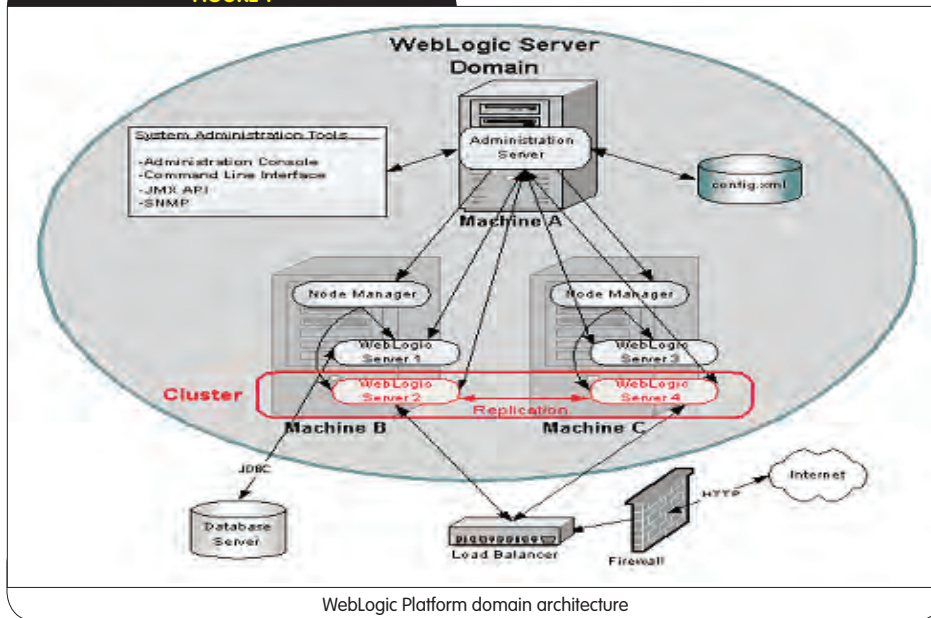
created as distributed destinations and contains physical members on the any of the available servers, the producer of the messages may continue to send messages without any interruption. The consumers, however, may need to reconnect to available members by incorporating logic within the Exception listeners. In

the case of an MDB, the container may provide the logic for reconnecting to the destination.

- The JMS server can administratively migrate to another available server. The migration of the JMS server assists in bleeding messages from the queue that went down with a downed server instance.

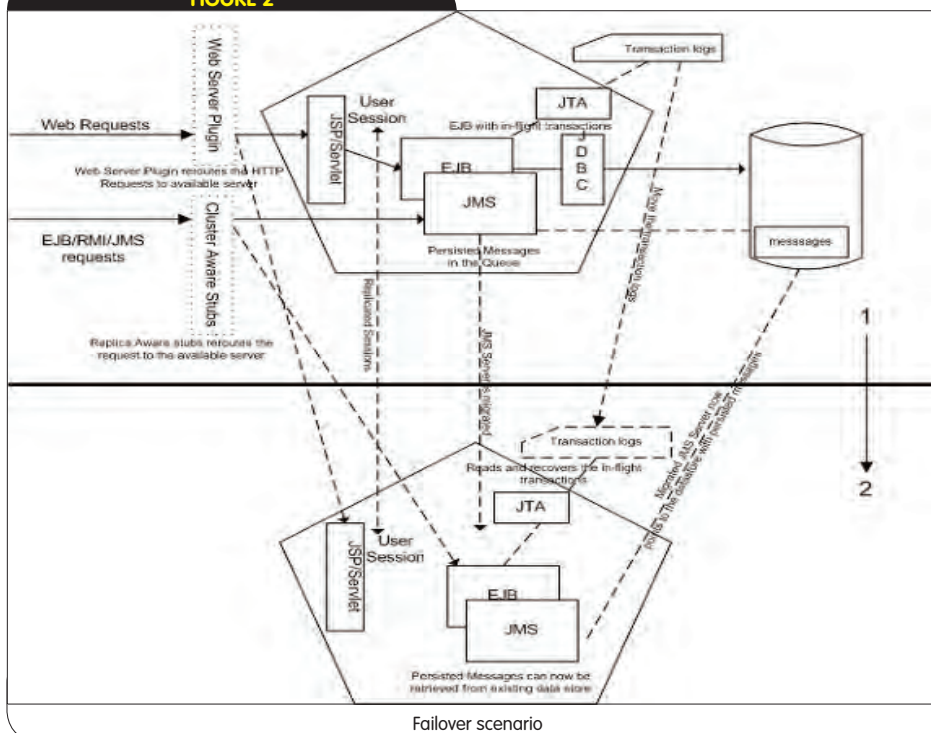
- Any in-flight transactions are handled as per the JTA specifications. Essentially, the administrator can move the transaction logs from the failed instance to another available instance. The Transaction Manager within the application server then attempts to complete those transactions based on the tlog entries.

FIGURE 1



WebLogic Platform domain architecture

FIGURE 2



Failover scenario

Highly Available Deployment Strategies

Having read the high-level introduction about a typical failover scenario, let's now discuss the possible high-availability strategies for deploying applications on the WebLogic Platform. It is no secret to architects that the bottom line in achieving high availability is avoiding single point of failure (SPOF). On the face of it, it appears one would always want to avoid SPOE, but in reality there many constraints in the application design, vendor product architecture, and deployment topology that force the infrastructure with one or more single point of failures. The following sections explain what the infrastructure team should do to avoid these in their deployment architecture.

Hardware Load Balancer

One of the first entry points to the application is via the hardware load balancers. Hardware load balancers have gone beyond simple load balancing/distribution of incoming HTTP traffic and now provide sophisticated algorithms to distribute IP traffic more efficiently, and provide a much higher level of fault tolerance. WebLogic clusters can use any of the sophisticated load balancing/failover algorithms supported by the hardware load balancer. Hardware load balancers are generally more fault resilient than Web server plugins. For a detail description of the capabilities of a hardware load balancer, readers should see the vendor-specific documentation and the BEA documentation on configuring hardware load balancers with WebLogic cluster (http://e-docs.bea.com/wls/docs81/cluster/load_balancing.html#1026240).

Web Server Farm

In some of the conventional Web facing applications, the Web servers front-end the application servers. Web server runs a Web server plugin that redirects/routes

Intersperse Manages the Risk in SOA Deployments



Over the past 25 years, enterprise application systems have advanced tremendously, increasing functional scope, geographic distribution, degree of interconnectedness, and speed of information delivery. While this evolution yielded enormous productivity benefits and competitive advantages, it also caused IT complexity to explode, as we've moved through computing phases: from the mainframe, through client/server, into web apps, and finally on to SOA applications. In addition to this increased complexity, each of these phases was shorter than the last, allowing IT teams less room for error (see Figure 1).

This all results in a relentless increase in risk – risk that has always been mitigated by IT management tools. New management solutions emerged each time we moved into a new phase, providing features and functions targeted to solve the specific needs of that phase.

This remains true today. Existing management tools were designed to address the needs of previous generations, and not to address the problems generated by today's advanced computing systems, which are increasingly built on J2EE and based on SOA principles.

Despite their many benefits, today's J2EE-based SOA application systems bring new challenges. Among other things, these systems:

- Are highly distributed, leveraging the Internet to coordinate business processes within and between enterprises.
- Have many more points of failure, as systems break down into granular software components, services, and interfaces, with dynamic, complex interdependencies, and relationships, any of which can cause failure either individually or through its interactions.
- Blur the line between application development and application integration, as systems are increasingly "built to integrate" with an emphasis on service reuse.

The BEA WebLogic Platform, from Workshop to WebLogic Integration, provides the market's most comprehensive toolset to design, build, and deploy SOA applications. The last piece of the production SOA puzzle is comprehensive, proactive management of SOA applications. This is a critical piece, as traditional management tools are insufficient to manage distributed, integrated, cross-application and cross-enterprise business processes – they were designed for another time and another world.

1980s Mainframe	Early 1990s Client-server	Late 90s-2000s Distributed Web Apps	Today SOA Apps
Market = Mainframe • IBM, Tandem, Fujitsu, Amdahl, Unisys	Market = Client-server Apps • SAP, PeopleSoft, Oracle Apps, Siebel, JD Edwards	Market = Web Apps • WebLogic, WebSphere, ATG, Oracle, iBos, Torcat, JRun • Client-server apps rewritten to leverage app servers	Market = SOA Apps
Mainframe = Monolithic and distributed	Applications = Monolithic (not componentized) and distributed Platform = The OS	Applications = Distributed and componentized Platform = The app server	Applications = Distributed, componentized, flexible and unpredictable Platform = The service
Principal Building Block = The application			Principal Building Block = The service
IT complexity and management costs increased dramatically			

Intersperse Manager is the only product designed specifically to provide production management of SOA applications. It discovers, monitors, and manages J2EE-based SOA applications deployed on Application, Integration, Process, or Portal Servers, and it ensures the continuity of the vital, integrated business processes that run today's enterprises. By simultaneously managing both vertical applications and horizontal integrations, Manager delivers proactive production management of service-oriented BEA applications. Intersperse extends management to the complete BEA WebLogic Platform, including WebLogic Integration and WebLogic Portal.

Intersperse Manager is truly a production management solution, leveraging the increasingly ubiquitous Java Management Extensions (JMX) standard. JMX facilitates standardized discovery, monitoring, and management of components, applications, and services, as well as enabling hot deployment and removal from already running systems. This is in contrast to some tools today that have defaulted to the use of invasive, application-altering byte-code instrumentation as their means to gather management data.

To fully reap the benefits and mitigate the risks of J2EE-based SOA deployments, businesses need a new kind of management tool, one designed specifically to serve the needs of these environments. Intersperse Manager is the only management tool designed to help master the rapidly growing complexity of SOA business systems. It gives developers, operators, and analysts comprehensive visibility into all relevant tiers, tools to proactively monitor and analyze system performance in business contexts, and the control to automatically correct problems in production. As organizations evolve their business systems to BEA-based SOAs, they cannot afford to be without Intersperse Manager.

the HTTP traffic. While a single Web server can distribute the traffic to multiple back-end application servers, in this case the Web server itself becomes a single point of failure. Therefore, one of the common strategies to avoid this scenario is to create a Web server farm. Typically, the load balancers are configured to maintain a sticky session with the Web server to which the first request from a given client was routed. In addition, the Web server plugin also maintains stickiness to the server that the first request was routed to. The Web server plugin maintains a list of available back-end servers, along with the primary/secondary pair for that particular client. In case of failure of one of the back-end application servers, the Web server plugin routes the request to the appropriate available server. Regardless of which Web server the request gets routed to, the Web server routes the request to the correct application server by inspecting the cookie in the HTTP header.

Number of Servers in the Cluster

Several deployments are composed of the two-node cluster. While this is sufficient for low-volume, nonmission-critical sites, it is recommended that a fault tolerant cluster contain a minimum of three servers. This is because during the HTTP session replication, only the deltas are replicated from primary to secondary, and only when HTTP requests are made. So, consider a two-node scenario:

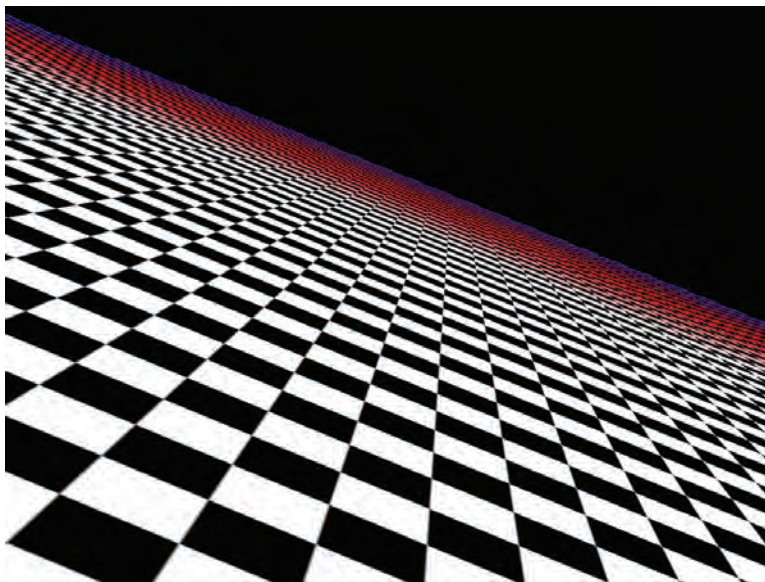
- Node A goes down, leaving Node B unpaired, hence no session replication for any activity during this duration
- Node B is now brought up, but there is no HTTP activity or the user does not modify the session values, so no session replication occurs
- Node A now goes down and any values added to the user session that had occurred during the interval when server B was down can be lost

So, it's always a good practice to have a minimum of three servers in a cluster to have a highly available user session.

Apart from avoiding user session loss, this architecture provides a higher redundancy.

Redundant Administrative Server Configuration

It is evident from the deployment architecture diagram that single point of failure in the platform is the administration server in a domain. Although failure of the administrative server does not have any impact on the availability of the application deployed on managed server instances, no further administrative functions can be performed. In addition, if the administrative server is down, no monitoring activities can be performed using the WebLogic console. To quickly recover from an administration server failure, administrators should maintain a backup of the configuration



file (config.xml) and other associated files to quickly restart the admin server on any other host.

Managed Server Independence

In the event the administration server cannot be brought up due to unknown technical difficulties, it is always a good idea to configure the managed servers to start in Managed Server Independence (MSI) mode. Essentially, when a managed server is brought up, it attempts to connect to the admin server; if the admin server is down or unreachable, it looks for a file named *msi-config.xml* and boots itself by obtaining the connection from this file.

Startup Modes

To further reduce the time required to bring the managed servers, additional servers can be started in a STANDBY mode. Although the typical start-up time for a managed server is less than a minute, depending on the number of application deployments, the initial number of configured EJBs, and the number of start-up classes loaded at the start-up, it can take several minutes before the server reaches the RUNNING state. In the STANDBY state, a server has initialized all of its services and applications; it can accept administration commands, and can participate in cluster communication. However, it does not accept requests from external clients. In such cases, if a managed server is configured to start in STANDBY mode, it can be made available with significantly less down time.

Process Monitoring

Invariably in most of the deployments, administrators have scripts that continuously monitor the Java processes. In some cases these scripts merely check if the process is running and in other cases it performs a WebLogic utility PING to ensure that the server is in fact responding. An advanced level of monitoring may also be enforced to verify the availability of the application.

Node Managers

Node manager is a Java utility that runs as a separate process from WebLogic Server and allows administrators

to perform common operations tasks for a managed server, regardless of its location with respect to its administration server. While use of the node manager is optional, it provides valuable benefits if your WebLogic Server environment hosts applications with high-availability requirements. Essentially, node managers monitor WebLogic Server's health and automatically restart the servers if they go down.

External Cluster Options

In addition to the services provided by the server and homegrown scripts, hardware clustering services can also be configured to monitor the server health and restart it. One big advantage of hardware-

clustering services is the ability to start the servers on different machines using the virtual IP addresses. For example, Veritas cluster service can be configured to start the WebLogic Server instances on a separate machine if a configured number attempts to start the server on a node that has failed.

Using a Clustered File System

Use a common file system for all of the servers in the cluster. This will ensure that each of the servers has access to the transaction log files in the event another server needs to be started on a different machine. While a common clustered file system is valuable in ensuring that the transaction logs are available to all of the servers, it is strongly recommended that this file system be mirrored.

Avoid Pinned Services

While clustering has been widely used for a few years now, there are quite a few applications out there that are still running with singleton/pinned services. While in some cases this may be because of the business requirement, in others it may be because of a not-so-well-thought-out design. The architects must review the application design architecture to avoid using those services that are not cluster aware.

Session Replication

The value of clustering Web applications lies in the ability to replicate the user sessions to a secondary server in the cluster. Unfortunately, there are still many application developers who do not take this into consideration and tend to store objects in sessions that cannot be replicated or that construct session objects that cause significant performance overhead during session replication. Application developers should review the documentation on best practices for creating replicable user sessions.

Replication Groups Within a Cluster

If there are multiple servers spanning more than one physical machine in a single WebLogic cluster, by default WebLogic chooses primary and secondary servers for session replication on different physical machines. Administrator can, however, configure replication groups as well to have better control over the failover policy.

Availability of Critical Applications

It is not necessary for an application to be unavailable only when the application server is down. WebLogic server by default does not assign any priority to any particular request. So, if there are multiple applications deployed on a single application server, an offending application can consume most of the resources such as execute threads, database connections, and JMS resources. In such a scenario, other applications deployed on the same cluster may be impacted – and may even become unresponsive – thereby becoming unavailable. While such issues in the applications should be resolved before they are moved into production, it is a good idea to provide some degree of isolation by allocating a custom execute queue for critical applications. This would shield the critical applications from being impacted by other applications.

JDBC High Availability

JDBC provides a physical connection to the underlying DBMS. The connection holds several parameters: transactional context, prepared statement caches, and so forth. These parameters cannot be failed over to another connection and therefore, failover of JDBC connections is not supported. However, if you have replicated, synchronized database instances, you can use a JDBC multi pool to achieve high availability of database connections. In such an environment, if a client cannot obtain a connection from one connection pool in the multi pool because the pool doesn't exist or because database connectivity from the pool is down, WebLogic Server will attempt to obtain a connection from the next connection pool in the list of pools. While it is not common to see the database servers going down, the true high availability for JDBC can be achieved by using Oracle Real Application Cluster (RAC). One should, however, be aware of some of the limitations (such as failover during XA transactions) that exist in the RAC implementation.

JMS High Availability

High availability for JMS is achieved by using cluster-wide deployments of JMS connection factories and distributed destinations. Distributed destinations are virtual destinations with actual physical members on one or more managed servers that are part of the cluster. The clients

perform a JNDI lookup using a distributed destination name, and the JMS server performs a routing of the requests to the underlying JMS objects. If one of the nodes that hosts the JMS physical member destination goes down, the JMS front end within the server routes additional messages to any other available member destinations. The consumers, however, can have a reconnect logic in the *onException()*, if *ExceptionHandler* has been defined for the consumer session.

Zero Downtime Architecture

Several enterprises require their customer-facing sites to be up even during the application upgrade window. In this scenario, a commonly used practice is to control the traffic via the load balancers and two-cluster mode. Essentially, there is a primary cluster and a back-up cluster. A new application version is deployed on the back-up cluster and the load balancer is enabled to route all of the new requests to the back-up cluster. In the meantime, the servers in the primary cluster are issued a graceful shutdown command. With graceful shutdown, the servers continue to process the already established sessions while not accepting any requests for new sessions. After all of the sessions on this primary cluster have either been invalidated or have expired, the servers are shut down. At this time, all of the new requests are routed to the newer version of the application in the back-up cluster.

Conclusion

While administrators can make every attempt to design a highly available deployment solution, it is equally important to design quick and reliable recovery procedures without impacting the data integrity. Several options exist for such procedures. The recovery of some of the services is automatic and out of the box, whereas other services need to be manually migrated. In such cases an administrator can automate these via scripting and based on the specific business rules of the organization.

The second article of this series will discuss the details of recovery of a failed server and critical components such as JMS and JTA.

References

- BEA product documentation: <http://edocs.bea.com>

Services-Oriented Architecture and Services-Oriented Development of Applications

A STRATEGY FOR TRANSITION



By Steve Buzzard

Author Bio:

Steve Buzzard is currently working as a J2EE principal architect with Anexinet Corporation (www.anexinet.com), a leading systems integration firm headquartered in Philadelphia, with offices in New York and Washington D.C. Steve has over 19 years of experience in professional software development and has been working almost exclusively with the WebLogic Technology Stack since late 1998.

Contact:

sbuzzard@anexinet.com

Services-oriented development of applications (SODA) is an important development model for enabling organizations to reorient business processes in the transition to a service-oriented architecture (SOA). This article describes one such approach.

Services-Oriented Development of Applications (SODA)

Gartner refers to SOA and SODA as foundational elements of future computing. SODA is a new style of developing software, designed to work specifically within the SOA paradigm. SOA represents a collection of loosely coupled, coarse-grained, heterogeneous components that can be easily snapped together using Web services. The result is enhanced developer productivity, code reuse, and business agility. The SOA Blueprints (www.middlewareresearch.com/soa-blueprints/), an industry-derived set of best practices and associated reference implementation, is an excellent resource for those looking to move to an SOA.

SODA is centered on the creation and assembly of services and service contracts first, deferring the design and implementation of the objects and components that realize the services until after the coarse-grained service contracts have been ironed

out. SODA developers focus more on the process flow within and between applications, and less on the code that creates the underlying system. For a brief and to the point description of SODA, see www.serviceoriented.org/soda.html.

Pain Points

The issues described below are certainly not all-encompassing, but are meant to exemplify the issues that often impact a development process (particularly when undertaken by larger enterprises with multiple development teams), and which might be alleviated by transitioning to a SODA-based development approach.

Dependencies / Bottlenecks

Bottlenecks related to the ordering of dependencies are usually the biggest issue. Developers are often waiting for a dependent item to be designed and/or implemented before they can continue with their tasking. One of the issues is the division of labor in a multi-team environment. A standard division of labor often employed in this environment consists of having different teams working on the same vertical slice of functionality in different tiers, causing these dependency-related bottlenecks to spring up.

Management and mitigation of dependency-related issues is crucial to ensuring that developers can continue to be productive throughout development iterations; however, it isn't possible to

“manage” a dependency issue away if many of the pieces are being developed in parallel (which is often the case with a fixed deadline).

Creation of the data model for each of an organization's releases (iterations) oftentimes begins roughly in parallel with the development of the software that interacts with it and not much earlier, which is not ideal but not always possible to avoid. Similarly, the domain model for a release may not be completed prior to a development iteration that will be using it. Proper planning in building out the data and domain models first is critical to avoiding these issues but if the project is already well underway, it is what it is (“next time, we’ll do it right”). These issues result in a bottleneck for the business services, object/relational persistence mapping, and enterprise application integration (EAI)/business-to-business (B2B) developers, who require both data and domain models to be completed before much of their work can commence. The dependency problems can quickly cascade.

Division of Labor

Development teams are often divided horizontally by tier (especially those that are geographically dispersed). One team may be working on the presentation tier and the other on the business tier. This division often exists out of necessity due to the fact that off-site teams may not have access to the back-end systems with which the business tier needs to interact. Development of the presentation-tier functionality does not require access to these systems (at least during the design and initial implementation phases of the iteration). This division of labor has its problems, though, in that the development of discrete pieces of functionality (use cases) is being performed by separate (again, often geographically separated) teams. These teams often communicate through interfaces to business services and the data transfer objects (DTOs) (<http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html>) exposed through them – a standard Java 2 Enterprise Edition (J2EE) best practice. The DTO structure and content often changes during a development iteration, as both the presentation and business services developers dig deeper into their designs and subsequent implementations. In addition, the structure/content of the DTOs is (mostly) driven by the structure of the domain model, and as it evolves, so do they. These issues could be lessened somewhat if the same team were working on an end-to-end slice of functionality (one team would be in control of the structure/content of the domain model, DTOs, business service interfaces, and the associated presentation). The key roadblock to going down a road with this approach is the capability to consistently simulate interfaces with the back-end systems, so that an off-site team can be given a full vertical slice. When teams *are* collocated, such back-end systems often have usage restrictions, especially when the back-end system is connected to and/or owned by another project, department, or business.

How Can a SODA Approach Help?

All resources are considered to be as services with SODA (from UI components on the front end to interaction with an external business partner on the back end). The primary development activity for the vast majority of developers here is the orchestration of those services. A small number of J2EE and integration gurus are responsible for the development of the implementation and/or extension of services in the case where the chosen SODA tools do not provide them off-the-shelf.

Dependency Abstractions – Abstraction is the key to SODA (along with self-description and discovery): abstraction of services implementations, abstraction of concrete data typing, and abstraction of activities such as the plumbing behind user interface and business services development.

Refocusing the Division of Labor – A SODA approach provides enough automation, implementation abstraction, and simulation support to allow for a more logical, vertical division of labor. Each team can take on a full vertical slice of related functionality and not be consistently dependent upon another team (who may be taking a different approach), as is the case with the current horizontal division.

Enabling Business Process Design – SODA allows the designer to develop both the presentation and business services functionality graphically, concentrating on the process flow, from Web page to back-end integration interaction. Services can be graphically configured out of the box, as can data transformation. The SODA tools behind the scenes automatically generate implementations of enterprise architecture and J2EE design patterns.

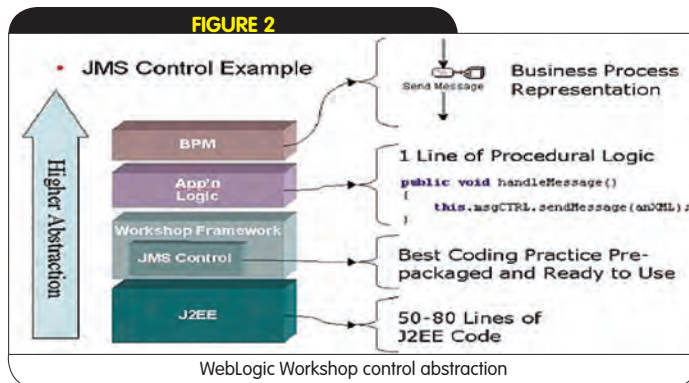
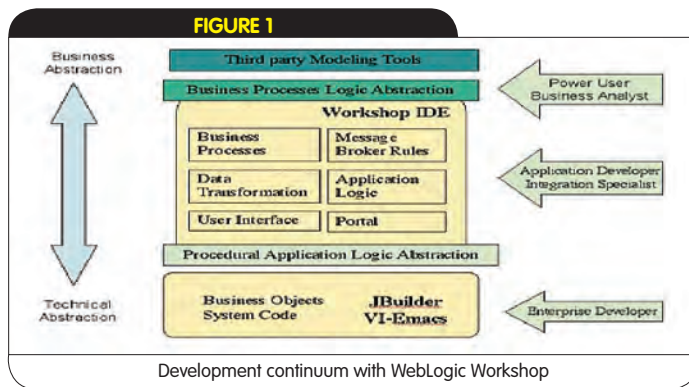
Service Publication and Discovery – SODA tools usually provide (or interact with) a Universal Description, Discovery, and Integration (UDDI) server (or other similar registry), thereby allowing the easy publication of services and their associated metadata to a centralized registry. Conversely, these tools provide for the discovery of services previously published to the registry. This allows a designer to discover and reuse a service with a couple of clicks of the mouse. If the service isn't available and the designer needs to configure one, he or she can then publish that service for others to reuse. The J2EE and Integration experts, who might need to build a service from scratch or extend an existing one, will also use the SODA tools to do this, providing them the same accessibility to the UDDI registry. EJBs and POJOs can be constructed using the chosen IDE and the resulting JARs imported into the SODA tools. This process can be glued together using Apache's Ant (<http://ant.apache.org/>).

Leveraging Diverse Skill Sets/Varying Levels of Expertise – SODA allows developers of all backgrounds who have an understanding of the business domain and use cases to quickly be productive in orchestrating use case process flows and formulating the data content and transformations that take place as the flow progresses. Presentation experts will perform orchestration of presentation flows. Integration/EAI experts can, in parallel, configure the services and data transformations that are required to wire together process flows between systems and businesses. J2EE gurus can work on custom implementations of discrete pieces of functionality where this is not provided with an out-of-the-box service. They can use the SODA tools to expose these implementations as services and then publish them for others to discover.

Integrating SODA Process

Transitioning the development team to a SODA approach will require not only a process change but a cultural one as well. The key changes required are described in the following subsections, but the overriding emphasis in each case must be:

1. A re-thinking of the way in which the domain, business, use case, and presentation components are realized (designed and implemented). In traditional processes, there often is no distinction between business services and use case services, while presenta-



tion services are not really thought of as services at all. In order to achieve domain independence (domain pluggability) while at the same time achieving business independence, the use case/business services distinction must exist. In addition, the components that make up both the Web page presentation as well as the page navigation are considered to be first class services in the SODA world: the orchestration of page navigation is really no different (mechanically) than the orchestration of business process flow. The orchestration and assembly of domain, business, use case, and presentation service layers is one of recursive composition:

- A presentation service is realized by the orchestration of action services within a page flow, their interaction with use case services, and finally, passing off control to one or more view services for rendering on the page.
 - A use case service is realized by the orchestration of other use case services, as well as lower-level business and domain services.
 - A business service is realized by the orchestration of other business services, as well as lower-level domain services (across more than one domain).
 - A domain service is realized by the orchestration of other domain services (provided they are in the same domain).
2. A focus on the use of SODA-enabling tools to provide and enforce the abstractions needed for virtually all domain and application interaction.

J2EE/Integration Component Design – Development staff whose J2EE and integration technology skill sets and experience outweigh their knowledge of the business and business modeling should be

assigned to build out and/or extend lower-level components upon which all other services are built.

Domain Service Design – There will need to be a strong emphasis on domain service design (that is, services that interact with the domain model in a use-case independent manner). It is important that the domain services be narrowly scoped, cohesive, and discrete. The set of services tied to a particular domain should allow that domain to be used independently of other domains.

Business Service Design – In the cases where services are desired that are use case-independent and yet require interaction with lower-level services across domains, a separate layer of services should be realized. This layer should be distinct and separate from the individual domain services, yet it should not be melded into the use case services tier. Business services, for the purposes of this document, are services that span individual domains while remaining independent and reusable across use-case realizations. Use case services, meanwhile, are application specific, and are not meant to be reused across systems. The relationship between use case services and business services is analogous to the relationship between the application service pattern (www.corej2eepatterns.com/Patterns2ndEd/ApplicationService.htm) and the session facade (www.corej2eepatterns.com/Patterns2ndEd/SessionFacade.htm).

Use Case Service Design – These services realize specific application use case scenarios, by orchestrating other use case services and lower-level business and domain services. Strong understanding of the particular application requirements is required here.

Presentation Service Design – These services orchestrate the page flow and render the resulting information on a Web page. Strong skills in human factors, HTML, and an understanding of an application flow from a page navigation perspective are required here.

Developer Roles – Successful integration of SODA requires that the developer roles are firmly defined, based on evaluated skill sets, and that the tasking truly reflects their role in the process, as described in the immediately preceding subsections.

Division of Labor – The division of labor for the development teams in a SODA environment should be made logically, based on related/dependent groupings of use cases.

Service Publication and Discovery – The use of a UDDI browser to discover and use services will need to be taught, as will the ways in which a service can be published and versioned (automation of this process is, as for most things, preferable).

Configuration Management – The better SODA tools integrate well with an SCC-compliant source control system, so configuration management, from this perspective, will not change fundamentally. The change comes in dealing with published and versioned services. An SOA-based system often looks up and uses services dynamically. The UDDI registry of services, therefore, should be separated by environments (e.g., development, integration, QA, production), much as the runtime system itself is. The development registry should also be synchronized with the source control system.

Integration with existing software – The existing component-based software will live side-by-side with the new SODA-based software in the source control system. The EJB and utility JARs, as well as existing Web applications, can be imported directly into most SODA tools. Most SODA tools also provide command-line equivalents to the functionality provided in the GUI. This is a standard requirement so that the system as a whole (whether developed using SODA techniques or more “traditional” component/object-based development) can be built using tools such as Ant.

Tools/Integrated Service Environment (ISE)

Within SODA, there is a concept called ISE. ISE refers to the tool suite needed to actually implement the concept of SODA and is the logical evolution of current IDEs. SODA is very dependent on the existence of a robust set of tools to handle interface abstraction, publication, discovery, simulation (mock implementations), and data transformation. It is key that the SODA-enabling tool(s) chosen support these important concepts (SODA's key differentiators). These differentiators are described in the subsections below.

Out-of-the-box Services Library – There has to be an extensive library of ready-made services for common enterprise application functionality (for the presentation, business, and integration tiers). These services must also be readily extensible and configurable.

Services Creation – Obviously, you have to be able to create services easily through graphical wizards. Existing or third-party components should be easily exposable as services through the tool.

Services Assembly (orchestration) – Perhaps the most important feature, in terms of developer productivity, is the ability to assemble (orchestrate) services into business process flows in a graphical, intuitive environment. This must include presentation (page/action navigation) as well as business services flows. Declarative exception, security, and transaction demarcation are also required here.

Services Repositories (tied into source control) – A centralized services repository, that uses a standard such as UDDI, is required. The capability, through the tool, to tie this repository to the project source control system is also desired, although this can be accomplished through an external tool, such as Ant.

Lookup and Dynamic Discovery – The services repository must provide for service browsing, lookup, and dynamic runtime discovery. Use of a UDDI server as the repository generally provides such features.

External Interface Specification (WSDL or other XML Schema)

– Services in a SODA environment are generally exposed as SOAP-based Web services, with this feature being an inherent part of that standard. The tool should provide this same capability for services that are not exposed as Web services.

Service Translation (versioning compensation) – The ability to dynamically detect the version of a particular service versus the version of the client of that service and to provide data transformation between the two, if necessary, is desired.

BEA WebLogic Workshop and Other Tools

BEA WebLogic's Workshop (<http://dev2dev.bea.com/wlworkshop/index.html>) ISE is probably the most complete in terms of enabling SODA within a J2EE environment. It provides support for the key differentiators and allows for a separation of concerns between the interfaces, orchestration of those interfaces, and the underlying implementation. There are certainly many other tools in this space. Some, like Collexa's Web services orchestration product, are complimentary and integrate well into Workshop. Others directly compete. Workshop does have some potential negatives: enabling many of the features currently ties you to the BEA WebLogic platform. Many features have been submitted and accepted by the Java Community Process as Java Specification Requests. With the Beehive initiative well underway in the Apache incubator, along with the associated Eclipse plug-in, Pollinate, vendor lock-in should disappear here. The article "Bridging the Gap: BEA WebLogic Integration" (<http://dev2dev.bea.com/pub/>

<http://dev2dev.bea.com/pub/>) details WebLogic Workshop in the context of SODA.

Proof of Concept (POC)

SODA can provide potential for significant productivity enhancement; however, it is not without its downsides:

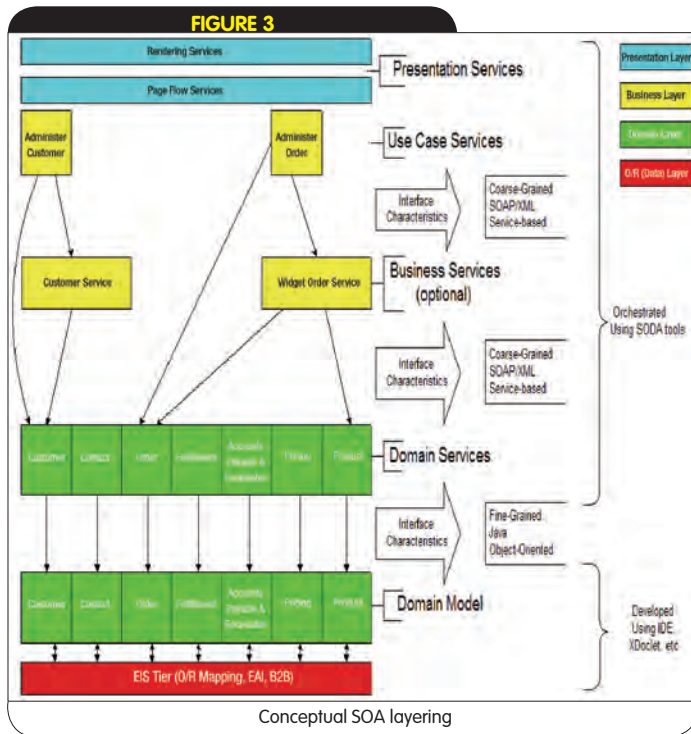
- The lack of current standards for tool support dictates a degree of vendor lock-in.
- SODA places much of the development burden on tools. This implies that the tool is adept at the generation of lower-level plumbing code that is either completely hidden from the developer or, if available for customization, can be easily maintained.
- Is the resulting architecture scalable and performant? Is it extensible?
- There needs to be significant effort given to training the development staff on the SODA tool(s) chosen. The shift in the process by which one develops software is not insignificant.

A POC is required to demonstrate/prove a SODA approach. This POC must be extensive enough to demonstrate/prove at least the following concepts:

- Services-based design paradigm shift. We especially need to dig into the various service layers (realize/utilize at least one domain, business, use case, and presentation service).
- Multi-team development. This includes source control integration and the assignment of specific roles related to SODA (presentation designer, use case designer, business/domain designer, J2EE expert, integration expert, etc.).
- A use case scenario realization that implements a full vertical slice. It is important that this use case scenario touches on all tiers, including (perhaps especially) the integration tier.
- Demonstration of the assembly and orchestration of the following service types:
 - Presentation and presentation flow
 - Basic Web services
 - Business process (workflow)
 - Messaging
 - Back-end integration (EAI)
 - Security
- Integration with lower-level J2EE components
- Integration with existing functionality
- Demonstration of the publication, and subsequent look-up/discovery of services and associated metadata (documentation)
- Demonstration of a development, build, deploy, test, and release cycle
- Demonstration of service simulations, to show alleviation of dependency bottlenecks

Service-Oriented Architecture and Domain Pluggability

This section discusses the extent to which an SOA can help better enable business domain pluggability. Pluggability, as defined here, is the capability to swap out one domain implementation (business services and associated domain model) and replace it with another. Pluggability requires that the domain implementation only be exposed through well-defined interfaces (the clients of the domain, including other domains, never know about anything but the interfaces). This can be achieved with a component-based approach; however, there are limitations to this:



the clients of the domain are generally bound to particular versions of the component interfaces (at least without jumping through reflection/classloader-related hoops or using some sort of brittle JNDI naming convention).

A domain model, generally, is too fine grained to be directly exposed as a set of services. For this reason, any inter-domain relationships realized within the domain model should be kept to a minimum, in order to achieve pluggability with respect to its clients (the domain services). This obviously implies that inter-domain relationships within the data model are likewise kept to a minimum. The domain model is only exposed to the domain services layer. The business devices (if needed) and use case services layers interact with the domain services. Domain services interact with their applicable domain model and convert the fine-grained, object-oriented interface of the model to a coarse-grained, services-oriented interface. See Figure 3 for a conceptual (example) illustration of how these layers might be defined.

Trade-offs

There are always trade-offs to consider when choosing an architectural approach, and moving to an SOA employing SODA is certainly no different. There are many interpretations of what exactly SOA (the concept of which has been around for a great many years) and SODA really mean. For the purposes of this document, services are defined as coarse-grained interfaces that are often generated by a tool and usually talk XML. These services can be, but are certainly not always, exposed as Web services. The services should be dynamically configurable at both design and runtime, so that a particular service can be reused for many different purposes. The service implementations themselves will be a mixture of custom-built and third party. These services are usually layered and are, for the most part, orchestrated into presentations and business processes. Whenever one talks of XML, layering, and

dynamic configuration, performance and scalability concerns enter the conversation. Likewise, SODA's great reliance on tools raises discussions on vendor lock-in. Reliance on particular toolsets also raises questions about software builds, change management, and deployment, and how these tools interoperate with the infrastructure currently in place.

Vendor Lock-in

One of the key value prospects of SODA is how tools will enable presentation and business process orchestration, (nearly) seamless XML/native language data transformations, design/runtime service configuration, registration, discovery, etc. All this is possible without any real all-encompassing SODA standards (there are, of course, piece-part standards such as Web services/SOAP, BPEL/BPML, WS-*<fill-in-the-blanks>*, etc.). This almost guarantees a certain level of vendor lock-in. The degree of vendor lock-in depends on the tools chosen and the use of those tools.

Performance and Scalability

There is often much concern around the impact to both performance and scalability of an application built on a layered, XML-based SOA. There are questions related to the potential overhead in using something as verbose as XML for intra-application communication and the serialization/deserialization potentially required by both client and service in order to translate the XML to the native language object representation(s). Scalability comes into play as more concurrent users, and larger payloads, interact with the application. Much of this can be mitigated via the judicious design of truly coarse-grained services and the enforcement of communication between well-defined service layers only. Limit long-lived state and ensure that operations are asynchronous where appropriate. SODA tools often help mitigate this concern as well by optimizing the XML transformations (or by natively operating on XML within their presentation and business process flows). In the end, though, there will certainly be more overhead and the early incorporation of application profiling under realistic use patterns must be ingrained into the process going forward to determine where optimization and tuning is required.

Build, Change Management, and Deployment

The tools picked must interoperate with the infrastructure currently in place, or provide the extensibility hooks to easily add this interoperability. Interoperability with the current build (usually Ant) and source control systems, and standard J2EE enterprise application packaged deployment units are all requirements.

Summary

The bottom line is that an SOA, built using the latest SODA-based tools, is certainly not a silver bullet (nothing is). A disciplined SOA approach can help to alleviate dependency bottlenecks somewhat, via data/service abstraction and simulation; however, the dependencies, especially at the domain layer, must still be managed. On the other hand, one of the benefits of an SOA is that the higher-level services can be developed relatively independently, thanks to the service and data abstraction. This allows the presentation and business process orchestrations to be developed independently as well. ●

Register Online
www.eclipseworld.net

Attend EclipseWorld, the enterprise development conference!

EclipseWorld is for enterprise developers, architects and development managers who want to take their company's applications to a higher level!

At EclipseWorld you will:

- Save money and improve developer productivity with Eclipse.
- Master techniques for building high-quality, more secure software.
- Go beyond the IDE to master the wide range of Eclipse technologies.
- Get deep inside Eclipse's open-source architecture.
- Discover the best, most effective Eclipse add-ins and plug-ins.
- Improve team collaboration using Eclipse.

Over 50 Tutorials & Classes To Choose From!



Early Bird
Rates
Expire
July 29!



BZ Media is a member of the Eclipse Foundation.



August 29-31, 2005



The Roosevelt Hotel • New York City

Produced by **BZ Media**

Platinum Sponsor

SYBASE

Gold Sponsors

IT LOG

Exadel

Media Sponsors

SDTimes
The Industry Magazine for Software Development Managers

Software Test & Performance

EclipseSource

Open Source

queue
SOFTWARE DEVELOPMENT
COMMUNITY'S ESSENTIAL SOURCE

JDJ

Software

WebSphere
JOURNAL

Methods & Tools

Extension
MEDIA

EclipseWorld™ is a trademark of BZ Media LLC.
Eclipse™ is a trademark of Eclipse Foundation Inc.

www.eclipseworld.net

Migrating a JBoss EJB Application to WebLogic

MODIFY A JBOSS APPLICATION FOR THE WEBLOGIC SERVER

By Deepak Vohra

The JBoss open source application server is commonly used in the development phase of a J2EE project. In the production phase the commercial BEA WebLogic server is preferred because of its enhanced set of features. Without modifications, an application developed in JBoss does not deploy in WebLogic server.

The deployment descriptors for the WebLogic server are different from the JBoss deployment descriptors. An application may be migrated to WebLogic by converting the vendor-specific deployment descriptors to WebLogic. In this tutorial an EJB application developed in JBoss will be migrated to WebLogic with MySQL as the database.

Preliminary Setup

Download and install the BEA WebLogic server (www.bea.com/framework.jsp?CNT=overview.htm&FP=/content/products/weblogic/server).

Create a server domain. Download the MySQL JDBC driver JAR file (www.mysql.com/products/connector/j/) and the MySQL database server (www.mysql.com/products/mysql/). Develop a Java application or obtain an XSLT utility to transform the JBoss deployment descriptors to WebLogic deployment descriptors with an XSLT.

Without deployment descriptor conversions, an application developed for JBoss does not deploy in WebLogic. In this tutorial, we will migrate an example entity EJB application developed in JBoss to WebLogic by converting the JBoss deployment descriptors to WebLogic deployment descriptors.

The example application consists of a Catalog entity EJB. The EJB's bean class (CatalogBean.java) is shown in Listing 1. The remote interface (Catalog.java) and home interface (CatalogHome.java) are shown in Listing 2 and Listing 3. The entity EJB classes do not need to be modified for deploying a JBoss EJB application to WebLogic. Only the deployment descriptors for an EJB are required to be modified.

Configuring WebLogic JDBC

In this section a JDBC connection will be configured with the MySQL database from the WebLogic server. First a JDBC Connection Pool is

Author Bio:

Deepak Vohra is a Sun Certified Java 1.4 Programmer and a Web developer

Contact:

dvohra09@yahoo.com

configured and subsequently a JNDI data source to access the JDBC connection pool is configured. Add the MySQL database driver JAR file, `mysql-connector-java-3.0.16-ga-bin.jar`, to the CLASSPATH variable of the examples server. The CLASSPATH variable for the WebLogic server is set in the `<BEA>\user_projects\domains\mydomain\startWebLogic` script. Double-click on the `startWebLogic` script file to start the WebLogic examples server. The server gets started on port 7001. Login to the WebLogic Administration Console with the URL <http://localhost:7001/console>. The login page for the Administration Console gets displayed. In the login page specify user name and password and log in to the administration console.

In the administration console select the Services>JDBC node. To configure a JDBC connection pool, right-click on the Connection Pools node and select Configure a new JDBCConnectionPool. In the Choose database frame displayed select MySQL as the Database Type. Select MySQL's Driver (Type 4) as the Database Driver. Click on the Continue button. Specify the connection properties for the JDBC connection. In the Database Name field specify `test`, the example MySQL database. In the Host Name field specify `localhost`. In the Database User Name field specify `root`. A password is not required to login to MySQL database with the root username, but the WebLogic server requires a password to be specified. Specify the password for the user name. Click on the Continue button.

In the Test database connection frame, the MySQL driver `com.mysql.jdbc.Driver` is specified in the Driver Classname field. The MySQL driver is used to establish a connection with the MySQL database. In the URL field specify `jdbc:mysql://localhost/test` as the connection URL for the database. To test the JDBC connection to the database click on the Test Driver Configuration button. If a connection gets established with the database, a "connection successful" message gets displayed. In the Create and deploy frame, select the server on which the connection pool is to be deployed. Click on the Create and deploy button to deploy the JDBC connection pool on the server. The configured connection gets deployed on the examples server and a node for the connection pool gets added to the JDBC>Connection Pools node. To modify the configuration of the connection pool, select the connection pool node and modify the settings in the different tabs: General, Target and Deploy, Monitoring, Control, Testing, Connections.

Next, configure a data source in the WebLogic server. Right-click on the Services>JDBC node and select Configure a new JDBCTxDataSource. Specify a data source name. In the JNDI Name field specify a JNDI name for the data source – `MySQLDS` for example. Click on the Continue button. In the Connect to connection pool frame, select a connection pool from the list of connection pools. Select the connection pool that was configured in the previous section and click on the Continue button. In the Target the data source frame select a server as the target server for the data source. Click on the Create button. The configured data source gets deployed on the examples server and a node for the data source gets added to the Data Sources node. To modify the data source select the data source node and modify the settings in the different tabs: Configuration, Target, and Deploy. The data source is available with the JNDI name `MySQLDS`, which was specified in the data source configuration.

Converting the JBoss EJB Application

In the previous section the WebLogic server was configured

with the MySQL database. In this section we'll convert the JBoss EJB application to a WebLogic EJB application, which involves converting the deployment descriptors. A JBoss entity EJB application consists of the EJB deployment descriptors (`ejb-jar.xml`, `jboss.xml`, and `jbosscmp-jdbc.xml`), the bean class (`CatalogBean.java`), the remote interface (`Catalog.java`), and the home interface (`CatalogHome.java`). To deploy the entity EJB in the JBoss server, an EJB JAR file is created; this EJB JAR file has the structure:

```
META-INF/
  ejb-jar.xml
  jboss.xml
  jbosscmp-jdbc.xml
CatalogBean.class
Catalog.class
CatalogHome.class
```

The structural and application assembly information for an EJB is specified in the deployment descriptors. Structural information includes specifying the EJB as a session EJB or an entity EJB. The application assembly information in the `ejb-jar.xml` deployment descriptor is specified in the *assembly-descriptor* element. The entity EJB deployment descriptors for JBoss are `ejb-jar.xml`, `jboss.xml`, and `jbosscmp-jdbc.xml`. The corresponding deployment descriptors for WebLogic are `ejb-jar.xml`, `weblogic-ejb-jar.xml`, and `weblogic-cmp-rdbms-jar.xml`. In the following sections the conversion between these files will be described.

The `ejb-jar.xml` deployment descriptor is the same for WebLogic and JBoss. The `ejb-jar.xml` deployment descriptor for the example entity EJB is shown in Listing 4. The example `ejb-jar.xml` defines an entity EJB with the EJB name "Catalog." The example EJB has the CMP fields: `catalogId`, `journal`, and `publisher`. The primary key field is `catalogId`.

Converting jboss.xml to weblogic-ejb-jar.xml

The `weblogic-ejb-jar.xml` and `jboss.xml` deployment descriptors are vendor-specific deployment descriptors for EJBs. To deploy a JBoss EJB application to the WebLogic application server, convert the `jboss.xml` deployment descriptor to `weblogic-ejb-jar.xml`. The root element in `weblogic-ejb-jar.xml` is *weblogic-ejb-jar*. The root element in `jboss.xml` is *jboss*. The JNDI name for an EJB is specified in the `jboss.xml` and `weblogic-ejb-jar.xml` deployment descriptors with the *jndi-name* element or the *local-jndi-name* element. The `jboss.xml` deployment descriptor for the example entity EJB is shown in Listing 5. The *DOCTYPE* element for the `weblogic-ejb-jar.xml` deployment descriptor is:

```
<!DOCTYPE weblogic-ejb-jar PUBLIC
"-//BEA Systems, Inc.//DTD WebLogic 8.1.0 EJB//EN"
"http://www.bea.com/servers/wls810/dtd/weblogic-ejb-jar.dtd" >
```

The *DOCTYPE* for the `jboss.xml` deployment descriptor is:

```
<!DOCTYPE jboss PUBLIC "-//JBoss//DTD JBOSS 4.0//EN"
"http://www.jboss.org/j2ee/dtd/jboss_4_0.dtd">
```

Convert the deployment descriptor `ejb-jar.xml` to the deployment descriptor `weblogic-ejb-jar.xml` with a custom XSLT stylesheet,

weblogic-ejb-jar.xslt, shown in Listing 6. The stylesheet creates weblogic-ejb-jar.xml, the WebLogic equivalent of JBoss's jboss.xml deployment descriptor. The weblogic-ejb-jar.xml file generated with the *weblogic-ejb-jar.xslt* stylesheet is shown in Listing 7.

Converting jbosscomp-jdbc.xml to weblogic-cmp-rdbms-jar.xml

The weblogic-cmp-rdbms-jar.xml deployment specifies the database persistence information for a CMP entity EJB. The weblogic-cmp-rdbms-jar.xml file includes the table name for an entity EJB, the data source to connect to the database, and the columns corresponding to the entity EJB CMP fields. The JBoss deployment descriptor that specifies the CMP entity EJB persistence information is jbosscomp-jdbc.xml. The jbosscomp-jdbc.xml for the example EJB is shown in Listing 8.

The root element of weblogic-cmp-rdbms-jar.xml is *weblogic-rdbms-jar*. The root element in jbosscomp-jdbc.xml is *jbosscomp-jdbc*. The *data-source-name* element, which specifies the data source to connect to the database in the *weblogic-cmp-rdbms-jar.xml* file, is equivalent to the *datasource* element in the *jbosscomp-jdbc.xml* deployment descriptor. The *field-map* element, which specifies the mapping of the entity EJB CMP fields to database table columns in *weblogic-cmp-rdbms-jar.xml*, is the equivalent of the *cmp-field* element in *jbosscomp-jdbc.xml*. The *dbms-column* element, which specifies a column name in *weblogic-cmp-rdbms-jar.xml*, is the equivalent of the *column-name* element in *jbosscomp-jdbc.xml*. The DOCTYPE for the *weblogic-cmp-rdbms-jar.xml* deployment descriptor is:

```
<!DOCTYPE weblogic-rdbms-jar PUBLIC
'-//BEA Systems, Inc.//DTD WebLogic 8.1.0 EJB RDBMS Persistence//EN'
'http://www.bea.com/servers/wls810/dtd/weblogic-rdbms20-persistence-810.dtd'>
```

The DOCTYPE for *jbosscomp-jdbc.xml* is:

```
<!DOCTYPE jbosscomp-jdbc PUBLIC "-//JBoss//DTD JBOSSCMP-JDBC 4.0//EN"
"http://www.jboss.org/j2ee/dtd/jbosscomp-jdbc_4_0.dtd">
```

Convert the deployment descriptor *jbosscomp-jdbc.xml* to *weblogic-cmp-rdbms-jar.xml* with the custom XSLT stylesheet, *weblogic-cmp-rdbms-jar.xslt*, shown in Listing 9. The stylesheet creates *weblogic-cmp-rdbms-jar.xml*, the WebLogic equivalent of JBoss's *jbosscomp-jdbc.xml* deployment descriptor. *weblogic-cmp-rdbms-jar.xml* is shown in Listing 10.

The DTDs for the WebLogic deployment descriptors are different from the JBoss deployment descriptors. With custom XSLTs,

which may be modified if additional elements are present in the deployment descriptors, the JBoss deployment descriptors get converted to WebLogic deployment descriptors. In the following section, the EJB application is deployed in the WebLogic server.

Deploying the EJB Application in WebLogic

After converting the JBoss EJB deployment descriptors to WebLogic deployment descriptors, create an EJB JAR file to deploy in the WebLogic server. The structure of the WebLogic JAR file is:

```
META-INF/
  ejb-jar.xml
  weblogic-ejb-jar.xml
  weblogic-cmp-rdbms-jar.xml
CatalogBean.class
Catalog.class
CatalogHome.class
```

Compile the example EJB classes and interfaces.


```
java Catalog.java CatalogBean.java CatalogHome.java
```

Copy the WebLogic deployment descriptors, *ejb-jar.xml*, *weblogic-ejb-jar.xml*, and *weblogic-cmp-rdbms-jar.xml*, to the META-INF directory. Create a JAR file from the WebLogic deployment descriptors, classes, and interfaces with the JAR (<http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/jar.html>) utility.

```
jar cf CatalogEJB.jar CatalogBean.class
  Catalog.class CatalogHome.class META-INF/*.xml
```

To deploy the WebLogic entity EJB application, copy the JAR file, *CatalogEJB.jar*, to the *<BEA>\user_projects\domains\mydomain\applications* directory, where *<BEA>* is the directory in which WebLogic server is installed. The EJB application gets deployed on the JBoss server when the server is started. The applications directory in the WebLogic application server is the directory that corresponds to the deploy directory in the JBoss application server.

Conclusion

An entity EJB application developed in JBoss may be migrated to the WebLogic application server by converting the deployment descriptors. Using a similar process, a JBoss J2EE Web application may be migrated to WebLogic by converting the *jboss-web.xml* deployment descriptor to *weblogic.xml*. 

Listing 1: CatalogBean.java

<pre>import javax.ejb.*; abstract public class CatalogBean implements EntityBean { private EntityContext ctx;</pre>	<pre>public CatalogBean() {}; public void setEntityContext(EntityContext ctx) { this.ctx = ctx; }</pre>	<pre>public void unsetEntityContext() { this.ctx = null; } abstract public String getCatalogId();</pre>
---	---	---


```

abstract public void setCatalogId(String
catalogId);

abstract public String getJournal();
abstract public void setJournal(String
journal);

abstract public String getPublisher();
abstract public void setPublisher(String
publisher);

public void ejbActivate() {

}

public void ejbPassivate() {

}

public void ejbLoad() {

}

public void ejbStore() {

}

public void ejbRemove()
throws RemoveException
{

}

public String ejbCreate(String catalogId)
throws CreateException
{

setCatalogId(catalogId);

return null;

}

public void ejbPostCreate(String catalogId)
{

}

```

```

}

Listing 2: Catalog.java
import java.rmi.RemoteException;
import javax.ejb.*;

public interface Catalog extends EJBObject {

public String getCatalogId()
throws RemoteException;

public String getJournal()
throws RemoteException;

public String getPublisher()
throws RemoteException;

public void setJournal(String journal)
throws RemoteException;

public void setPublisher(String publisher)
throws RemoteException;

}

```

```

Listing 3: CatalogHome.java
import javax.ejb.CreateException;
import javax.ejb.EJBHome;
import javax.ejb.FinderException;
import java.rmi.RemoteException;

public interface CatalogHome extends EJBHome {

public Catalog create(String catalogId)
throws CreateException, RemoteException;

public Catalog findByPrimaryKey(String
catalogId)
throws FinderException, RemoteException;

}

```

```

Listing 4: ejb-jar.xml
<?xml version="1.0"?>
<!DOCTYPE ejb-jar PUBLIC
"-//Sun Microsystems, Inc.//DTD Enterprise
JavaBeans 2.0//EN"

```

```

"http://java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar>
<enterprise-beans>
<entity>
<ejb-name>Catalog</ejb-name>
<home>com.ejb.CatalogHome</home>
<remote>com.ejb.Catalog</remote>
<ejb-class>com.ejb.CatalogBean</ejb-
class>
<persistence-type>Container</persis-
tence-type>
<prim-key-class>java.lang.String</prim-
key-class>
<reentrant>False</reentrant>
<cmp-version>2.x</cmp-version>
<abstract-schema-name>CatalogBean</ab-
stract-schema-name>
<cmp-field>
<field-name>catalogId</field-name>
</cmp-field>
<cmp-field>
<field-name>journal</field-name>
</cmp-field>
<cmp-field>
<field-name>publisher</field-name>
</cmp-field>
<primkey-field>catalogId</primkey-field>
</entity>
</enterprise-beans>
</ejb-jar>

```

```

Listing 5: jboss.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jboss PUBLIC "-//JBoss//DTD JBOSS
4.0//EN" "http://www.jboss.org/j2ee/dtd/jboss_
4_0.dtd">
<jboss>
<enterprise-beans>
<entity>
<ejb-name>Catalog</ejb-name>
<jndi-name>com.ejb.CatalogHome</jndi-name>
</entity>
</enterprise-beans>
</jboss>

```

```

Listing 6: weblogic-ejb-jar.xslt
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
version="1.0" xmlns:xsl="http://www.
w3.org/1999/XSL/Transform">
<xsl:output method="xml" doctype-public="-//
BEA Systems, Inc.//DTD WebLogic 8.1.0 EJB//EN"
doctype-system="http://www.bea.com/servers/
wls810/dtd/weblogic-ejb-jar.dtd"/>
<xsl:template match="/jboss">
<weblogic-ejb-jar>

```

```

<xsl:apply-templates select="enterprise-beans/
entity"/>
</weblogic-ejb-jar>
</xsl:template>
<xsl:template match="entity">
<weblogic-enterprise-bean>
  <ejb-name><xsl:value-of select="ejb-
name"/></ejb-name>
  <entity-descriptor>
    <persistence>
      <persistence-use>
        <type-identifier>WebLogic_CMP_RDBMS</type-
identifier>
      </persistence-use>
    </persistence>
  </entity-descriptor>
  <jndi-name><xsl:value-of select="jndi-
name"/></jndi-name>
</weblogic-enterprise-bean>
</xsl:template>
</xsl:stylesheet>

```

Listing 7: weblogic-ejb-jar.xml

```

<?xml version="1.0"?>
<!DOCTYPE weblogic-ejb-jar PUBLIC
"-//BEA Systems, Inc.//DTD WebLogic 8.1.0
EJB//EN"
"http://www.bea.com/servers/wls810/dtd/web-
logic-ejb-jar.dtd" >
<weblogic-ejb-jar>
  <weblogic-enterprise-bean>
    <ejb-name>Catalog</ejb-name>
    <entity-descriptor>
      <persistence>
        <persistence-use>
          <type-identifier>WebLogic_CMP_RDBMS</type-
identifier>
        </persistence-use>
      </persistence>
    </entity-descriptor>
    <jndi-name>com.ejb.CatalogHome</jndi-name>
  </weblogic-enterprise-bean>
</weblogic-ejb-jar>

```

Listing 8: jbosscomp-jdbc.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jbosscomp-jdbc PUBLIC "-//JBoss//DTD
JBOSSCMP-JDBC 4.0//EN"
"http://www.jboss.org/j2ee/dtd/jbosscomp-

```

```

jdbc_4_0.dtd">
<jbosscomp-jdbc>
  <defaults>
    <datasource>java:MySqlDS</datasource>
    <datasource-mapping>mySQL</datasource-map-
ping>
  </defaults>
  <enterprise-beans>
    <entity>
      <ejb-name>Catalog</ejb-name>
      <create-table>true</create-table>
      <table-name>Catalog</table-name>
      <cmp-field>
        <field-name>catalogId</field-name>
        <column-name>catalogId</column-name>
      </cmp-field>
      <cmp-field>
        <field-name>journal</field-name>
        <column-name>journal</column-name>
      </cmp-field>
      <cmp-field>
        <field-name>publisher</field-name>
        <column-name>publisher</column-name>
      </cmp-field>
    </entity>
  </enterprise-beans>
</jbosscomp-jdbc>

```

Listing 9: weblogic-cmp-rdbms-jar.xslt

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  version="1.0" xmlns:xsl="http://www.
w3.org/1999/XSL/Transform">
  <xsl:output method="xml" doctype-public="-//
BEA Systems, Inc.//DTD WebLogic 8.1.0 EJB RD-
BMS Persistence//EN" doctype-system="http://
www.bea.com/servers/wls810/dtd/weblogic-rd-
bms20-persistence-810.dtd"/>
  <xsl:template match="/jbosscomp-jdbc">
    <weblogic-rdbms-jar>
      <xsl:apply-templates select="enterprise-beans/
entity"/>
      <create-default-dbms-tables>AlterOrCreate</
create-default-dbms-tables>
    </weblogic-rdbms-jar>
  </xsl:template>
  <xsl:template match="entity">
    <weblogic-rdbms-bean>
      <ejb-name><xsl:value-of select="ejb-
name"/></ejb-name>
      <data-source-name><xsl:value-of se-
lect=".../defaults/datasource"/></data-
source-name>

```

```

      <table-map>
        <table-name><xsl:value-of select="table-
name"/></table-name>
        <xsl:apply-templates select="cmp-field"/>
      </table-map>
    </weblogic-rdbms-bean>
  </xsl:template>
<xsl:template match="cmp-field">
  <field-map>
    <cmp-field><xsl:value-of select="field-
name"/></cmp-field>
    <dbms-column><xsl:value-of
select="column-name"/></dbms-column>
  </field-map>
</xsl:template>
</xsl:stylesheet>

```

Listing 10: weblogic-cmp-rdbms-jar.xml

```

<?xml version="1.0"?>
<!DOCTYPE weblogic-rdbms-jar PUBLIC
'-//BEA Systems, Inc.//DTD WebLogic 8.1.0 EJB
RDBMS Persistence//EN'
'http://www.bea.com/servers/wls810/dtd/web-
logic-rdbms20-persistence-810.dtd'>
<weblogic-rdbms-jar>
  <weblogic-rdbms-bean>
    <ejb-name>Catalog</ejb-name>
    <data-source-name>java:MySqlDS</data-
source-name>
    <table-map>
      <table-name>Catalog</table-name>
      <field-map>
        <cmp-field>catalogId</cmp-field>
        <dbms-column>catalogId</dbms-column>
      </field-map>
      <field-map>
        <cmp-field>journal</cmp-field>
        <dbms-column>journal</dbms-column>
      </field-map>
      <field-map>
        <cmp-field>publisher</cmp-field>
        <dbms-column>publisher</dbms-column>
      </field-map>
    </table-map>
  </weblogic-rdbms-bean>
  <create-default-dbms-tables>AlterOrCreate</
create-default-dbms-tables>
</weblogic-rdbms-jar>

```


Looking to Stay Ahead of the *i*-Technology Curve?

Subscribe to these **FREE** Newsletters >

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

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

And best of all – they're **FREE!**

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

IT solutions
JOURNAL

NET JOURNAL

JDJ
JOURNAL

WebServices
JOURNAL

Information STORAGE+ SECURITY
journal

LINUXWORLD
MAGAZINE

wireless
BUSINESS TECHNOLOGY

LINUX BUSINESS WEEK

XML JOURNAL

MX
developer's journal

WebSphere
JOURNAL

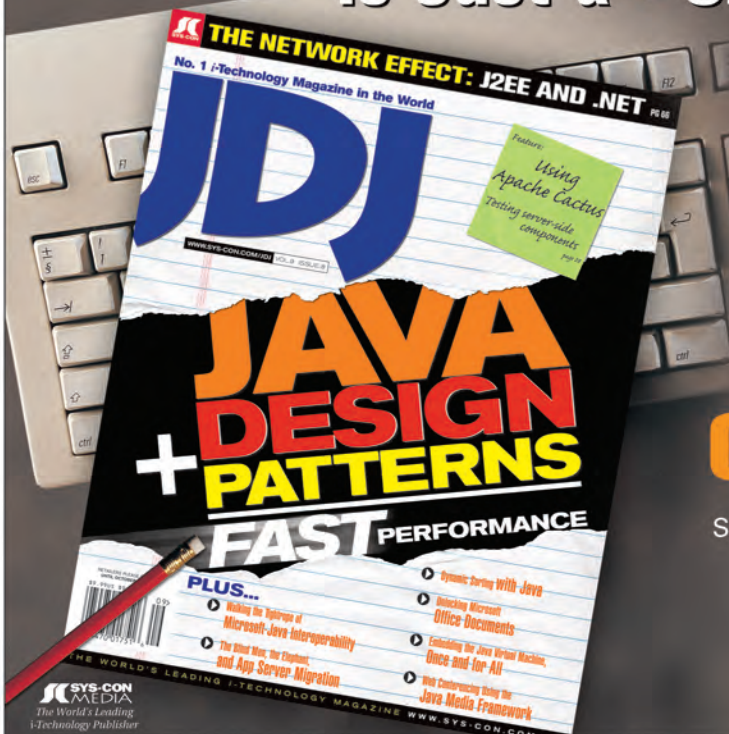
wldj
THE LEADING INDEPENDENT SOURCE FOR BUSINESS PERFORMANCE

COLDFUSION Developer's Journal

SYS-CON MEDIA

The World's Leading *i*-Technology Publisher

The World's Leading Java Resource Is Just a >Click< Away!



JDJ is the world's premier independent, vendor-neutral print resource for the ever-expanding international community of Internet technology professionals who use Java.

Only \$**69**⁹⁹ ONE YEAR 12 ISSUES

Subscription Price Includes **FREE** JDJ Digital Edition!

www.SYS-CON.com/JDJ
or 1-888-303-5282

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

Developing a Service Information Portal on the WebLogic Platform

MOVING TO JAVA PORTLETS ON WEBLOGIC PORTAL



By Eric Peterson

Portals can serve many purposes, including the delivery of information used for IT management purposes. A team of engineers within Hewlett-Packard has developed a portal-based J2EE-application, Service Information Portal (SIP), which brings together various management products from HP OpenView, HP's IT management software, and provides a single portal view to many groups of users.

Originally developed before the Java Portlet standard (JSR-168) was released, this product delivers its own portal server and framework. While the team successfully integrated data from the various management systems, the portal lacked the ability to have more flexible layout and navigation. The portal was also limited when it attempted to integrate data from outside sources.

With the Java community's adoption of JSR 168 the SIP team saw the clear benefits of exiting the portal server market. The SIP product provides the content customers want in a single access point, rather than the underlying proprietary portal framework. Many customers of SIP have also built their own enterprise information portals. A big part of portals' value is in the ability to consolidate many pieces of data from various places into one place, so having both an EIP and SIP was a problem for customers.

This growth in the portal server market provided the SIP team with many options for a reference portal server. The team had additional requirements that were atypical of many portal projects. They must deliver functionality in a new portal server that is equivalent to what was available in previous versions. Such functionality that was previously available includes authorization, portal layout and view design, and configuration and application of data filtering. The product had survived for several years and was on version 3.2; developing a migration path was critical as well.

This article describes SIP as it exists in version 3.2, and the changes needed to move to a JSR-168 application using BEA WebLogic Portal Server. It also discusses the benefits of moving to WebLogic Portal Server, compared to using SIP 3.2.

Overview of SIP 3 Portal Architecture

SIP's modules are conceptually very similar to Java Portlets. The modules' implementations are based on Servlet technology, but the configurations of SIP modules and Java Portlets are very different. Within SIP 3, a single module (servlet instance) can be used with many different configurations, and no concept of user-specific settings (portlet preferences) is available. SIP contains a large number of module configurations, instead of a great number of user-specific preferences, as Java Portlets provide. For some installations, a few thousand module configurations would not be unusual. Listing 1 shows a section of what a SIP 3 view file might look like, just to display two columns with a module in each.

SIP 3 contains portal view files for portal layout and overall view and navigation configuration.

Author Bio:

Eric Peterson works within HP's OpenView organization as a software engineer on the Service Information Portal product. He has worked on J2EE Web applications and Web site development for the past eight years, for companies that range from small local ISPs to international enterprises.

ericpeterson@hp.com

These portal views are similar to BEA portal template files; they contain the sheets within the view (BEA pages) and module (portlet) placement. However, all of the module configuration information is stored within the SIP view files. Within Java Portlets, portlet configuration is kept in the portlet.xml file. The view file might have five sheets (think BEA pages) and 20 modules (think portlets). The lack of significant overhead encountered when many module definitions are present is a key difference between module configurations stored in view files versus portlet definitions stored within a portlet.xml file. The modules' implementation is a single servlet instance receiving multiple requests with varying configuration information. In a portal server, each unique configuration for a portlet might mean a new portlet instance.

Data filtering is a key piece of SIP functionality. Sometimes a specific audience should only have access to certain management data. An IT service provider may want to provide unique views of management data to different audiences - HR may need to have one view of IT services, and accounting should have their own individualized view. This concept is even more critical when applied to an outsourcing situation, where isolating a customer's access to their own pertinent data can have serious business and legal implications.

Basic SIP authentication requires user validation. The traditional concepts of users and roles are defined within SIP 3, even if this already exists within the authentication system. SIP requires knowledge of roles and users to perform authorization for various portal activities, including who can view or edit which items. Additionally, SIP supports alternative authentication mechanisms such as LDAP, the underlying Web server authentication, or an API to create custom authentication.

The SIP 3 portal framework contains several facets of the term "role" that are critically important. The SIP role delineates which data filter to use and which view a user will see. This, in turn, defines all of the module configurations (see Figure 2). In addition, an SIP user only actively uses one role at a time, whereas, in most Web application servers, a user can be in more than one role simultaneously. If an SIP 3 user has more than one role available, he or she is presented with a drop down box, which allows the user to switch from one portal role to another (see Figure 1).

Portal Server Analysis

The team had to evaluate and choose a portal server that would suit their emerging needs. Two key themes played into portal server selection: the team needed to deliver the same functionality as had been previously available, and they wanted to develop within a robust portal platform that would allow many other integration possibilities. There are some strong open source portal platforms, but these platforms are typically geared toward a more technical community; they may not be JSR 168 compliant, and they may not have sufficient easy-to-use, robust integration possibilities available.

The team chose BEA WebLogic portal server 8.1 for their next

TABLE 1

	SIP 3.2 & Proprietary Portal Framework	BEA WebLogic Portal & SIP 4 as a portlet application
How Content Is Delivered	SIP modules, around 10-15 available to use. Modules are mainly focused on OpenView products.	Portlets, BEA has a large portal solutions catalog as well as support for any general Java Portlet. Has many options for all types of integrations.
Authentication and Authorization	An open framework to support other authentication systems, authorization is within SIP and requires knowledge of all users and roles. There are limited authentication options out of the box. Only very broad authentication control is available.	An open framework to support other Authentication systems, authorization is done within BEA entitlements and requires knowledge of roles. Many authentication options are available. Entitlements allow broad or granular control over authentication.
Presentation Options	Flat: A collection of sheets (similar to BEA pages), with each sheet having two columns of content.	Hierarchical: Books, pages, with nested books and more pages. Multiple menu options are available for navigation and pages support 1-4 columns of portlets.

A comparison between SIP 3.2 and BEA WebLogic Portal Server

release. The portal solutions catalog (<http://dev.bea.com/products/wportal/psc/index.jsp>) provided many integration possibilities, and the portal administration application provided a user-friendly administration tool with much of the same functionality available within the previous versions of SIP.

Moving to Java Portlets and BEA's Portal Server

With a promising portal server, the team was ready to exit the portal server market and enter the portlet content market. Issues the team needed to resolve included:

- Moving modules to portlets
- Translating portal views
- How data filtering would be supported
- Coupling more loosely with authentication and authorization requirements

Moving Modules to Portlets

The first part of the project involved migrating a SIP module to a Java Portlet. Luckily, the Portlet API and Servlet API are very similar in their use of request and response objects. This simplified the move, since all of the modules were Servlets. Much of the code could remain, with minimal refactoring.

FIGURE 1



SIP 3 screen shot. Note the drop down in the upper left that allows users to toggle between roles (Admin: Live Demo).

One of the major differences the team encountered was the use of the PortletURLs and render parameters. By using servlets, SIP 3 could get the entire HTTP request and response, and send any parameters back and forth through Web pages. SIP 3 uses a common coding practice, which involves passing a module ID parameter to help identify a request for a certain module configuration. When a module ID is encountered during a drill down, the SIP portal server displays only the requested module in a maximized window. The SIP portal server had no support for drilling down into two modules at the same time within the same browser. The following JSP snippet is similar to what is found in SIP 3:

```
<a href="portal?service=<%=serviceID%>&moduleID=<%=moduleID%>">
  navigation link</a>
```

In the world of portlets for SIP 4, PortletURLs help pass the correct parameters to the correct portlet. A portal server must be able to determine which portlet should receive which parameters and ensure that the portlets retrieve the correct parameters. Using the aforementioned technique would not work; parameter pass-

ing needed to be redone. This problem is especially evident when testing in other portal servers, which are free to encode portlet parameters in any way they wish. Changes were made in the SIP portlet code, the display, or both. The following technique corrects this problem of parameter passing:

```
PortletURL drillDownURL = response.createRenderURL();
drillDownURL.setParameter("service", serviceID);
```

In addition, the following is updated in the JSP:

```
<a href="<%= drillDownURL.toString()%>">navigation link</a>
```

The SIP product predated some of the Web development tools and technologies used today, and the work described above to correct URL parameter passing was a common task during migration. However, by transferring URL handling into the Portlet API, some additional options were made available. Previously, a drill down in modules would only display the content of a single module. Using the Portlet APIs, the team can drill down and navigate in multiple portlets simultaneously. This can be especially handy when SIP is used to diagnose IT issues, which commonly requires access to a few different spots to obtain the full picture.

Translating Portal Views

The portal view functionality is an easy migration to BEA's terms and technologies. In fact, BEA's concept of books, pages, and nested books provides a significant improvement to SIP users. SIP 3 only supports one level of sheets, which is equivalent to a single BEA portal book with as many pages as needed. There are no possibilities of hierarchical navigation and portal layout within SIP 3. Figure 1 shows the "sheets" provided in SIP 3, and Figure 3 shows SIP content with a nested book and the multiple-level navigation. SIP 3 stored its view files in XML as shown in Listing 1, so an XSL transform can migrate a SIP 3 view file to a BEA portal template file.

Supporting Data Filtering

The data filtering tools within SIP did not change in their capability, but their application was modified. Data filtering can be a key piece of functionality for users, and it's very dependant upon the currently selected role of the SIP 3 user. For SIP 4, a portlet preference is used. When a user accesses a portlet that employs data filtering, SIP 4 obtains the user's filter preference. The filters within SIP 4 contain a new security specification, which details whether a filter is to be available to a list of roles, to all authenticated users, or to all users (including anonymous access). See Figure 4 for the updated relationship between organization filters and SIP portlets.

Moving data filtering into portlet preferences allows far fewer portlets to serve a larger audience of users. In fact, this is true with all portlet preferences. SIP 3 uses module configurations within a view file as shown in Listing 1. If one user prefers a different module view, SIP 3 requires a different role, portal view, and set of module configurations. In contrast, portlet preferences allow any number of users to customize their own personal preferences from a single portlet configuration. Therefore, hundreds or thousands of module configurations used in SIP 3 can now equal a handful of portlet configurations.

FIGURE 2

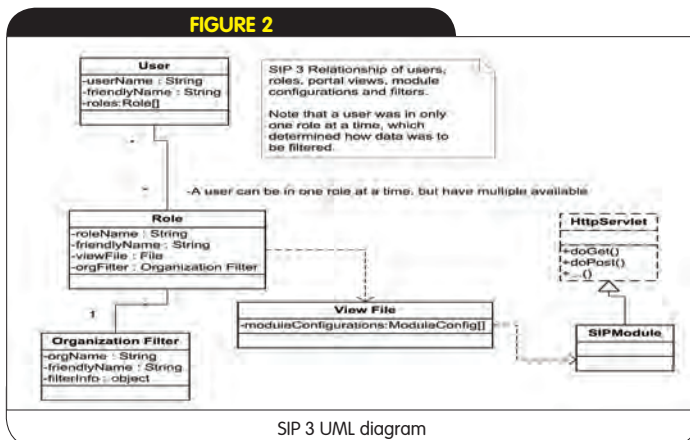
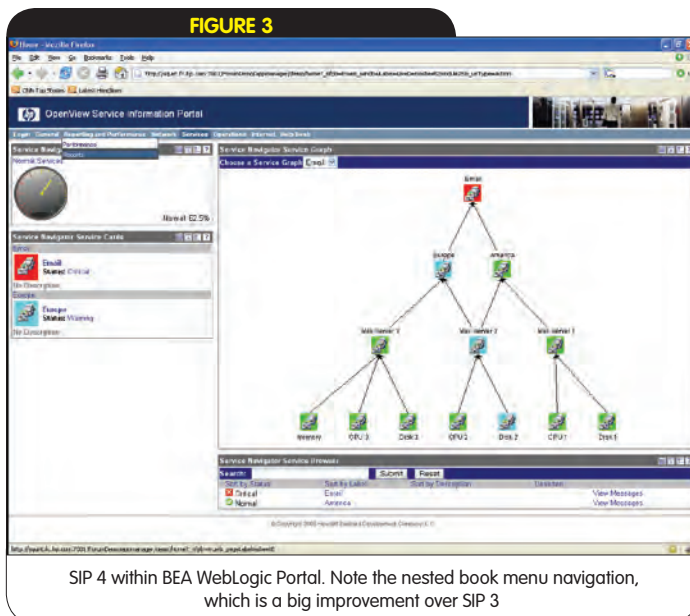


FIGURE 3



REPRINT!

Once
you're in
it...



...reprint it!

- ColdFusion Developer's Journal
- Java Developer's Journal
- Web Services Developer's Journal
- Wireless Business & Technology
- XML Journal
- PowerBuilder Developer's Journal
- .NET Developer's Journal

Contact Dorothy Gil
201 802-3024
dorothy@sys-con.com

REprints



WEEKEND WITH EXPERTS



Get a Java injection on the go!

Spend a weekend and have a lunch with experts in an intimate learning environment in the city near you.

No salesmen allowed. Join our technical discussions and hands-on workshops.

The next Roadshow is in New York City on October 15 and 16.

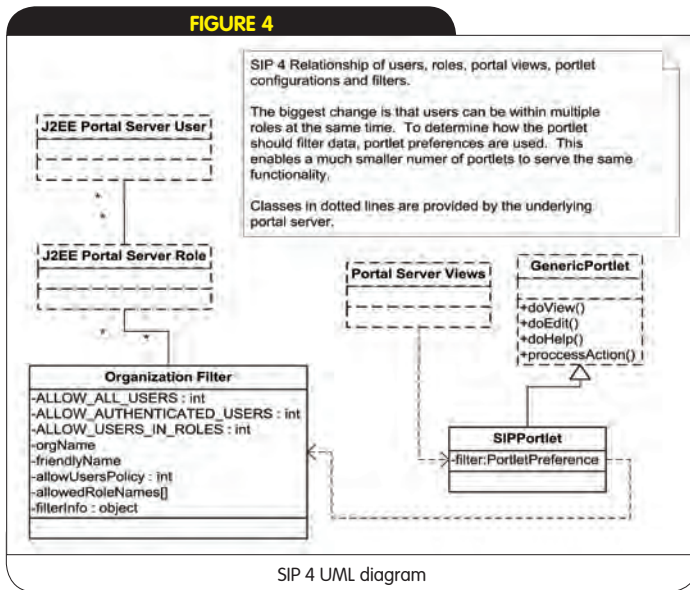
www.weekendwithexperts.com

WLDJ ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Blog-n-Play.com	www.blog-n-play.com	201-802-3000	22
EV1 Servers	www.ev1servers.net	800-504-SURF	9
HP	http://devresource.hp.com/d2d.htm	800-752-0900	13
Information Storage & Security	www.issjournal.com	888-303-5282	41
Intersperse	www.intersperse.com	800-340-4553	2, 19
IT Solutions Guide	www.sys-con.com	201-802-3021	39
JDJ	www.sys-con.com/jdj	888-303-5282	33
LinuxWorld Conference & Expo	www.linuxworldexpo.com		17
LinuxWorld Magazine	www.linuxworld.com	888-303-5282	39
Motive	www.motive.com/within1	512-531-2527	52
NetIQ	www.netiq.com/solutions/web	408-856-3000	51
Parasoft	www.parasoft.com/soaptest_wdj	626-256-3680	3
Quest Software	www.quest.com/wldj	949-754-8633	7
SandCherry	www.sandcherry.com	866-383-4500	15
Smart Data Processing Inc	www.weekendwithexperts.com		37
SYS-CON e-newsletters	www.sys-con.com	888-303-5282	33
SYS-CON Publications	www.sys-con.com/2001/sub.cfm	888-303-5282	45
SYS-CON Reprints	www.sys-con.com	201-802-3026	37
Tangosol	www.tangosol.com	617-623-5782	11
EclipseWorld	www.eclipseworld.net		27
Wily Technology	www.wilytech.com	888-GET-WILY	5
WLDJ	www.sys-con.com/weblogic/	888-303-5282	43

This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

FIGURE 4

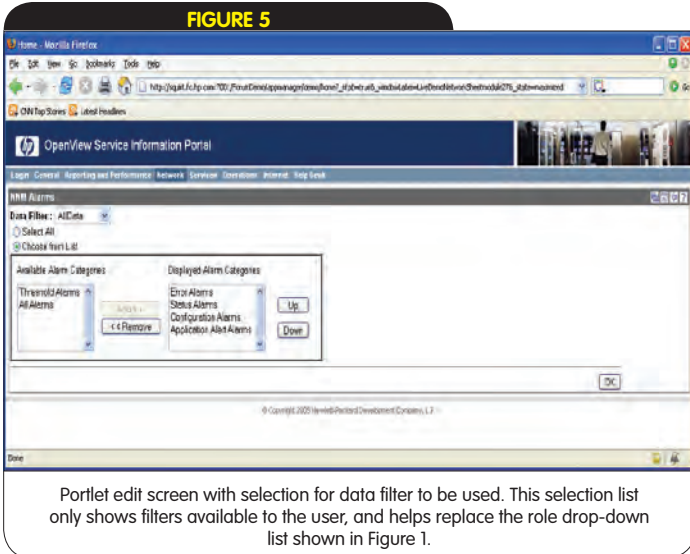


```

" SIPAdministrator "{Grp(SIPAdministrator)}" ""
# add the user "sipadmin" to the group "SIPAdministrator"
invoke -mbean Security:Name=myrealmDefaultAuthenticator -method
addMemberToGroup SIPAdministrator sipadmin
    
```

In summary, the concepts described above highlight the translation of SIP 3 into SIP 4, a Java Portlet application on BEA's portal server. The task of creating users, groups, and roles moved easily into BEA's default security implementation, but still provides many other authentication and authorization options. Designing portal layout and navigation is much more flexible and powerful with BEA portal books and pages. Data filtering, tied to the active role in SIP 3, is now set according to portlet preferences in SIP 4. Finally, the hundreds to thousands of module configurations have been reduced to a handful of portlet configurations with the addition of user-specific portlet preferences.

FIGURE 5



Listing 1: Snippet from a SIP 3 sample view configuration file, with a module configuration contained in two columns. It is common to find 20-30 module configurations contained in a single view file, with most installations containing many view files. This translated to hundreds, if not thousands, of module configurations.

```

...
<Sheet id="sheet1" title="General">
  <Column width="narrow">
    <ModuleInstance classid="com.hp.ov.portal.modules.bookmarks" display="yes" help="/OvSipDocs/C/help/SIP/bookmkView.html" id="module7" rollupState="down" title="Bookmarks">
      <Bookmarks>
        <SharedGroup name="hpsipintegrations"/>
        <Entry href="http://www.hp.com" target="bookmarkwin" title="HP Primary Site"/>
        <Entry href="http://www.openview.hp.com" target="bookmarkwin" title="HP OpenView Site"/>
        <SharedGroup name="hplist"/>
        <SharedGroup name="hpsipintegrations"/>
      </Bookmarks>
    </ModuleInstance>
  </Column>
  <Column width="wide">
    <ModuleInstance classid="com.hp.ov.portal.modules.message-board" display="yes" help="/OvSipDocs/C/help/SIP/messageView.html" id="module8" rollupState="down" title="Message Board">
      <MessageBoard>
        <Message filename="default"/>
        <Message filename="Welcome"/>
      </MessageBoard>
    </ModuleInstance>
  </Column>
</Sheet>
...
    
```

Authentication and Authorization

Authentication and authorization concepts were also an easy item to migrate in BEA. All of the BEA tools and integration options to manage users, roles, and groups can now be used for SIP. For existing SIP users, a WebLogic *weblogic.Admin* batch file can be created to convert all of the SIP users and roles into BEA users, roles, and groups. The following is a *weblogic.Admin* batch file snippet created during migration.

```

# create the user "sipadmin"
invoke -mbean Security:Name=myrealmDefaultAuthenticator -method
createUser sipadmin sipadmin "SIP Admin"
# create the group "SIPAdministrator"
invoke -mbean Security:Name=myrealmDefaultAuthenticator -method
createGroup SIPAdministrator "SIP Administrator"
# create the role "SIPAdministrator", any users in the group
"SIPAdministrator"
invoke -mbean Security:Name=myrealmDefaultRoleMapper -method createRole
    
```


Conclusion

The SIP team moved from proprietary modules to Java Portlets, and found a roughly equivalent, if not much improved, alternative in the BEA WebLogic portal server. While moving to Java Portlets provided some challenges, there were just as many unanticipated benefits from switching to WebLogic portal. WebLogic portal can be a very compelling platform for a JSR 168 application, using vendor-neutral technologies for implementation and rich tools for portal development and deployment. One of the better moves we made during our project was to work with other portal servers such as Liferay. Working with other portal servers really highlights what JSR 168 provides and doesn't provide, and how portal servers can implement compliance in their own way. If you believe your portlets will be used in a portal server other than BEA, working with other portal servers early is key. Our team also found the use of Eclipse as the IDE and remote debugging of BEA was a big time saver. Using Eclipse was also a benefit in that we can develop in a single environment and test portlets in more than one portal server easily.

References

- www.managementsoftware.hp.com/products/sip/
- www.openview.hp.com/
- www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/weblogic/portal/

www.linuxworld.com

Connect

Subscribe Today!

Connect online for fastest service... don't miss another issue of LWM!

SAVE 30% OFF!
REGULAR ANNUAL COVER PRICE \$71.76

YOU PAY ONLY \$49⁹⁹
12 ISSUES/YR

LOG ON TO

SYS-CON MEDIA
The World's Leading i-Technology Publisher

Connect

www.linuxworld.com

Reach Over 100,000 Enterprise Development Managers & Decision Makers with...



Offering leading software, services, and hardware vendors an opportunity to speak to over 100,000 purchasing decision makers about their products, the enterprise IT marketplace, and emerging trends critical to developers, programmers, and IT management

Don't Miss Your Opportunity to Be a Part of the Next Issue!

Get Listed as a Top 20* Solutions Provider

For Advertising Details
Call 201 802-3021 Today!

*ONLY 20 ADVERTISERS WILL BE DISPLAYED. FIRST COME FIRST SERVE.



The World's Leading i-Technology Publisher

Defragment Your View of the World for a Quiet Life

AN INNOVATIVE APPROACH TO THE DISTRIBUTED DATA PROBLEM



By Peter Holditch

The value of two phase commit transactions has always been that programmers can write applications that access data spread across multiple databases and be confident that any updates that are made will be consistently reflected in all of the databases, or none, at the end of the transaction.

Clearly, the reality underlying this observation is that data is split across multiple stores. And how! As project-based applications are built, or bought, data about a single business entity (let's take "customer" as an example) become diffused over many back-end systems, and before long the poor guy is on the phone, complaining that you got his personal details wrong, despite his having informed you of his change of address five times in the last six weeks. But that's just the customer perspective. All sorts of other people feel this pain; managers can't get consolidated views of the things they care about: customers, products, sales, you name it they are doubtless populating their own personal spreadsheets with data gleaned from five or six different back ends in order to inform their usual run of day-to-day decisions.

So, the IT department set off in response on a crusade to build a portal to centralize access to the data, which is good as far as it goes – at least now the code that is the equivalent of that spreadsheet is managed somewhere on a server (although the manager may debate this as a benefit, since now he can't change anything for three months, while he waits for a developer to become free). All this turns out to be a mirage in the longer term as well – as the data landscape changes at the back end, unpredictable quantities of this front-end data integration logic gets broken, and needs rework; not to mention the fact that almost none of it is reusable – a million different little queries for a

million different specific purposes, and growing fast.

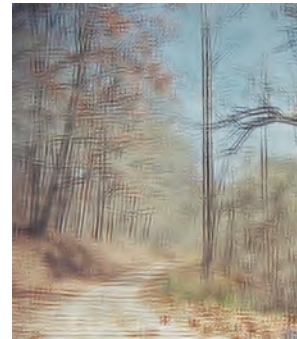
This landscape is about as far from the "service-oriented architecture" vision as it is possible to be – all of the pain of a stovepiped approach to functionality (and therefore data) in one painful place. It is because of the importance of logically organized, coherent data, and the grim reality of its fragmented storage that a data services layer is one of the first architectural layers that gets looked into as IT departments start to review their desired reference architecture for a move to SOA.

In the past, this problem has been addressed by building data-mart type intermediate databases, drip-fed via ETL of data from the fragmented data sources of record. This does provide for a consistent data model to query, but at the cost of building and maintaining the ETL and with the requirement that the data cannot HAVE to be absolutely up-to-date. After all, you are querying a consistent view of the data, but it's a snapshot – not the real thing. Just imagine if there were an infrastructure that could do for the logical presentation of data what the transaction manager does for its transactional consistency...

Well, there is! And as with all of the best and most creative artists, it has recently changed its name. The product formerly known as Liquid Data – now called AquaLogic Data Services – is an infrastructure that provides for a systematic solution to these issues.

Enter ALDS

At the root, you model your data as you'd like it to appear (the logical data model, canonical data representation, call it what you will) your data as it physically exists (be that data held in relational schemas, applications, Web services, or whatever) and then you declaratively define



Author Bio:

Peter Holditch joined BEA as a consultant in the Northern European Professional Services organization in September 1996. He now works as a Pre-Sales Architect in the UK. Peter has a degree in electronic and computer engineering from the University of Birmingham. When he's not pre-selling architecture, he likes to build furniture, brew beer and enjoy the long hot British summers.

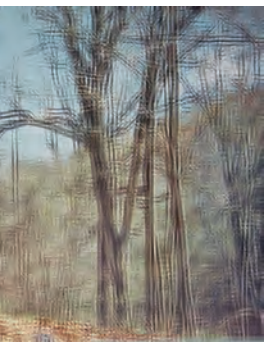
Contact:

peter.holditch@bea.com

how the physical data elements relate to the logical data model using the W3C standard XQuery language. In this way, you have solved the fundamental problem of understanding where the data you want to look at comes from, and you have done it in a systematic, managed way. So, looks as good as the portal vs. spreadsheet phase of evolution then?

Actually, it looks a lot better. Recall that all we did so far is declare how the data is built up via relationships. The actual execution of the queries is managed at runtime by the ALDS query engine, which is guided by your declared relationships. What you run queries against is the logical data model, and what that physically causes to happen is driven by what data you are looking for – if you look for a single element of data that comes ultimately from a single back-end database or application, the query that actually runs will just retrieve the necessary information, not all that it would take to populate the entire logical data model. We have achieved, therefore, not only centralized management of data relationships, but reusable relationships – instead of the million aforementioned combinations of data retrieval for a million different specific purposes we have one declared relationship with as many different filters applied to it at runtime as are needed to address all of the different use cases. Wow, reuse! Sounds like a bit of a SOA to me! Maybe it's not an accident that the name changed and the product became part of BEA's new AquaLogic Service Infrastructure product family!

And because the queries are being pushed down to the data sources of record at runtime, the data we see is absolutely up-to-date, removing another of the problems with the “just add another database” technology elastoplast traditional solution to this problem. Finally in addition to all this, ALDS allows you to model the relationships between your physical data sources, so as the back ends change, you can do quick impact analyses of what will be impacted (and, of course, the impact will be restricted to the ALDS tier – the front ends will continue to work against the same logical data model).



But that's not all: the product allows updates as well as queries – updates are decomposed and pushed to the back-end data sources by the query engine, and there is an update framework where you can manage the consistency of updates across multiple stores as appropriate

And what's the role of the transaction manager in all of this? Well, if all your data sources are xa compliant, ALDS can use it to give you update capability with no need to write any code. Magic, I'll have one!!

In conclusion, I would encourage anybody who is grappling with these types of issues to give ALDS a test run. Because it's an innovative approach to the distributed data problem, a lot of people's initial reaction to it is “oh, it won't go fast enough” or some such. In practice, I have never failed to see these same individuals hearts melted as they see the power (and efficiency) of the product and its approach to data integration. Go on... Give it a spin! You know it makes sense! Read more about the ALDS product at <http://dev2dev.bea.com/dataservices/>. ●

Subscribe Today!

— INCLUDES —
FREE DIGITAL EDITION!
(WITH PAID SUBSCRIPTION)
GET YOUR ACCESS CODE INSTANTLY!



The major infosecurity issues of the day... identity theft, cyber-terrorism, encryption, perimeter defense, and more come to the forefront in ISSJ the storage and security magazine targeted at IT professionals, managers, and decision makers

SAVE 50% OFF!

(REGULAR NEWSSTAND PRICE)

Only \$39⁹⁹

ONE YEAR
12 ISSUES

www.ISSJournal.com
or 1-888-303-5282

 **SYSCON
MEDIA**

The World's Leading i-Technology Publisher

- continued from page 6

Enterprise Service Bus (ESB)

Enterprise Integration also lacked standards a few years ago. Most of the integration solutions were ad hoc and used

different vendor products. Interoperability was always an issue as different vendor products supported different protocols and tools. Integration was always time-

consuming and very expensive. The promise of ESB is to solve these problems through a standards-based service-oriented architecture (SOA) approach.

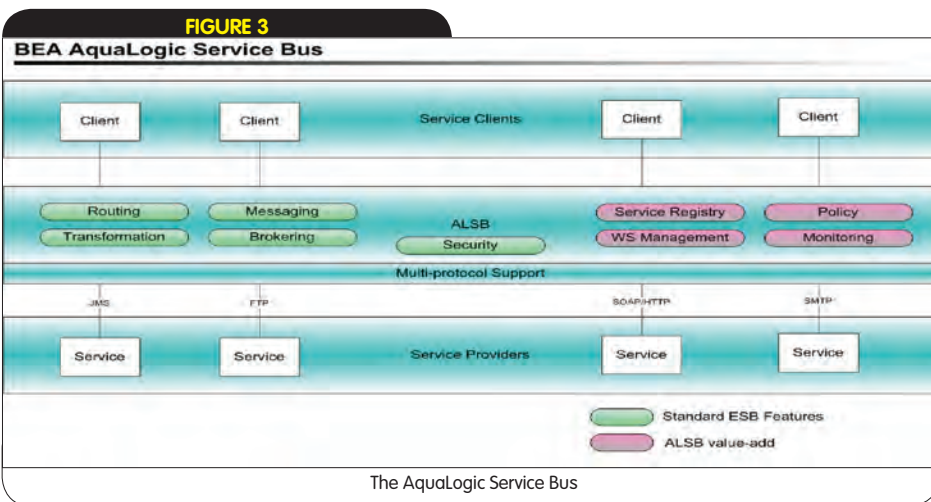
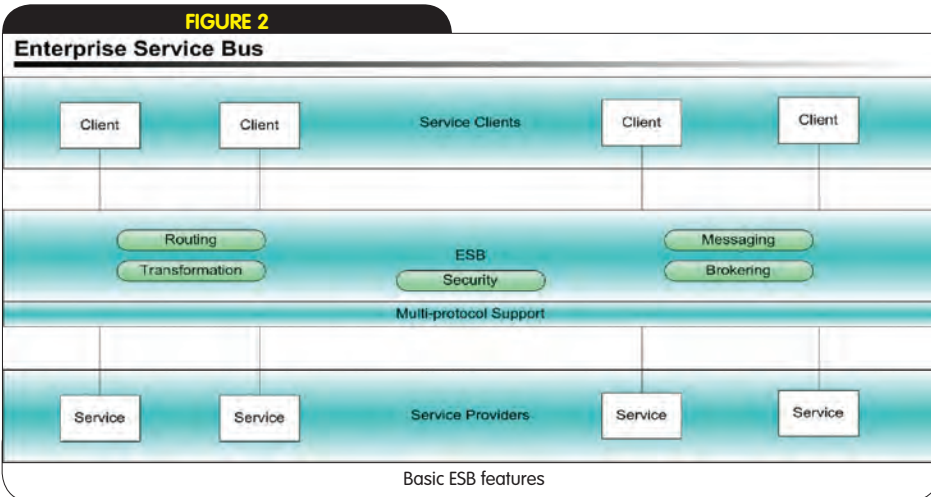
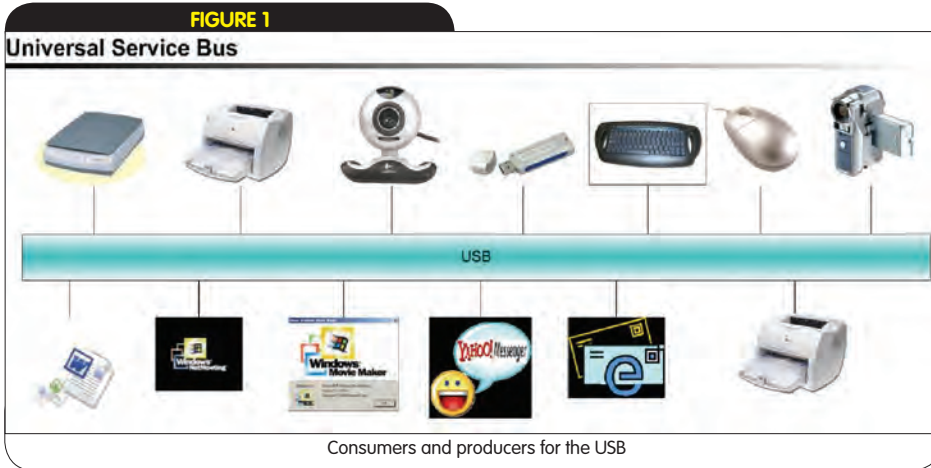
Like the USB, how nice would it be to buy vendor products as appliances and plug them into your infrastructure and use them without much hassle! Developments in XML and Web services standards have paved way for this. Acceptance of XML as a native data format and vendor adoption of Web services has resulted in a common language of communication between disparate enterprise systems. This has solved most of the application interoperability issues. Organizations such as WS-I and SOAP Builders have been working to make sure that services are interoperable between vendor implementations and hardware. USB standards revolutionized the PC connectivity world and these XML and Web services standards are revolutionizing software development and the integration world.

Looking at the history of software architecture, monolithic applications evolved into client-server, 3-tier, N-tier, and into SOA. The trend has been changing from developing everything in-house to procuring out-of-the-box vendor products when possible. This trend clearly indicates a move towards reducing development effort. Application servers took care of infrastructure services such as security, transactions, scalability, etc., and let the developers concentrate on the business logic. Off-the-shelf products were built to run on the application server. In spite of all of these developments, the development effort was still significantly higher and reusability across the enterprise was very low.

IT is moving towards more of a configuration-based approach than ad hoc development. Instead of buying "frameworks" and developing their own applications, companies prefer to buy off-the-shelf products and configure them without much effort. ESB is a big step in this right direction. Now, let's take a look at some of the characteristics of ESB.

Loose Coupling

ESB provides a platform to connect, discover, and access services regardless of how the services themselves are implemented and what hardware and software platform they are running on. This means that the ESB would support multiple protocols and access methods.



THE WORLD'S LEADING INDEPENDENT WEBLOGIC DEVELOPER RESOURCE

Helping you enable intercompany collaboration on a global scale

- Product Reviews
- Case Studies
- Tips, Tricks and More!



***Only \$49.99 for 2 years (12 issues)**

Now in More Than 5,000 Bookstores Worldwide

Go Online & Subscribe **Today!** www.WLDJ.com

SYS-CON.COM/WEBLOGIC/

SYS-CON Media, the world's leading publisher of *i*-technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of WebLogic.



ESB also decouples the consumers and providers of service. By providing this location transparency, it allows the providers to change without affecting the consumer, which is a very powerful feature for today's agile enterprises.

Like the USB hub, the ESB is extensible and scalable. It can span across multiple pieces of hardware, but can still function as a single distributed unit. You should be able to connect ESBs together and share services across the enterprise locally and globally.

Routing

Routing is an important feature of ESB. ESB should support configurable, dynamic routing. Routing allows business processes to be implemented and changed through configuration. This is similar to configuring scanner software for a printer, e-mail application, etc. Once done, a document can be scanned and printed or e-mailed using a touch of a button. The scanner software routes the document to printer or e-mail based on the configuration.

The advent of XML-enabled Content Based Routing (CBR) provided the ability to identify specific fields in the message, which allowed routing of the message to services based on those fields. ESB could examine the message content and route it to the appropriate service.

Transformation

Another concept to understand is transformation. The data in the devices can be represented in multiple formats. For example, when you scan a document, you can use OCR to read it into a MS Word document that represents it in text format, or stores it as a JPG image, or prints it on paper in a hard form. This is analogous to the transformation that happens in the ESB.

Since ESB connects multiple applications and services, it is imperative to support the different formats of data that they require. Thus, an ESB should provide support for transformation of data. A technique to avoid decomposition of data is using a canonical data model across the services. This also reduces the number of transformation types to be supported, as any data type can be transformed to Canonical and to another data type rather than having transformations for every data type. XQuery and XSLT are typically used to

perform XML-to-XML transformation.

Business Process

ESB enables moving the business process logic out of applications and into the bus. This way, applications or services can be orchestrated at a higher level compared to hard-coded process logic. This improves the agility of the applications.

AquaLogic Service Bus (ALSB)

Previously code named QuickSilver, AquaLogic Service Bus is an integral part of BEA's AquaLogic product family, which also includes AquaLogic Data Services, AquaLogic Enterprise Security, and AquaLogic service registry. Built on the industry-leading, proven WebLogic Platform that focuses on developers, AquaLogic focuses on application development and integration through configuration and composition.

AquaLogic is much more than an ESB. In addition to providing a robust implementation of the ESB, it also offers service management capabilities. Figure 3 shows the ALSB with standard ESB capabilities and BEA value-add.

ALSB supports UDDI v3 for service registration and discovery. It provides dynamic routing and transformation capabilities across heterogeneous platforms. As depicted in the diagram, it supports multiple protocols.

ALSB also has security and policy enforcement, message tracking and monitoring, SLA alerts, and service-versioning capabilities. This unique combination of service management capabilities with the ESB makes ALSB a very powerful product. ALSB complements other BEA products such as WebLogic Integration and Liquid Data.

Summary

In this article we looked at the analogy of USB to understand the ESB. The success of USB indicates that ESB is also poised to succeed, if executed well. The idea behind both of these technologies is the same: how to provide a standards-based platform to connect the consumers and providers to share and reuse, thus providing value to the customers. The BEA AquaLogic Service Bus combines the ESB capabilities with management and monitoring capabilities. ●

THREE REASONS TO

blog-n-play.com

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

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

2 Own a most prestigious blog address!

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

3 Best blog engine in the world...

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

www.TAMI.linuxworld.com

"Many blogs to choose from"

PICK YOUR MOST PRESTIGIOUS ADDRESS

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

3 MINUTE SETUP

Sign up for your FREE blog Today!

blog-n-play.com™
i-Technology Blogs Read by Millions *beta*

www.blog-n-play.com

— This site will go beta February 15, 2005!

A LIMITED TIME SAVINGS OFFER FROM SYS-CON MEDIA

SUBSCRIBE TODAY TO MULTIPLE MAGAZINES

AND SAVE UP TO \$340 AND RECEIVE UP TO 3 FREE CDs!*

RECEIVE YOUR DIGITAL EDITION ACCESS CODE INSTANTLY WITH YOUR PAID SUBSCRIPTIONS



3-Pack
Pick any 3 of our magazines and save up to **\$210⁰⁰**
Pay only \$99 for a 1 year subscription plus a **FREE CD**

- 2 Year – \$179.00
- Canada/Mexico – \$189.00
- International – \$199.00

CALL TODAY! 888-303-5282

Pick a 3-Pack, a 6-Pack or a 9-Pack

<input type="checkbox"/> 3-Pack	<input type="checkbox"/> 1YR	<input type="checkbox"/> 2YR	<input type="checkbox"/> U.S.	<input type="checkbox"/> Can/Mex	<input type="checkbox"/> Int'l.
<input type="checkbox"/> 6-Pack	<input type="checkbox"/> 1YR	<input type="checkbox"/> 2YR	<input type="checkbox"/> U.S.	<input type="checkbox"/> Can/Mex	<input type="checkbox"/> Int'l.
<input type="checkbox"/> 9-Pack	<input type="checkbox"/> 1YR	<input type="checkbox"/> 2YR	<input type="checkbox"/> U.S.	<input type="checkbox"/> Can/Mex	<input type="checkbox"/> Int'l.

TO ORDER • Choose the Multi-Pack you want to order by checking next to it below. • Check the number of years you want to order. • Indicate your location by checking either U.S., Canada/Mexico or International. • Then choose which magazines you want to include with your Multi-Pack order.

LinuxWorld Magazine
U.S. - Two Years (24) Cover: \$143 You Pay: \$79.99 / Save: \$63 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72 You Pay: \$39.99 / Save: \$32
Can/Mex - Two Years (24) \$168 You Pay: \$119.99 / Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$84 You Pay: \$79.99 / Save: \$4
Int'l - Two Years (24) \$216 You Pay: \$176 / Save: \$40 + FREE \$198 CD
Int'l - One Year (12) \$108 You Pay: \$99.99 / Save: \$8

JDJ
U.S. - Two Years (24) Cover: \$144 You Pay: \$99.99 / Save: \$45 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72 You Pay: \$69.99 / Save: \$12
Can/Mex - Two Years (24) \$168 You Pay: \$119.99 / Save: \$48 + FREE \$198 CD
Can/Mex - One Year (12) \$120 You Pay: \$89.99 / Save: \$40
Int'l - Two Years (24) \$216 You Pay: \$176 / Save: \$40 + FREE \$198 CD
Int'l - One Year (12) \$108 You Pay: \$99.99 / Save: \$8

Web Services Journal
U.S. - Two Years (24) Cover: \$168 You Pay: \$99.99 / Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84 You Pay: \$69.99 / Save: \$14
Can/Mex - Two Years (24) \$192 You Pay: \$129 / Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96 You Pay: \$89.99 / Save: \$6
Int'l - Two Years (24) \$216 You Pay: \$170 / Save: \$46 + FREE \$198 CD
Int'l - One Year (12) \$108 You Pay: \$99.99 / Save: \$8

.NET Developer's Journal
U.S. - Two Years (24) Cover: \$168 You Pay: \$99.99 / Save: \$68 + FREE \$198 CD
U.S. - One Year (12) Cover: \$84 You Pay: \$69.99 / Save: \$14
Can/Mex - Two Years (24) \$192 You Pay: \$129 / Save: \$63 + FREE \$198 CD
Can/Mex - One Year (12) \$96 You Pay: \$89.99 / Save: \$6
Int'l - Two Years (24) \$216 You Pay: \$170 / Save: \$46 + FREE \$198 CD
Int'l - One Year (12) \$108 You Pay: \$99.99 / Save: \$8

Information Storage + Security Journal
U.S. - Two Years (24) Cover: \$143 You Pay: \$49.99 / Save: \$93 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72 You Pay: \$39.99 / Save: \$39
Can/Mex - Two Years (24) \$168 You Pay: \$79.99 / Save: \$88 + FREE \$198 CD
Can/Mex - One Year (12) \$84 You Pay: \$49.99 / Save: \$34
Int'l - Two Years (24) \$216 You Pay: \$89.99 / Save: \$126 + FREE \$198 CD
Int'l - One Year (12) \$108 You Pay: \$59.99 / Save: \$48

Wireless Business & Technology
U.S. - Two Years (12) Cover: \$120 You Pay: \$49.00 / Save: \$71 + FREE \$198 CD
U.S. - One Year (6) Cover: \$60 You Pay: \$29.99 / Save: \$30
Can/Mex - Two Years (12) \$120 You Pay: \$69.99 / Save: \$51 + FREE \$198 CD
Can/Mex - One Year (6) \$60 You Pay: \$49.99 / Save: \$10
Int'l - Two Years (12) \$120 You Pay: \$99.99 / Save: \$20 + FREE \$198 CD
Int'l - One Year (6) \$72 You Pay: \$69.99 / Save: \$2

MX Developer's Journal
U.S. - Two Years (24) Cover: \$143 You Pay: \$49.99 / Save: \$93 + FREE \$198 CD
U.S. - One Year (12) Cover: \$72 You Pay: \$39.99 / Save: \$32
Can/Mex - Two Years (24) \$168 You Pay: \$79.99 / Save: \$88 + FREE \$198 CD
Can/Mex - One Year (12) \$84 You Pay: \$49.99 / Save: \$34
Int'l - Two Years (24) \$216 You Pay: \$89.99 / Save: \$126 + FREE \$198 CD
Int'l - One Year (12) \$108 You Pay: \$59.99 / Save: \$48

ColdFusion Developer's Journal
U.S. - Two Years (24) Cover: \$216 You Pay: \$129 / Save: \$87 + FREE \$198 CD
U.S. - One Year (12) Cover: \$108 You Pay: \$89.99 / Save: \$18
Can/Mex - Two Years (24) \$240 You Pay: \$159.99 / Save: \$80 + FREE \$198 CD
Can/Mex - One Year (12) \$120 You Pay: \$99.99 / Save: \$20
Int'l - Two Years (24) \$264 You Pay: \$189 / Save: \$75 + FREE \$198 CD
Int'l - One Year (12) \$132 You Pay: \$129.99 / Save: \$2

WebSphere Journal
U.S. - Two Years (24) Cover: \$216 You Pay: \$129.00 / Save: \$87 + FREE \$198 CD
U.S. - One Year (12) Cover: \$108 You Pay: \$89.99 / Save: \$18
Can/Mex - Two Years (24) \$240 You Pay: \$159.99 / Save: \$80 + FREE \$198 CD
Can/Mex - One Year (12) \$120 You Pay: \$99.99 / Save: \$20
Int'l - Two Years (24) \$264 You Pay: \$189.00 / Save: \$75
Int'l - One Year (12) \$132 You Pay: \$129.99 / Save: \$2

PowerBuilder Developer's Journal
U.S. - Two Years (24) Cover: \$360 You Pay: \$169.99 / Save: \$190 + FREE \$198 CD
U.S. - One Year (12) Cover: \$180 You Pay: \$149 / Save: \$31
Can/Mex - Two Years (24) \$360 You Pay: \$179.99 / Save: \$180 + FREE \$198 CD
Can/Mex - One Year (12) \$180 You Pay: \$169 / Save: \$11
Int'l - Two Years (24) \$360 You Pay: \$189.99 / Save: \$170 + FREE \$198 CD
Int'l - One Year (12) \$180 You Pay: \$179 / Save: \$1

WLDJ
U.S. - Four Years (24) Cover: \$240 You Pay: \$99.99 / Save: \$140 + FREE \$198 CD
U.S. - Two Year (12) Cover: \$120 You Pay: \$49.99 / Save: \$70
Can/Mex - Four Years (24) \$240 You Pay: \$99.99 / Save: \$140 + FREE \$198 CD
Can/Mex - Two Year (12) \$120 You Pay: \$69.99 / Save: \$50
Int'l - Four Years (24) \$240 You Pay: \$120 / Save: \$120 + FREE \$198 CD
Int'l - Two Year (12) \$120 You Pay: \$79.99 / Save: \$40

*WHILE SUPPLIES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE

6-Pack
Pick any 6 of our magazines and save up to **\$340⁰⁰**
Pay only \$199 for a 1 year subscription plus 2 **FREE CDs**

- 2 Year – \$379.00
- Canada/Mexico – \$399.00
- International – \$449.00

9-Pack
Pick 9 of our magazines and save up to **\$270⁰⁰**
Pay only \$399 for a 1 year subscription plus 3 **FREE CDs**

- 2 Year – \$699.00
- Canada/Mexico – \$749.00
- International – \$849.00

Subscribe Online Today www.sys-con.com/2001/sub.cfm



Paths to SOA

HIGH ROADS, LOW ROADS, AND ROADS LESS TRAVELED



By Thomas Erl

Author Bio:

Thomas Erl is the founder of SOA Systems Inc. (www.soasystems.com), an enterprise solutions provider specializing in SOA consulting, planning, and training services. Thomas is the author of *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services* (Prentice Hall PTR), the best-selling book of 2004 in both SOA and Web Services categories, as well as the upcoming *Service-Oriented Architecture: Concepts, Technology, and Design* (Prentice Hall PTR). He is a member of OASIS and is active in related-research efforts, such as www.serviceorientation.org. Thomas also participates as a speaker and instructor for private and public events and conferences, and has published numerous papers.

Contact:

terl@soasystems.com

Many are comparing notes on two well-publicized paths to achieving SOA. The bottom-up approach is currently the most common variety, where Web services are created on an “as need” basis to fulfill mostly integration-related requirements. These services are typically application specific and simply re-create traditional integration channels over the open Web services communication framework.

The top-down approach, on the other hand, is one of analysis, deep thought, and patience. Service-orientation is infused into the business process layer so that services can be modeled in alignment with business models. The models themselves may need to be built or further refined in order to fully incorporate service-orientation principles.

Then, of course, there’s the middle ground – an approach that tries to balance the requirements of the top-down strategy with the efficiency of the bottom-up approach. Known as the “agile” or “meet-in-the-middle” strategy, this path is somewhat of a roller-coaster ride, and sure to challenge the most seasoned project manager.

This article explores these three approaches to building service-oriented solutions. Before we chart each path, though, let’s begin by establishing the common project phases associated with a generic SOA delivery life cycle.

SOA Delivery Life-Cycle Phases

Development projects for service-oriented solutions are, on the surface, much like other custom development projects for distributed applications. Web services are designed, developed, and deployed alongside standard components and the usual supporting cast of front and back-end technologies. Once you dig a bit deeper under the layers of service-orientation, though, you’ll find that in order to

properly construct and position services as part of a standardized SOA, traditional project cycles require some adjustments.

Looking at Figure 1, you may wonder why the first two-phase names are prefixed with “service-oriented” when the remaining phases have names that begin with just “service.” The main reason this distinction is made is because it is during the analysis and design stages that SOA characteristics and service-orientation principles are actually incorporated into the solution being built – so much so, that they warrant unique analysis and design processes that are distinctly “service-oriented.” The service phases are primarily concerned with the delivery of services that implement the results of service-oriented analysis and design efforts. Let’s now explain each of these life-cycle phases.

Service-Oriented Analysis

It is in this initial stage that we determine the potential scope of our SOA. Service layers are mapped out and individual services are modeled as service candidates that compose a preliminary SOA. (Formal service-oriented analysis and step-by-step service modeling processes are provided as part of Chapters 11 and 12 in *Service-Oriented Architecture: Concepts, Technology, and Design*.)

Service-Oriented Design

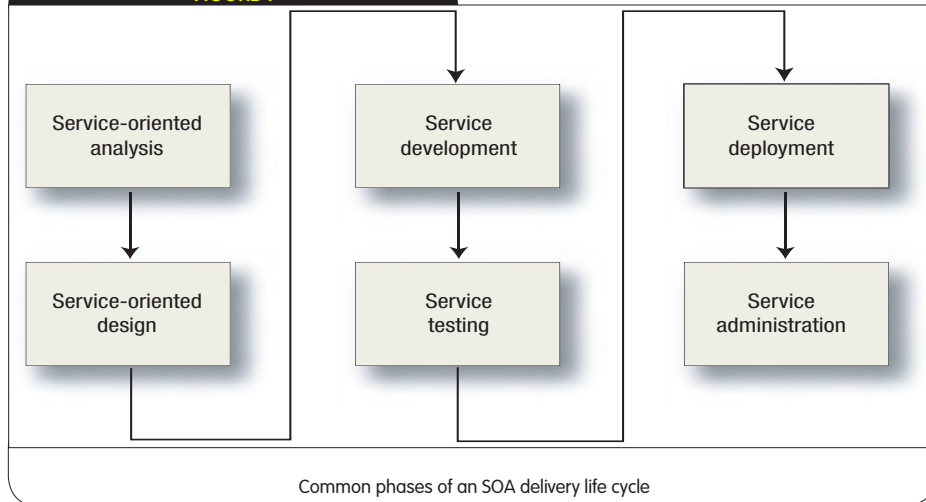
Once we know what it is we want to build, we need to determine how it should be constructed. Service-oriented design is a heavily standards-driven phase that incorporates industry conventions and service-orientation principles into the service design process.

This phase therefore confronts service designers with key decisions that establish the hard logic boundaries encapsulated by services. The service layers designed during this stage can also include the orchestration layer, which results in a formal business process definition. (Four step-by-step design processes are provided within Chapters 13 to 16 in *Service-Oriented Architecture: Concepts, Technology, and Design*.)

Service Development

Next, of course, is the actual construction phase. Here development platform-specific issues

FIGURE 1



Common phases of an SOA delivery life cycle

come into play, regardless of service type. Specifically, the choice of programming language and development environment will determine the physical form services and orchestrated business processes take, in accordance with their designs. (SOA support in .NET and J2EE platforms is explored in Chapter 18 of *Service-Oriented Architecture: Concepts, Technology, and Design*.)

Service Testing

Given their generic nature and potential to be reused and composed in unforeseeable situations, services are required to undergo rigorous testing prior to deployment into a production environment. Below is a sampling of some of the key issues facing service testers.

- What types of service requestors could potentially access a service?
- Can all service policy assertions be successfully met?
- What types of exception conditions could a service be potentially subjected to?
- How well do service descriptions communicate service semantics?
- Do revised service descriptions alter or extend previous versions?
- How easily can the services be composed?
- How easily can the service descriptions be discovered?
- Is compliance to WS-I profiles required?
- What data typing-related issues might arise?
- Have all possible service activities and service compositions been mapped out?
- Have all compensation processes been fully tested?

- What happens if exceptions occur within compensation processes?
- Do all new services comply with existing design standards?
- Do new services introduce custom SOAP headers? And, if yes, are all potential requestors (including intermediaries) required to do so, capable of understanding and processing them?
- Do new services introduce functional or QoS requirements that the current architecture does not support?

Service Deployment

The implementation stage brings with it the joys of installing and configuring distributed components, service interfaces, and any associated middleware products onto production servers. Typical issues arising during this phase include:

- How will services be distributed?
- Is the infrastructure adequate to fulfill the processing requirements of all services?
- How will the introduction of new services affect existing services and applications?
- How should services used by multiple solutions be positioned and deployed?
- How will the introduction of any required middleware affect the existing environment?
- Do these services introduce new versions of service descriptions that will need to be deployed alongside existing versions?
- What security settings and accounts are required?
- How will service pools be maintained to accommodate planned or unforeseen scalability requirements?
- How will encapsulated legacy systems

with performance or reliability limitations be maintained and monitored?

Service Administration

Once services are deployed, standard application management issues come to the forefront. These are similar in nature to the administration concerns for distributed, component-based applications, except that they may also apply to services as a whole (as opposed to services belonging to a specific application environment). Issues frequently include the following.

- How will service usage be monitored?
- What form of version control will be used to manage service-description documents?
- How will messages be traced and managed?
- How will performance bottlenecks be detected?

Note that though SOA governance comes into effect at this stage, it is a critical part of service-oriented solution delivery that must be planned for well ahead of time.

SOA Delivery Strategies

These life-cycle stages represent a simple, sequential path to building individual services. We now need to organize them into a process that can:

- accommodate our preferences with regards to which types of service layers we want to deliver
- coordinate the delivery of application, business, and process services
- support a transition toward a standardized SOA while helping us fulfill immediate, project-specific requirements

The last item on the above list poses the greatest challenge. The success of SOA within an enterprise is generally dependent on the extent to which it is standardized when it is phased into business and application domains. However, the success of a project delivering a service-oriented solution is generally measured by the extent to which the solution fulfills expected requirements within a given budget and timeline.

To address this problem, we need a strategy. This strategy must be based on an organization's priorities in order to establish the correct balance between the delivery of long-term migration goals with the fulfillment of short-term requirements.

FIGURE 2

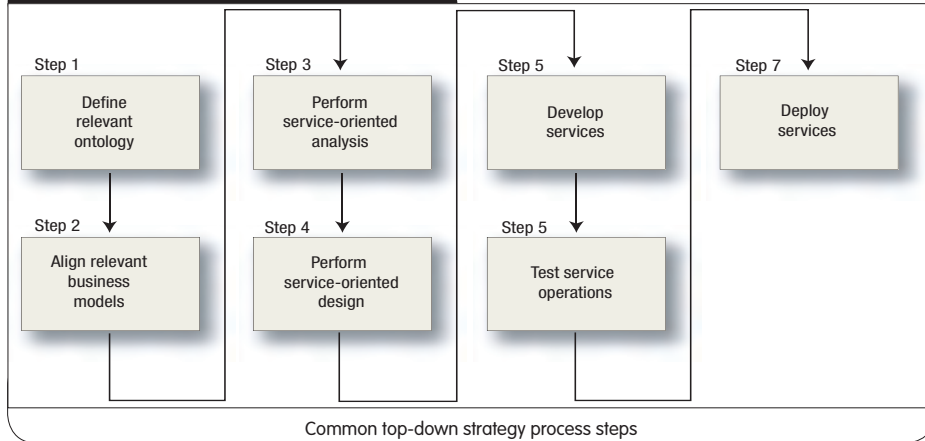
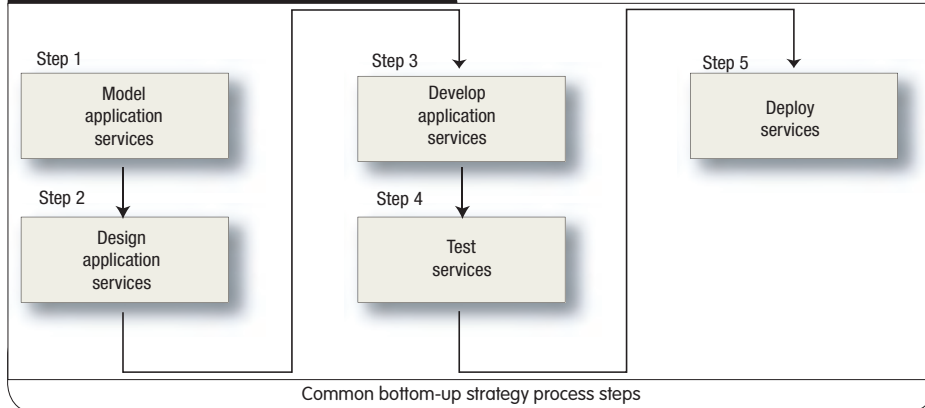


FIGURE 3



The top-down, bottom-up, and agile strategies we introduced at the beginning of this article each address this problem with differing priorities and practical considerations. The following three sections provide process descriptions and explore the pros and cons of each approach.

The Top-Down Strategy

This is very much an “analysis first” approach that requires not only business processes to become service-oriented, it also promotes the creation (or realignment) of an organization’s overall business model. This process is therefore closely tied to or derived from an organization’s existing business logic, and it commonly results in the creation of numerous reusable business and application services. The top-down approach will typically contain some or all of the steps illustrated and described in Figure 2. Note that this process assumes that business requirements have already been collected and defined.

Step 1: Define relevant enterprise-wide ontology

Part of what an ontology establishes is a classification of information sets processed by an organization. This results in a common vocabulary, as well as a definition of how these information sets relate to each other. Larger organizations with multiple business areas can have several ontologies, each governing a specific division of business. It is expected that these specialized ontologies all align to support an enterprise-wide ontology.

If such a business vocabulary does not yet exist for whatever information sets a solution is required to work with, then this step requires that it be defined. A significant amount of up-front information gathering and high-level business analysis effort may therefore be required.

Step 2: Align relevant business models (including entity models) with new or revised ontology

Once the ontology is established, existing business models may need to be adjusted (or

even created) in order to properly represent the vocabulary provided by the ontology in business modeling terms. Entity models in particular are of importance, as they can later be used as the basis for entity-centric business services.

Steps 3 and 4: Perform service-oriented analysis and service-oriented design

The aforementioned service-oriented analysis and service-oriented design processes are completed.

Step 5: Develop the required services

Services are developed according to their respective design specifications and the service descriptions created in Step 4.

Step 6: Test the services and all service operations

The testing stage will require that all service operations undergo necessary quality assurance checks. This typically exceeds the amount of testing required for the automation logic being implemented because reusable services will likely need to be subjected to testing beyond the immediate scope of the solution.

Step 7: Deploy the services

The solution is finally deployed into production. An implementation consideration beyond those we originally identified as part of this step is the future reuse potential of services. To facilitate multiple service requestors, highly reusable services may require extra processing power, and may have special security and accessibility requirements that will need to be accommodated.

Pros and Cons

The top-down approach to building SOA generally results in a high-quality service architecture. The design and parameters around each service are thoroughly analyzed, thereby maximizing reusability potential and opportunities for streamlined compositions. All of this lays the groundwork for a standardized and federated enterprise where services maintain a state of adaptability, while continuing to unify existing heterogeneity.

The obstacles to following a top-down approach are usually associated with time and money. Organizations are required to invest significantly in up-front analysis projects that can take a great deal of time

(proportional to the size of the organization and the immediate solution), without showing any immediate results.

The Bottom-Up Strategy

This approach essentially encourages the creation of services as a means of fulfilling application-centric requirements. Web services are built on an “as need” basis, and modeled to encapsulate application logic in order to best serve the immediate requirements of the solution. Integration is the primary motivator for bottom-up designs, where the need to take advantage of the open SOAP communications framework can be met by simply appending services as wrappers to legacy systems. A typical bottom-up approach follows a process similar to the one explained in Figure 3. Note that this process assumes that business requirements have already been collected and defined.

Step 1: Model required application services

This step results in the definition of application requirements that can be fulfilled through the use of Web services. Typical requirements include the need to establish point-to-point integration channels between legacy systems or B2B solutions. Other common requirements emerge out of the desire to replace traditional remote communication technology with the SOAP messaging communications framework.

For solutions that employ the bottom-up strategy to deliver highly service-centric solutions, application services will also be modeled to include specific business logic and rules. In this case, it is likely that two application service layers will emerge, consisting of hybrid and utility services. Those services classified as reusable may act as generic application endpoints for integration purposes, or they may be composed by parent hybrid services.

Step 2: Design the required application services

Some of the application services modeled in Step 1 may be delivered by purchasing or leasing third-party wrapper services, or perhaps through the creation of autogenerated proxy services. These services may provide little opportunity for additional design. Custom application services, though, will need to undergo a design process wherein existing design standards are

applied to ensure a level of consistency.

Step 3: Develop the required application services

Application services are developed according to their respective service descriptions and applicable design specifications.

Step 4: Test the services

Services, their associated solution environment, and underlying legacy logic are tested to ensure that processing requirements can be met. Performance and stress-testing measures are often used to set the processing parameters of legacy systems exposed via wrapper services. Security testing is also an important part of this stage.

Step 5: Deploy the services

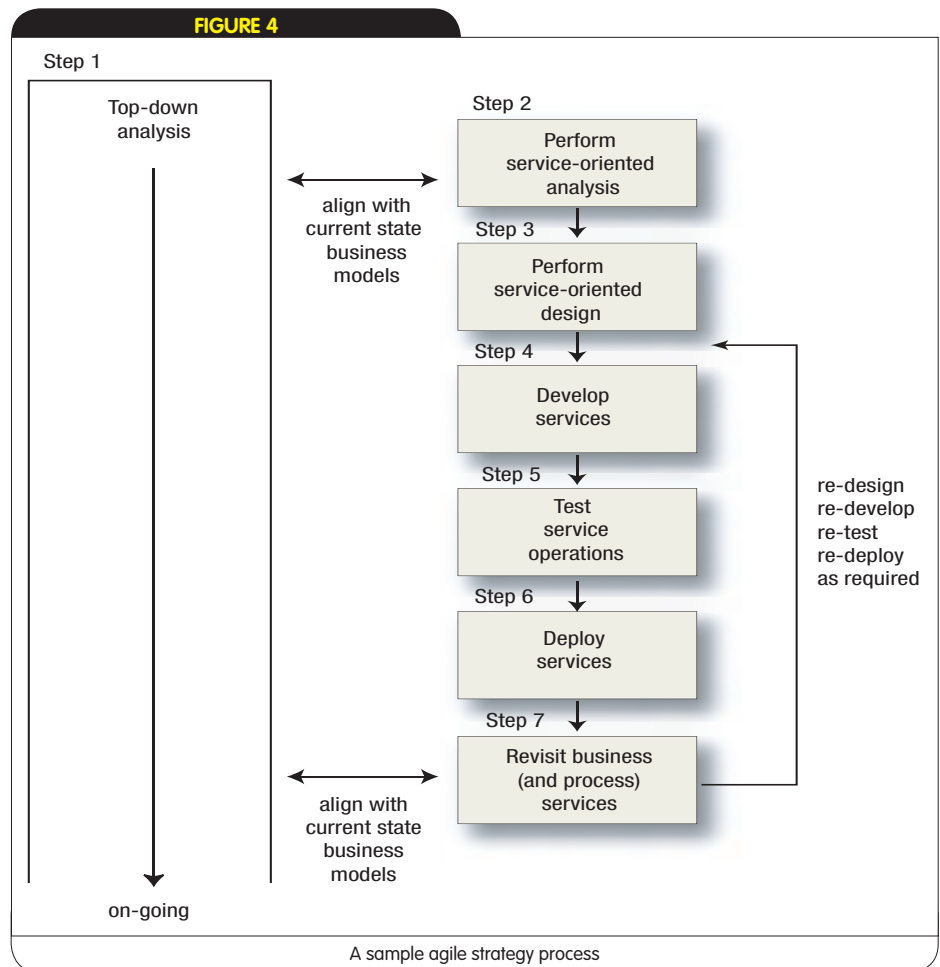
The solution and its application services are deployed into production. Implementation considerations for

application services frequently include performance and security requirements.

Pros and Cons

The majority of organizations that are currently building Web services apply the bottom-up approach. The primary reason behind this is that organizations simply add Web services to their existing application environments in order to leverage the Web services technology set. The architecture within which Web services are added remains unchanged, and service-orientation principles are therefore rarely considered.

As a result, the term that is used to refer to this approach – “the bottom-up strategy” – is somewhat of a misnomer. The bottom-up strategy is really not a strategy at all. Nor is it a valid approach to achieving contemporary SOA. This is a realization that will hit many organizations as they begin to take service-orientation, as an architectural model, more seriously. Although the bottom-up design allows for the efficient creation



of Web services as required by applications, implementing a proper SOA at a later point can result in a great deal of retro-fitting, or even the introduction of new standardized service layers positioned on top of the non-standardized services produced by this approach.

The Agile (“Meet-in-the-Middle”) Strategy

The challenge remains to find an acceptable balance between incorporating service-oriented design principles into business analysis environments without having to wait before integrating Web services technologies into technical environments. For many organizations it is therefore useful to view these two approaches as extremes, and to find a suitable middle ground.

This is possible by defining a new process that allows for the business-level analysis to occur concurrently with service design and development. Also known as the meet-in-the-middle approach, the agile strategy is more complex than the previous two, simply because it needs to fulfill two opposing sets of requirements. The process steps shown in Figure 4 demonstrate an example of how an agile strategy can be used to reach the respective goals of the top-down and bottom-up approaches.

Step 1: Begin the top-down analysis, focusing first on key parts of the ontology and related business entities

The standard top-down analysis begins but with a narrower focus. The parts of the business models directly related to the business logic being automated receive immediate priority.

Step 2: When the top-down analysis has sufficiently progressed, perform service-oriented analysis

While Step 1 is still in progress, this step initiates a service-oriented analysis phase. Depending on the magnitude of analysis required to complete Step 1, it is advisable to give that step a good head start. The further along it progresses, the more service designs will benefit.

Once the top-down analysis has sufficiently progressed, model business services to best represent the business model with whatever analysis results are available. This is a key decision point in

this process. It may require an educated judgment call in order to determine whether the on-going top-down analysis is sufficiently mature to proceed with the creation of business service models. This consideration must then be weighed against the importance and urgency of pending project requirements.

Step 3: Perform service-oriented design

The chosen service layers are defined and individual services are designed as part of a service-oriented design process.

Steps 4, 5, and 6: Develop, test, and deploy the services

Develop the services and submit them to the standard testing and deployment procedures.

Step 7: As the top-down analysis continues to progress, revisit business services

Perform periodic reviews of all business services to compare their design against the current state of the business models. Make a note of discrepancies and schedule those services most out of alignment for a redesign. This will typically require an extension to an existing service in order for it to better provide the full range of required capabilities. Once redesigned, a service will need to again undergo standard development, testing, and deployment steps.

In order to preserve the integrity of services produced by this approach, the concept of immutable service contracts

needs to be strictly enforced. Once a contract is published, it cannot be altered. Unless revisions to services result in extensions that impose no restrictions on an existing contract (such as the addition of new operations to a WSDL definition), Step 7 of this process will likely result in the need to publish new contract versions and the requirement for a version management system.

Pros and Cons

This strategy takes the best of both worlds and combines it into an approach for realizing SOA that meets immediate requirements without jeopardizing the integrity of an organization’s business model and the service-oriented qualities of the architecture.

While it fulfills both short- and long-term needs, the net result of employing this strategy is increased effort associated with the delivery of every service. The fact that services may need to be revisited, redesigned, redeveloped, and redeployed will add up proportionally to the amount of services subjected to this retasking step.

Additionally, this approach imposes maintenance tasks that are required to ensure that existing services are actually kept in alignment with revised business models. Even with a maintenance process in place, services still run the risk of misalignment with a constantly changing business model.

Conclusion

Choosing an appropriate delivery strategy is a critical part of SOA planning. Assessing these and other approaches will confront you with some important decisions. The choice of strategy will determine the extent to which your service-oriented modeling and design efforts can realize those aspects of SOA most relevant to your organizational priorities and goals. In other words, the path you take will ultimately determine your destination.

• • •

This article contains excerpts from *Service-Oriented Architecture: Concepts, Technology, and Design* by Thomas Erl (approximately 800 pages, hardcover, ISBN: 0131858580, Prentice Hall/Pearson PTR, Copyright 2005). For more information, see www.serviceoriented.ws.





Forget something?

Post-launch is NOT the time to be verifying web applications.

The wild blue yonder of operational monitoring and management is extremely unforgiving. Which means that going live with the monitoring software you used in development is a great way to go dead—quickly! You simply can't support operations if your staff is drowning in details provided by development profiling tools and debuggers. **Let NetIQ cover your apps...with AppManager.**

AppManager—the industry's easiest-to-use Systems Management suite—is a proven management system for monitoring J2EE application servers, databases, operating systems and even end-user response time. NetIQ's AppManager monitors ALL application components—not just your server. **NetIQ. Nobody does UNIX better. Nobody.**

Visit us at www.netiq.com/solutions/web to learn how we can help you address the challenges of your operational monitoring and management.



A man with a goatee and closed eyes is meditating in a server room. He is sitting cross-legged on the floor, wearing a light-colored polo shirt and khaki pants. The server racks are visible in the background, creating a sense of depth and perspective.

True application management starts from within.

Motive brings an enlightened approach to application management, building management intelligence into the application itself. Only Motive transcends the fragmented, disjointed reality of traditional application management, to deliver the visibility, insight and automation that support and development teams need to keep applications running smoothly.

This is no mere vision for the future. It's here now in application management products from Motive. Learn more about Motive's breakthrough approach in a new white paper about built-in management at www.motive.com/within1.

 **motive**
www.motive.com