SCA • ANTIPATTERNS • WEB SERVICES • NEWS

WebSphere® JOURNAL

www.WebSphere.SYS-CON.com

The World's Leading Independent WebSphere Resource

DECEMBER 2005 VOLUME 4 ISSUE 12

FROM THE EDITOR

No Predictions, Just Content

BY ROGER STRUKHOFF • PAGE 4

NEWS ROUND-UP

IBM Supports Open Document Format as Way to Energize Emerging Economies

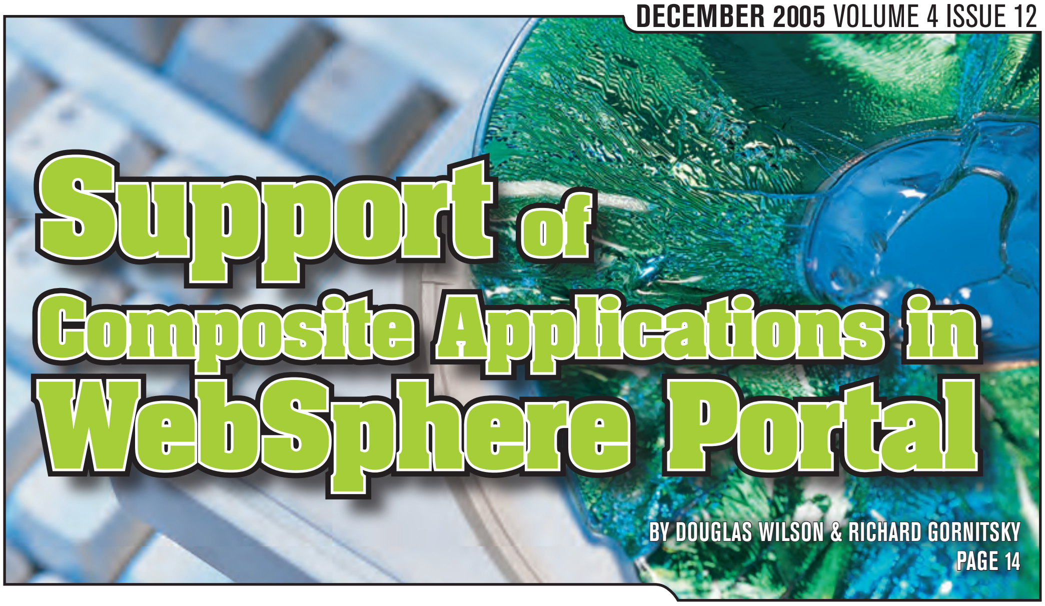
IBM, Oracle, BEA, Jobs, Microsoft, and Sun Come Out Gunning at SYS-CON Application Server Shoot-Out

PAGE 36

FINAL THOUGHTS

How to Keep Your Job

BY ROGER STRUKHOFF • PAGE 50



Support of Composite Applications in WebSphere Portal

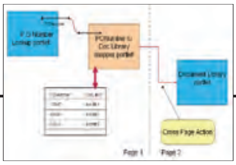
BY DOUGLAS WILSON & RICHARD GORNITSKY

PAGE 14

Strategies for Software Development Project Success

The Importance of Synchronization

BY GORAN **BEGIC** PAGE 6

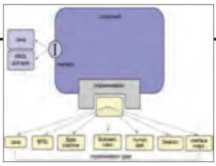


Building SOA Solutions with Service Component Architecture

Part One of a Two-Part Article

BY ROLAND **BARCIA** & JEFF **BRENT**

PAGE 20



SOA Antipatterns

The obstacles to the adoption and successful realization of Service-Oriented Architecture

BY JENNY **ANG**, LUBA **CHERBAKOV** & DR. MAMDOUH **IBRAHIM**

PAGE 26

Connect Non-SOAP HTTP Requesters and Providers to WebSphere Application Server V6 Enterprise Service Bus

WAS and ESB

BY GREG **FLURRY**

PAGE 38

NEW!

CodePro AnalytiX™

Imagine your software team creating higher quality, more reliable, Java systems... with less time and money.

Sound like a fantasy?

...**CodePro AnalytiX** lets you do exactly that, or

Your money back.

AnalytiX automates the process of building quality into your software as it's developed. Using the latest software development technology ***AnalytiX*** monitors Java code development in real-time, as its written. It identifies and automatically fixes hundreds of issues that would reduce quality and reliability.

Powerful wizards generate unit and regression tests and analyze code coverage, helping drive out problems early in the development process.

Less experienced developers code like experts, experts create tight, maintainable systems.

AnalytiX also allows you to define, distribute, and enforce corporate coding standards and quality measures across your development teams...no matter where they're located.

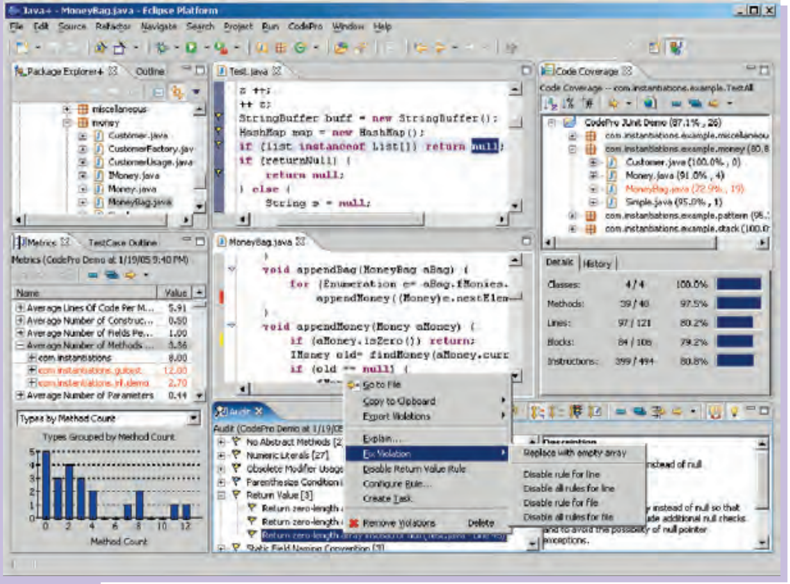
Early identification and correction of code quality issues dramatically improves your application development process and reduces the time and money it takes to develop your high performance Java systems.

From the development tool experts at Instantiations. No one outside of IBM has longer or deeper experience in creating Eclipse-based Java development tools.

Put our experience to work for your team.

www.instantiations.com
1-800-808-3737

Continuous Collaborative Code Analysis for Rational®, WebSphere® & Eclipse



```
z ++;
++ z;
StringBuffer buff = new StringBuffer();
HashMap map = new HashMap();
if (list instanceof List[]) return null;
if (returnNull) {
   return null;
} else {
   String s = null;
```

```
   }
   void appendBag(MoneyBag aBag) {
      for (Enumeration e= aBag.fMonies.
         appendMoney((Money)e.nextElem
   }
   void appendMoney(Money aMoney) {
      if (aMoney.isZero()) return;
      IMoney old= findMoney(aMoney.curr
      if (old == null) {
```

Metrics (CodePro Demo at 1/19/05 9:40 PM)

Name	Value
Average Lines Of Code Per M...	5.91
Average Number of Construc...	0.50
Average Number of Fields Pe...	1.00
Average Number of Methods ...	3.36
com.instantiations	8.00
com.instantiations.gubest	12.00
com.instantiations.jf.demo	2.70
Average Number of Parameters	0.44

Audit (CodePro Demo at 1/19/05

- No Abstract Methods [2
- Numeric Literals [27]
- Obsolete Modifier Usage
- Parenthesize Condition
- Return Value [3]
  - Return zero-length
  - Return zero-length
  - Return zero-length array instead of null
- Static Field Naming Convention [3]

Code Coverage -- com.instantiations.example.TestAll

- CodePro JUnit Demo (87.1%, 26)
  - com.instantiations.example.miscellaneou
  - com.instantiations.example.money (80.8
    - Customer.java (100.0%, 0)
    - Money.java (91.0%, 4)
    - MoneyBag.java (72.9%, 19)
    - Simple.java (95.0%, 1)
  - com.instantiations.example.pattern (95.
  - com.instantiations.example.stack (100.0

Classes:	4 / 4	100.0%	
Methods:	39 / 40	97.5%	
Lines:	97 / 121	80.2%	
Blocks:	84 / 106	79.2%	
Instructions:	399 / 494	80.8%	

...instead of null ... instead of null so that ... de additional null checks and to avoid the possibility of null pointer exceptions.

Key Product Capabilities

- Detect and correct code quality issues ...*Automatically*
- Define, distribute and enforce code quality standards across development teams
- 700+ rules and metrics/300+ Quick Fixes
- Code metrics with drilldown and triggers
- Custom rules and rule sets
- JUnit test case generation
- Code coverage analysis
- Dependency analysis and reporting
- Integrated team collaboration
- Javadoc analysis and repair
- Seamless integration with IBM Rational Developer, WebSphere Studio and Eclipse

Risk Free Trial Copy

www.instantiations.com/analytix

eclipse READY

IBM Business Partner

eclipse FOUNDATION MEMBER

instantiations

FROM THE EDITOR

No Predictions, Just Content

BY ROGER STRUKHOFF

As the year closes it's time for industry columnists to make their bold, wrongheaded predictions for 2006. You won't find that sort of nonsense here. There's no Top 10 or crystal ball or cute list of Things to Watch.

What you will find in this issue is more of the same. A couple more articles about SOA, a recap of the most recent product and alliance events regarding IBM, the long-awaited (and marvelous) final installment about Composite Applications, a WAS article, and some material about software development in general.

As we move into 2006, keep expecting the same steady stream of articles about the various aspects of developing apps with IBM WebSphere. WebSphere cuts across all industries. It is involved in all aspects of governmental application development. And it is involved in a very competitive space characterized by technological innovation and marketing hype—how "proprietary," "commercial," "blended," or "open" are your requirements and how important are these terms to what environment you choose?

The answer to the last part of this question is probably "not very," and market research continues to show WebSphere as the clear leader in web services application development.

That said, some words about SOA are needed. As more companies, either reflexively or deterministically, decide to integrate service-oriented architectures into their app development strategy and tactics, the innovation and competitive issues can be expected to continue. SOA is still in its infancy, a SOA product per se is obviously not something that one can buy, and there is no precise specification for what a SOA is.

Yet SOA is as a SOA does, so companies will continue to define their needs and turn to IBM and its competitors to help fill these needs. At *WebSphere Journal*, we will follow that story, as well as the many internal dynamics involving WebSphere, as well as some of the people involved. We are planning a full "Interview Issue" for mid-year, as well as focuses on different aspects of WebSphere *besides* incorporating it for SOA development as the year progresses.

In that spirit, we do need to let our audience know that we are always on the lookout for good article and interview ideas. Address your proposals to editorial@sys-con.com and be specific about what you want to write about or whom you want us to interview. If you have created a good technical article that is looking for a place to be published, by all means send it. We do accept unsolicited articles on occasion.

OK, time for one prediction. As we move through 2006, expect IBM to continue market leadership in the web services application development space, even as competitive pressures from companies such as BEA, Jboss, and Microsoft remain strong. Not much of a prediction, to be sure, and not one that goes out on any sort of limb. But who cares about predictions? The day-to-day reality of app development is that there is no time for naval gazing or facile prognostication. There is work to be done!

ABOUT THE AUTHOR

Roger Strukhoff, editor-in-chief of *WebSphere Journal*, is West Coast Bureau Chief for the SYS-CON News Desk, and President of www.wdva.com. He spent 15 years with Miller Freeman Publications and The International Data Group (IDG), then co-founded CoverOne Media, a custom publishing agency that he sold in 2004. His work has won awards from the American Business Media, Western Press Association, Illinois Press Association, and the Magazine Publishers' Association. Read his blog at http://www.rssblog.linuxworld.com.

roger@sys-con.com

WebSphere® JOURNAL

Advisory Board

Richard Arone, David Caddis, Mike Curwen, Devi Gupta, Lloyd Hagemo, Barbara Johnson, Shannon Lynd, James Martin, Doug Stephen, Victor Valle

Editor-in-Chief	ROGER STRUKHOFF
Contributing Editor	PATTI MARTIN
Product Review Editor	JAY JOHNSON
Editor	NANCY VALENTINE
Associate Editor	SETA PAPAZIAN

Subscriptions

For subscriptions and requests for bulk orders,
please send your letters to Subscription Department.
888-303-5282 201-802-3012
SUBSCRIBE@SYS-CON.COM
Cover Price: $8.99/Issue Domestic: $149/YR (12 Issues)
Canada/Mexico: $169/YR Overseas: $179/YR
(U.S. Banks or Money Orders)
Back issues: $15 U.S. $20 All others

PRESIDENT AND CEO	FUAT KIRCAALI
PRESIDENT, SYS-CON EVENTS	GRISHA DAVIDA
GROUP PUBLISHER	JEREMY GEELAN
SENIOR VP, SALES & MARKETING	CARMEN GONZALEZ
VP, INFORMATION SYSTEMS	ROBERT DIAMOND
VP, SALES & MARKETING	MILES SILVERMAN
PRODUCTION CONSULTANT	JIM MORGAN
FINANCIAL ANALYST	JOAN LAROSE
CREDITS & ACCOUNTS RECEIVABLE	GAIL NAPLES
ACCOUNTS PAYABLE	BETTY WHITE
ADVERTISING DIRECTOR	ROBYN FORMA
ADVERTISING MARKETING DIRECTOR	DENNIS LEAVEY
ADVERTSING SALES MANAGER	MEGAN MUSSA
ADVERTISING SALES MANAGERS	DOROTHY GIL KERRY MEALIA
LEAD DESIGNER	ABRAHAM ADDO
ART DIRECTOR	ALEX BOTERO
ASSOCIATE ART DIRECTORS	ANDREA BODEN TAMI BEATTY LOUIS F. CUFFARI
VIDEO PRODUCTION	FRANK MORICCO
WEB DESIGNERS	STEPHEN KILMURRAY VINCENT SANTAITI SHAWN SLANEY
ONLINE EDITOR	ROGER STRUKHOFF
CIRCULATION SERVICE COORDINATORS	EDNA EARLE RUSSELL LINDA LIPTON
CUSTOMER RELATIONS MANAGER	BRUNI STAROPOLI

Editorial Offices

SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645
Telephone: 201 802-3000 Fax: 201 782-9637
SUBSCRIBE@SYS-CON.COM
WebSphere® Journal (ISSN# 1535-6914)
is published monthly (12 times a year).
Postmaster send address changes to:
WebSphere Journal, SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645

WORLDWIDE NEWSSTAND DISTRIBUTION
CURTIS CIRCULATION COMPANY, NEW MILFORD, NJ

NEWSSTAND DISTRIBUTION CONSULTANT:
GREGORY ASSOCIATES, W.R.D.S.
732 607-9941, BJGASSOCIATES@CS.COM

FOR LIST RENTAL INFORMATION:
KEVIN COLLOPY: 845 731-2684, KEVIN.COLLOPY@EDITHROMAN.COM
FRANK CIPOLLA: 845 731-3832, FRANK.CIPOLLA@EPOSTDIRECT.COM

SYS-CON MEDIA

Application: Operations Monitor
Customer Transactions per Second
Debit Account
Bill Payment
Authenticate
Account Info
Cardholder Service
Loan Processing

The Importance of Synchronization

Strategies for Software Development Project Success

BY GORAN BEGIC

A seasoned software development professional offers personal advice and describes best practices for improving project success, touching on communication, use cases, testing, and marketing.

ABOUT THE AUTHOR

Goran Begic is a Senior IT Specialist with IBM.

While working on both in-house and external software development projects in the last couple of years, it occurred to me that certain challenges and best practices kept repeating across different projects. As I learned how to swim through the challenges and ride the waves of the best practices, I decided to make some notes, which I will summarize in this article.

In this part one of two series, I will examine two categories:

1. Compensating for lack of face-to-face communication; and
2. Writing better use cases.

This rather idiosyncratic advice is by no means comprehensive. My intent is not to teach you how to build software, but rather to discuss a few technical and non-technical issues that are critical to project success. Often, seemingly unimportant factors can make the difference between a good project and an excellent one. My advice focuses on the relatively small portion of overall project effort that these crucial factors typically represent.

I. Compensating for lack of face-to-face communication

Everyone knows that the single most important success factor for a project is good communication. This is a huge topic, but I will focus on one aspect that can really make a difference.

The benefits of face-to-face communication

Blessed are the teams that reside in a single location. Often, this is the case for small, startup businesses in which a couple of determined and ambitious people share the same goals and experience the same details of everyday life. Also, some larger organizations that effectively use agile methods base their internal process on face-to-face communication. When a team resides in a single building, people can meet face to face during lunch, in the corridor, and after hours, and these informal meetings are often more productive than regular scheduled meetings. I can remember numerous times when I copied notes from a crowded whiteboard after a long discussion and then used these shared ideas as the basis for a new plan.

However, business success and growth often makes it impossible to confine a high-quality team to a single location. Acquisitions, joint projects, and offshore development create demand for successful communication across different time zones and different cultures. In such cases, what are the best alternatives to face-to-face meetings?

Communication in a distributed environment

The first answer that came to mind was probably “email.” I’ve found that email is a really useful communication channel, but it’s also the first channel to be abused. What use are mile-long email threads when there is no time to read them? A good rule of thumb: If a thread contains more than three messages, you should probably switch to the telephone. This little rule has saved me numerous hours of writing pointless text in Lotus Notes.

In addition, there are other communication channels that -- although they cannot replace the intimacy of face-to-face meetings – can improve collaboration and understanding within a team:

- Project home page. Setting up a home Web page gives project leaders an excellent channel for one-way communication (or two-way communication, if they use wikis or forums/blogs). A Web page can explain what the project is, define main goals, and introduce team members. Managers can also use this page to post coding standards, "how to's" that will help team members set up the environment, and so forth.

- Online chat. This form of communication creates a virtual collaboration environment and fosters an easygoing exchange of ideas among team members. It is typically much more efficient to ping a colleague and ask for help via an instant messaging system than to play phone tag with that person. For example, IBM's system, Lotus Notes SameTime, integrates with the Lotus Notes directory, so you can easily find the right contact in your organization and see whether that person is logged onto SameTime. If you see a "Busy" indicator, you can assume that the person would not answer a phone call, either. SameTime also allows you to invite multiple people into a chat, so it's a wonderful way to conduct joint decision making on the spur of the moment, without going through the bother of setting up a formal meeting.

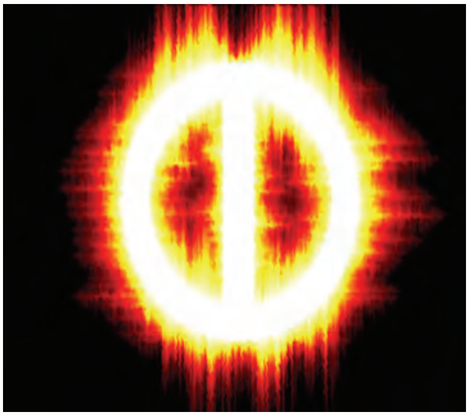
Some chat programs include other interesting features such as customized availability messages, the ability to save transcripts for future reference, and so forth. Keep in mind that different chat tools provide different levels of security, not all of them equally well-suited for professional use.

- Defect tracking system. This is another effective communication channel. Every software development project must cope with feature enhancement requests and bug reports at some point. An effective defect tracking system, such as IBM Rational ClearQuest, can help teams bring order to these requests and reports. In particular, the Unified Change Management (UCM) capabilities that you get by integrating Rational ClearQuest with IBM Rational ClearCase, allow you to associate activities with defect records and source code, which greatly improves the quality of communication on day-to-day issues. For example, if I need help with a component that isn't working right, I can look up tickets for that component in the defect-tracking database and find out whether the component is undergoing further work, as well as who is doing that work.
- "A cup of coffee." On the other side of every communication tool is a person. Isn't it great when someone calls you to just ask how you are doing, especially when you're working under pressure? Although your colleagues may be on the other side of the globe, it's important to remember how important these informal conversations can be for consolidating a team effort. Even if you are using the most sophisticated management and reporting tools, sometimes the only way to get the right information is by talking with another person. How do you invite your distant colleague for a cup of coffee? Maybe an unscheduled phone call shortly before the end of the day would do. Remember: You'll get such calls more often if you initiate them often yourself.

2. Writing better use cases

From a certain perspective, we can think of use cases as a form of communication. However, they are much more than that, so I treat them separately here.

Use cases have two enormous advantages over other system planning techniques. First, they give everyone involved in the project a correct understanding of the final user's needs and requirements. Second, you can assemble them long before the first prototypes of the developed code are ready. Knowing the application's key elements and the user's needs opens up the door for fruitful collaboration with the customer on one side, and between the different roles and teams involved in product assembly on the other side.



Simply put, a use case is a fairly simple document that describes either:

- Key scenarios that users must execute as part of their work (business use case).

or

- Interactions of the user (actor) with the system used for business (system use case).

Perhaps you are thinking, "So what? That's just another document." Actually, I thought the same thing before I first faced the challenge of having to put requirements and users expectations into a document that could be easily shared across different project roles. Very early in the project, the testing team and the documentation team started asking detailed questions on what we would be doing and how everything would work together. Until I discovered use cases, my answers were inconsistent, confusing, and very time consuming.

Of course, once I discovered how useful use cases could be for planning and answering these questions, I faced another challenge: learning how to write use cases effectively.

Know the user

Before jumping into interactions and scenarios, it is important to understand all of the targeted customer groups, differentiate these groups from one another, and specify their requirements and needs. Different user groups have different business use cases. For example, it is very likely that a tester tasked with validating compliance with a certain set of globalization standards will have different needs than a developer with a similar responsibility. The developer might have the knowledge, tools, and authority to fix some deviations right away, but the tester would likely have to file the defect and then wait for the developer to fix it.

Unless you understood the specific needs of these two user groups when you were designing your application, you would risk creating a system that would not satisfy the requirements of either one.

Business use cases versus system use cases

Depending on the system in question, there may be a significant difference between business and system use cases. One depicts a user's business and the other a user's interaction with the system. However, the difference between these two types of use cases can become blurry. For example, in a software development business, the system IS the business. While you are developing and testing a specific system feature, it is easy to forget that the final user may not use the system the same way you do – or even have the same needs.

To understand the distinction between a business use case and a system use case, imagine that a software team has been charged with developing a software application that must perform equally well in different local environments around the globe. To ensure that the application meets these requirements, the team must comply with a set of minimum globalization rules.[1] These rules are hard-coded in a set of specifications that cannot be modified at will. The development team needs to audit the code against these rules, and currently they must do this manually. A programmer (auditor) must open each source file and find any inconsistencies with the specified rules.

Here is an example of the business use case. It is one of the business use cases for reviewing some aspect of code quality, often a time consuming manual activity.

For each source file in the workspace:

1. Open the source file.
2. Read the code line by line (note: requires deep coding knowledge).
3. If you find a suspicious method, consult the Web page containing the rules (note: very time consuming).
4. Resolve the problem by fixing the code, if possible (business use case for the developer).
   a. Alternative flow for the tester role: If you cannot fix the problem right away, document the finding in a Word document and submit it to the developer.
5. Open the next source file and repeat the steps.

As you can see, this is a tedious process – some steps in the use case are manual and prone to user errors and therefore good candidates for automation to speed the process up. The focus of the business use case is the process with, or without any tools.

On the other hand the system use case focuses on the interactions of the user with the system – in our example a tool for automated validation of source code.

Example system use case

Precondition: Globalization rules are enabled in the code review feature of the IDE.

1. Open the code review tool in the IDE.
   a. (Steps for opening the tool).
2. Run code review.
   a. Main flow -- click on the button "Run."
   b. (List of alternative flows described in detail in a new section).
3. Review the results of the automated review (findings).
4. For each finding, resolve the finding:
   a. Main flow: Submit a defect.
   b. (List of alternative flows described in detail in a new section).
5. If you cannot fix the code right away, submit a defect.
   a. (Steps for defect submission).

Often a system use case has a purpose of improving the business scenarios through the introduction of tools and automation that help improving the effectives of the user's business. A good system use case fits well in the user's business scenarios and use cases and improves them. The comparison of the two also provides an insight into the return of the investment (ROI) of the planned system.

The value of system use cases

Once you define the user's "as is" process and goals, you can specify what portion of the user's activities an automated system can supply. A short document with steps such as the ones described above would provide plenty of information to the reader on what the final product should look like, how it would help the customers with their business, and so forth -- even if this information is not explicitly spelled out.

Here are some obvious benefits of having a system use case for code reviews:

- Even without having a line of code classified as "code review ready for testing," a tester can go ahead and create a test plan for the functionality described above.
- In addition, at this point, a documentation person could go ahead and create and/or append the structure of the documentation for the code review.
- The development manager could then optimize the development schedule, so that the first development build would satisfy at least the main flow of this use case.

Use cases accelerate the development project and engage all team members right at the beginning of the project. If you don't already focus your work around use cases, give them a try. The effort is well worth it.

Getting the right level of detail for system use cases

How much information should you include in each use case step?

My preference is to derive and group the system use cases around the business use cases and also group a number of user interactions under one group of steps. The group of steps explains the purpose of the step and is likely to remain unchanged if some detail in the implementation changes. This approach helps to make the document more readable, it connects the system use case with the business use case, and it allows the use case to be easily modified if the product implementation changes.

For example, the simple steps in our system use case above, "Run code review," convey a wealth of information to the careful reader, including:

- Code review is integrated in the IDE.
- Code review must be engaged, or "run."
- Code review provides a list of findings about deviations from the set of validated standards.
- Some of the findings have quick fixes.
- Code review will have quick fixes.

3. Ensuring effective testing

Once the use cases are set in place and

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

the team has agreed that they represent the right way to go, the use cases become the foundation for the rest of the plan. In fact, this is the only way to take advantage of the benefits they offer.

The engineering team builds a development plan that includes, at the very least, a list of components to be built and a timeframe for each of them. It is very important to create clear traceability between features needed for the main use cases and the components necessary for the features to work.

Identifying these core components and defining their use cases are crucial steps that allow early testing of the application functionality. If the core components are delivered early in the development cycle, then the tester can start writing test scripts for the basic set of rules and validate that the tool functions properly.

In our example, the system use case “Run code review” enabled the tester to make a test plan for this core functionality even before the code was written and also to create a set of manual test scripts for both the main flow and alternative options.

Types of testing

The simplest form of testing -- and a very effective one -- is to assemble a number of educated users to exercise various features of the application under test and report issues (findings, defects) to the code development team. The metrics for this form of validation are simple: The more users you have, the more defects you’ll detect. Different user groups will use the tool in different ways and further improve the number of detected problems. However, there are some issues related to this. By the time the software is ready for user consumption, there may not be enough time to launch an extensive test program. Different users may be on different product builds. Even more important, depending entirely on human beings is very expensive and very unreliable.

There are still more things to consider. Without a clear testing plan, it is impossible to assume that the same features of each version of the application will be tested the same way. Without assistance from testing tools, the sequence of test steps will most likely differ for each test, and there will be slippage on some rare -- but potentially important and costly -- scenarios.

Manual testing and regression testing

Manual testing refers to a set of actions aimed at validating a specific system response. The alternative to manual testing is automated testing. Both are types of functional testing. Automated testing implies that you use a specialized tool or batch script to exercise a set of application components, record the application’s response, compare it against expected values, and decide whether the test was successful or not -- all without any human intervention. Automation gives you the ability to repeat the same tests over and over again, with much better precision than humans would ever be able to achieve. Examples of automated functional testing tools are IBM Rational Robot and IBM Rational Functional Tester.

Regression testing, which is used to measure application quality, can be either manual or automated. You can assemble a regression testing suite from automated functional tests or from automated developer tests (see below). The key element for reliable regression testing is exact repeatability for these tests. Therefore, it is necessary to precisely define test steps in documents called test scripts, and then follow these exact steps during each regression test. Then, you can confidently use the test results not only to report problems, but also to measure quality.

There is a high correlation between success in testing and the amount of time you invest in test planning, documenting manual tests, and automation. Here are some specific suggestions for effective testing:

- Define your test plans around use cases. Start testing the main use-case flows first, and then expand into alternative flows once you cover all the main use-case scenarios. The key is maintaining the proper granularity and modularity for your use cases, as described above.
- Organize your manual tests around a test plan, and start documenting and analyzing test results in a uniform way with the first batch of tests that you implement. Repeatable, uniform execution (even if manual) will improve the quality of metrics that you collect over time.
- Automate first the tests that have relatively simple possible execution paths but require a lot of data to be entered with each run. Feed the test scripts with ready-made data pools (i.e., do data-driven testing).

Developer testing

Often, there is a big obstacle to converting use cases to effective tests: A large portion of the code base is not available for functional testing until late in the development cycle. Therefore, it would be good if some components were tested before they were assembled into the running application. This is where developer testing fits in.

Developer testing is a set of activities focused on improving code quality and often conducted by a developer. Developer testing has two main aspects:

- Automated unit and component testing. This includes code reviews, unit/component testing, and code-coverage analysis.
- Manual testing and debugging. This includes execution trace, assertions, memory leak detection and memory usage analysis, performance profiling, thread analysis, and so forth.

Automated batch tests with dedicated tools such as JUnit -- a code review tool integrated in IBM Rational Application Developer -- or IBM Rational PurifyPlus provide an additional means to ensure high-quality software. Finding and fixing defects in the development environment means fewer functional problems later on. And that leaves more time for writing code and introducing more automation. In addition, having reliable repeatability for these automated unit tests and code reviews allows you to collect valuable metrics about code quality, with the same benefits described previously.

For most organizations, the main hurdle to implementing developer testing is the learning curve for the required tools. Often, individual developer testing tools focus on a rather narrow aspect of software quality. If team members do not have experience with automated developer testing, then finding the right tools and deciding what types of tests to automate can present con-

siderable challenges. Therefore, the development plan should not only build in time for developing and debugging application components, but also dedicate time and resources for the training, setup, analysis, and reporting involved in implementing automated development tests. This initial investment will quickly bring returns by reducing the number of functional problems left for the dedicated testing teams to detect. It will also raise the level of understanding of the code base among team members.

Here are suggestions for getting started with developer testing:

- If you are doing unit testing, start collecting code coverage data while running the unit tests to assess the completeness of the unit testing suite.
- If you are developing C++ applications, run your key use cases with the tool for analysis of dynamic memory allocation. Memory corruption problems are one of the key problem areas in all native C/C++ applications, and the root cause of many unexpected and hard-to-reproduce defects.
- Collect performance baselines for the methods in your components for at least each integration build, and monitor the development of performance over time.
- If you are developing a Java/J2EE application, monitor the memory usage and the number and types of objects in memory during a couple of basic smoke tests.
- Apply a handful of the most important static analysis rules at the beginning, use them against each build, and add more rules for your code base over time. This will reduce the number of false positives in the results.

Developer testing doesn't replace functional regression testing – manual or automated. It simply improves the effectiveness of testing as a whole, by detecting problems in the development environment, when they're relatively easy to fix.

Positive versus negative testing

Positive tests focus on validating an application's main flows, as defined and prioritized in the use-case documentation. Negative tests often focus on testing the application's capability limits (i.e., on "breaking the application").

In my opinion, a good test plan should clearly focus on validating use-case paths, but include a healthy dose of limit testing. Often, you can assess limits through data-driven testing, which applies different data sets against a single test case in order to validate the application's response to a certain problem or exception – non-standard character sets, for example.

4. Supporting marketing efforts

In today's business-driven, dynamic environment, understanding the software market is key to project success. In fact, an organization's degree of understanding may have an even bigger impact on success than the level of communication, project planning, quality, and testing that you achieve. Even a perfectly organized, well-tuned software development organization will fail if its product misses the needs of the market or targets the wrong audience.

So how can those who are doing the engineering work with people in marketing to improve a product's chances for success? Below are some suggestions.

Driving beta programs and customer references

The most effective way for software teams to help market a new product is to obtain positive references from early customers and share them with prospects via their organization's marketing channels. No matter how skilled or creative your marketing people are, they can't collect success stories overnight. It takes time for customers to familiarize themselves with the new tool and apply it. Therefore, planning for the new project should include a significant amount of time and resources for an early adoption program.

Begin at the outset of the project by engaging the marketing team, a number of field representatives, and a customer from your target audience in discussions about features and other requirements for the new product. Get in touch with key customers who requested enhancements in your product, present a plan and timeline for a beta program, and make sure that this plan includes the customer reference as a target. By communicating this requirement ahead of time, you can help set correct expectations for the program and improve the chances of meeting your marketing timeline. Often, customers must go through a number of legal hoops and approvals before they can provide a reference that will help you highlight the new solution's benefits.

Early collaboration

It's also a good idea for software engineering to start working with the marketing organization early in a project, because marketing people can help you obtain the information necessary to create effective use cases. They can help you define your target audience and market because they constantly work with focus groups and analysts who maintain a clear picture of the current market. At the same time, the work you do to define goals for a feature or product release will yield well-prepared documentation that can help the marketing organization understand the product early on and enable it to describe and promote the product accurately. The key word is collaboration.

A good practice is to assemble a collaborative project team to help everyone in the organization understand what needs to be done. In addition to gathering the marketing input described above, this collaborative team should also do the following:

- Recognize key sponsors and engage them in the planning phase (review of goals, vision, business goals, and use-case scenarios). The sponsors can be external investors, potential customers, or even groups within your organization.
- Get information about the client's support requirements, including their level of expertise and training needs.
- Get information from the sales team about the desired availability date, key customers, and major concerns.
- Assemble presentation material to introduce other parts of the organization to the project so that they can begin participating. Collaborative team members should stay up to speed on project goals, plans, and activities throughout the project lifecycle so that they can continue to educate interested parties.

– continued on page 42

Imagine seeing EAI's impact on business process performance in real-time...

Now imagine implementing it in minutes, *instead of* months

To get your free copy of the informative new article

"3 Steps to Uncomplicating Business Process Management"

go to www.nastel.com/uncomplicate
or call 800-580-2344 or
+631-761-9100

Business happens in real time.
If something impacts your business processes,
you need to know. *Now.*

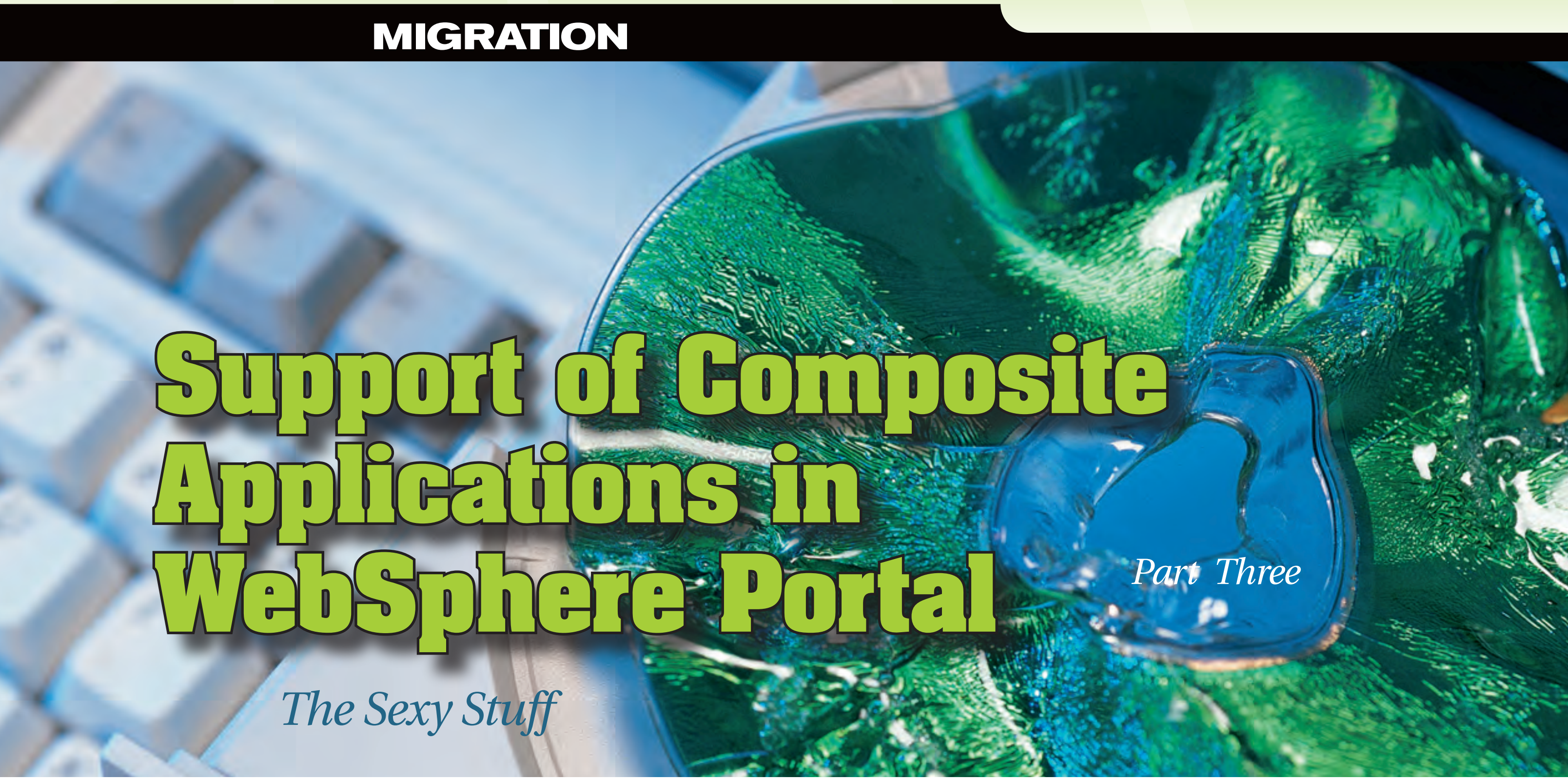
AutoPilot™ from Nastel monitors and manages all of the complex events that comprise your business-critical transactions, giving you deeply granular visibility from end to end for all your business processes. AutoPilot lets you define customized, actionable alerts in the context of each user's specific business function to streamline your business processes and eliminate costly failures, bottlenecks and redundancies. With AutoPilot, you can see the impact of EAI on your business process performance in real time, on-demand.

Unlike other monitoring software, though, AutoPilot's unique architecture takes the complexity out of implementation. It doesn't require months of programmers and consultant's time before you reap the benefits. With AutoPilot, you simply point and click, and within minutes you're in total control.

NASTEL™
TECHNOLOGIES, INC.

Put Your Application Infrastructure On AutoPilot

MIGRATION



Support of Composite Applications in WebSphere Portal

Part Three

The Sexy Stuff

BY DOUGLAS WILSON & RICHARD GORNITSKY

In the first article of our three part series, we introduced you to the composite application model and how WebSphere Portal supports it. A composite application is a class of application whose fundamental construction model is the composition of parts or components. It is formed by choosing components from a catalog or palette, placing those components on some design surface, and interconnecting the components to create the behavior desired.

ABOUT THE AUTHOR

Douglas Wilson & Richard Gornitsky are distinguished industry professionals

Composite applications within WebSphere Portal are based on portlets. Portlets can be considered to be modules of the composition model since they are each self-contained applications that support multiple states and view modes. While simply exposing business data and operations through a portlet offers a lot of value, even greater value can be obtained when portlets work together. To create a composite application they must be able to pass data to each other, react to the data, trigger events, and respond to events. Communication and interaction with other composite applications are through defined interfaces such as messaging, event handling, or services. Each portlet can be removed/added from the model or page without impacting other portlets. The foundation of the composition model is the portal infrastructure which provides access to user profile information and remote content, the ability to communicate and trigger other portlets and to participate in windows and action events, store persistent data, and look up credentials.

In the second article, we showed how to make composite programs with co-operative portlets, both programmatically and with the Portlet Wiring Tool. We showed how easy it was to add a third party portlet on to composite portlet and how IBM was committed to this technology.

In the last article of this series, we will dwell into some advance techniques that will highlight the power of composite applications.

Filters and Adaptors

Composite applications are made up of loosely coupled portlets or co-operative portlets. However sometimes output or input properties change and you will want to integrate the portlets without having to touch the original source code. You may not even be able to touch the source code since you don't own it.

This is where Filters/Adaptor portlets come in. The Filter/Adapter portlet massages an output property of a co-operative portlet such that it is recognizable by the input property of another co-operative portlet. You place the Filter/Adaptor portlet in a 1 pixel container on a Portlet Page (which makes it appear invisible) and wire the source portlet to the Filter/Adapter portlet and then wire the Filter/Adaptor portlet to the target portlet.

So let's show you how to do it! We are going to use the property broker shipping example that comes with WebSphere Portal. The source code can be found in [WP-HOME]/installableApps. The shipping example consists of five C2A portlets. The first portlet, Orders, displays order information for a given month and provides information on order id, customer id, and status. To find more detail information, click on the icon associated with the field you are interested in.

Now for our example (and for simplicity sake), we are going to pretend that the customer 2867377 has merged with customer 3542877 and has been assigned its customer number. So rather then recoding the target or the source portlet or change the database, we are going to create a portlet that will intercept the value transmitted by the source portlet, massage the value, and re-transmit it to the target portlet. Basically this is a poor man's WebSphere Message Broker.

No changes have to be made to the original target and source portlets (order and customer detail) however for clarity sake the type associated with customer id should be made for both the input and output. So the input and output of Customer Id type associated with the order portlet and the composite adapter is *compositeType* while the output and input associated with composite adapter and order portlet is *CompositeType2*. This prevents multiple menu action selections appearing on the C2A menu drop of the original source portlet (causing mass confusion on the user).

The following is the declarative code for the WSDL;

```
?xml version="1.0" encoding="UTF-8"?>
<definitions name="CompositeportletadapterPortlet
_Service"
  targetNamespace="http://www.ibm.com/wps/c2a/exam-
ples/shipping"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:portlet="http://www.ibm.com/wps/c2a"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/
soap/"
  xmlns:tns="http://www.ibm.com/wps/c2a/examples/
shipping"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <types>
    <xsd:schema
      targetNamespace="http://www.ibm.com/wps/c2a/
examples/shipping">
      <xsd:simpleType name="compositeType">
        <xsd:restriction base="xsd:string" />
      </xsd:simpleType>
      <xsd:simpleType name="compositeType2">
        <xsd:restriction base="xsd:string" />
      </xsd:simpleType>
    </xsd:schema>
  </types>

  <message name="Compositeportletadapterportlet_
Request">
    <part name="Compositeportletadapterportlet_
RequestPart"
      type="tns:compositeType" />
  </message>

  <message name="Compositeportletadapterportlet_
Response">
    <part name="Compositeportletadapterportlet_
ResponsePart"
      type="tns:compositeType2" />
  </message>

  <portType name="Compositeportletadapterportlet_
Service">
    <operation name="new_customer_ID">
      <input message="tns:Compositeportletadapterpo
rtlet_Request" />
      <output
        message="tns:Compositeportletadapterportlet
_Response" />
    </operation>
  </portType>

  <binding name="Compositeportletadapterportlet_
Binding"
    type="tns:Compositeportletadapterportlet_
Service">
    <portlet:binding />
    <operation name="new_customer_ID">
      <portlet:action name="map_customer_id"
        caption="map.customer.id" type="simple">
      </portlet:action>
      <input>
        <portlet:param name="customerId"
          partname="Compositeportletadapterportlet_
RequestPart"
          caption="customer.id" description="input.
customer.id" />
      </input>
      <output>
        <portlet:param name="customerIdOutput"
          partname="Compositeportletadapterportlet_
ResponsePart"
          description="input.customer.id"
boundTo="session" />
      </output>
    </operation>
  </binding>

</definitions>
```

Now what is interesting here is that the output of the source portlet and the input of the target portlet do not have to match. The property broker will do the appropriate transformation. The output of the composite adapter will have to be either a request attribute or a session attribute since you can only set a request parameter indirectly.

Now you can pass an object instead of a string by specifying the class object using the class attribute on the portlet:param entry in WSDL file or the equivalent programmatic API.

However we would not recommend passing a general object due to the overhead and performance impact. To pass a parameter as an object, the object must be serializable and the source and target portlet must be able to load the objects using the same class loaders.

Since the object is not a string, the request parameter option can be used as a binding option. The C2A JSP tags can only transmits their value as string so defining the output associated with these tags as objects, well, doesn't make sense.

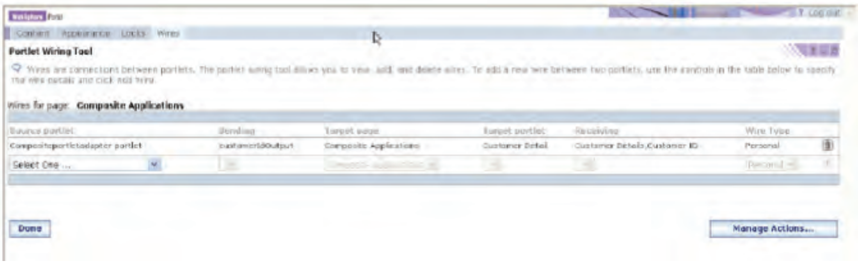
So now let us look at the code, specifically the actionPerformed method that gets triggered when an input with the same type is received. It is pretty simple.

```
public static final String CUSTOMER_ID = "custom-
```

```
erId";
public static final String CUSTOMER_IDOUTPUT =
"customerIdOutput";
public static final String MAP_CUSTOMER_ID =
"map_customer_id";
public static final String OLD_VALUE = "2867377";
public static final String NEW_VALUE = "3542877";
...

public void actionPerformed(ActionEvent event)
throws PortletException {
    String actionString = event.getAction-
String();
    PortletRequest request = event.getRequest();
    CompositeportletadapterPortletSessionBean
sessionBean = getSessionBean(request);
    if (MAP_CUSTOMER_ID.equals(actionString)) {
      String customerId = request.
getParameter(CUSTOMER_ID);
      if (customerId.equals(OLD_VALUE)) {
        request.getPortletSession().
setAttribute(CUSTOMER_IDOUTPUT,
                NEW_VALUE);
        sessionBean.setFormText(NEW_VALUE);
      } else {
        request.getPortletSession().
setAttribute(CUSTOMER_IDOUTPUT,
                customerId);
        sessionBean.setFormText(CUSTOMER_
IDOUTPUT);
      }
    }

    ...
  }
```

Basically, the method retrieves the customer id as a request parameter. If the customer id is 2867377, then it is reset to 34542877 otherwise we don't touch it. The output parameter is bound to a session attribute and set to the customer id value.

Now to make everything work, we place the portlets on a page. Click on the wire tab and define (as shown in figure 1) a wire from the Composite Adapter Portlet to the Custom Detail Portlet.



Portlet Wiring Tool

Wires are connections between portlets. The portlet wiring tool allows you to view, add, and delete wires. To add a new wire between two portlets, use the controls in the table below to specify the wire details and click Add Wire.

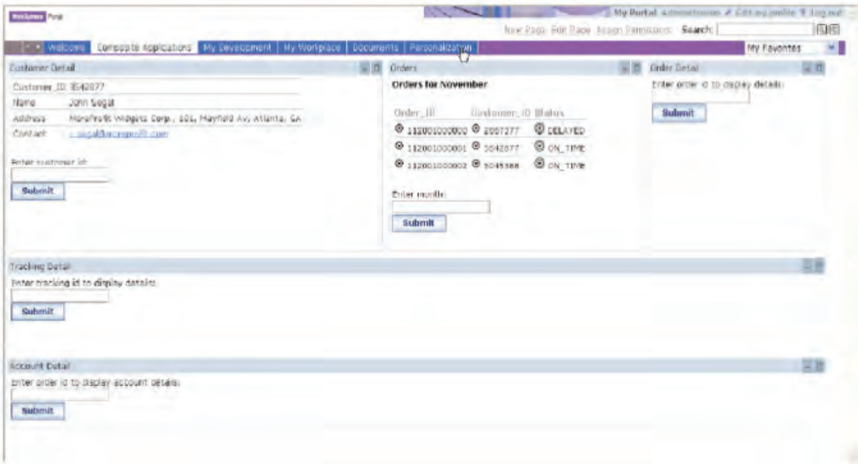
Wires for page: Composite Applications

Source portlet	Sending	Target page	Target portlet	Receiving	Wire Type
Compositeportletadapter portlet	customerIdOutput	Composite Applications	Customer Detail	Customer Details,Customer ID	Personal
Select One ...					Personal

Figure 1: Wiring the Adapter Portlet to the Customer Detail Portlet

Next add the Composite Portlet Adapter to the page and assign **NoSkin** to it. Figure 2 shows the result. When you click on customer 2867377, customer detail for 34542877 will appear.

The composite portlet adapter was created using declarative cooperative portlets because they are less intrusive. You can change the input and output properties with out touching the source. Most of your changes require access to the WSDL file and rarely to the JSP file. However we did can also be done programmatically with the property service broker APIs.



Customer Detail

Customer_ID 3542877
Name John Segal
Address Nonprofit Widgets Corp., 101, Mayfield Av, Atlanta, GA
Contact

Enter customer id:

Orders

Orders for November

Order_ID	Customer_ID	Status
◉ 112001000000	◉ 2867377	◉ DELAYED
◉ 112001000001	◉ 3542877	◉ ON_TIME
◉ 112001000002	◉ 3045388	◉ ON_TIME

Enter month:

Order Detail

Enter order id to display details:

Tracking Detail

Enter tracking id to display details:

Account Detail

Enter order id to display account details:

Figure 2: Composite Portlet Adapter in action

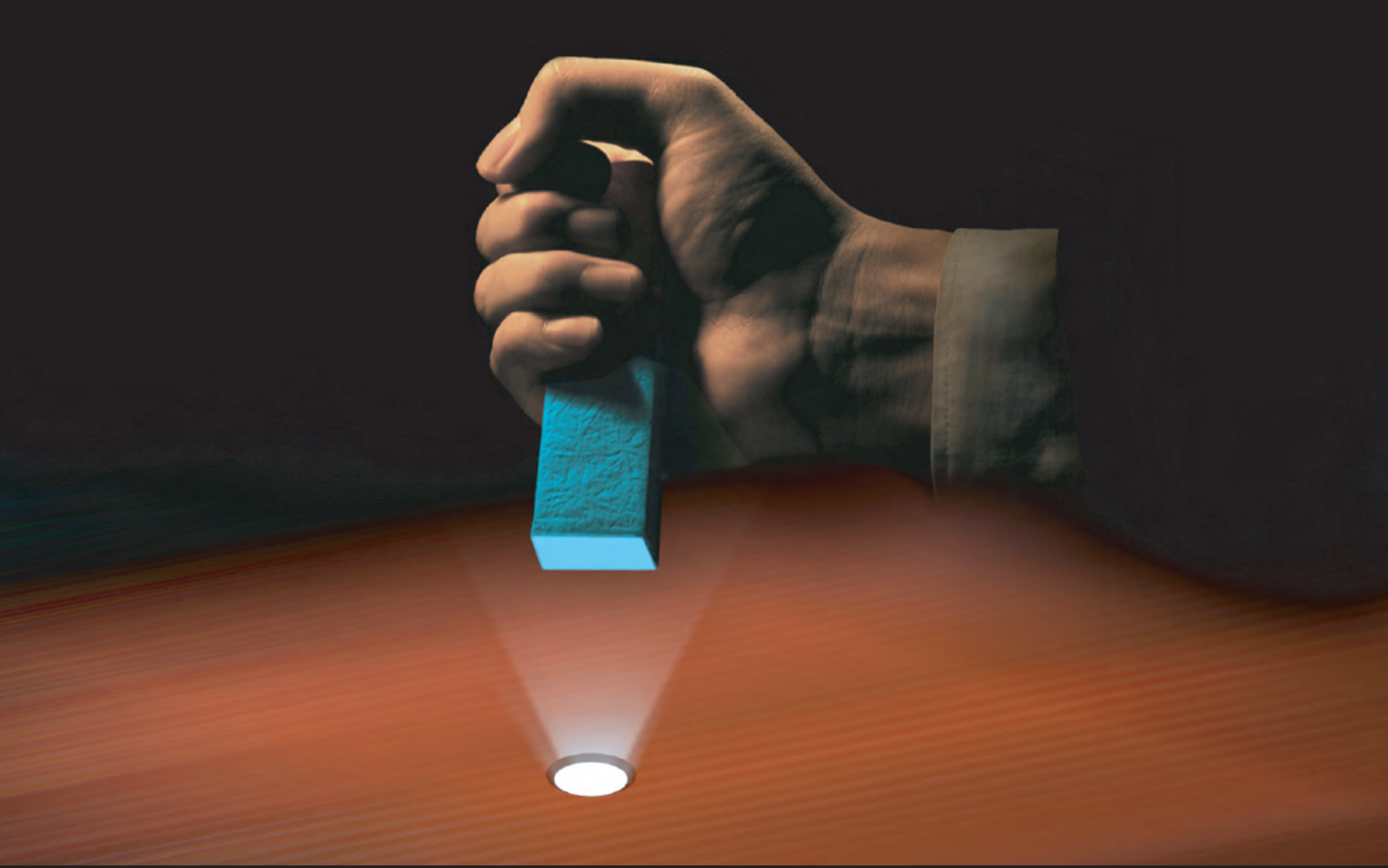
Lists and Lookup Maps

Now let us look at a more complex example. One of the most common patterns our customers implement is to associate objects from dissimilar systems. This can easily be done with composite applications.

Let's take our shipping example. Instead of wanting more information on the customer, we now want to see the purchase order associated with the order id. The purchase order can be on an ERP system or just a PDF file stored in a file system or database.

To implement this function, we create a helper portlet that has the action "Show Related Documents" associated with the purchase order number parameter. When the user clicks on the order id in the order parameter, the value will be sent to the target portlet, our helper portlet, which will use the P.O. number as a key and search the database for a pointer. This pointer can be link to a URL which contains the document or a key to access the document on a ERP system. The helper portlet will then send key/URL as an output parameter to the document library portlet on the next page. This implies that you have created a cross-page wire from the helper portlet to the document library portlet. The document library portlet will retrieve the document. If it is on a database, it will use a key, if it is on an ERP, it will be done through a mediator, and if it is in a file system, it will be accessed by a URL.

By wait a minute! You just realized that we had a portlet send a message to another portlet cross-page. Everybody knows that portals can't do that. Well, in WebSphere Portal 5.1.0.1+, it can be done.



It can easily be implemented using the Portlet Wiring Tool or programmatically. Just make sure that all cooperative portlets using this feature are declared Global. This feature is also great tool to use if you need to emulate a servlet application using portlets.

Click-To-Action

Many people wrongly feel that when they see a click-to-action application, they are restricted to using the C2A icon or C2A menu appearance. But actually, WebSphere Portal provides tremendous flexibility for C2A applications.

The C2A icon and menu can be controlled from the theme. The icon image, **c2a.gif, c2awired.gif, c2amultiple.gif**, and **c2awiredmultiple.gif,** can all be replaced with icon images of your choice so long as they occupy similar real-estate. They can be found in the themes/html directory.

Menu styles for the C2A portlet can also be modified. They are found in the default style sheet in **themes/html/Styles.css**. The following are some of the style rules available for your modification:

```
.portlet-menu {
   background-color: #EEEEEE;
   border-style: solid;
   border-color: #333333;
   border-width: 1px;
    padding: 3px;
}
.portlet-menu-item {…
}
.portlet-menu-item-selected {…
}…
```

Additional C2A modification can be done through the internal interface *ActionTriggerMarkupExtended* found in the package **com.ibm.wps.pb.property**. It provides two methods *getOnClickMarkup* and *getShowActionsMarkup* that provide some cool customization features. *getShowActionsMarkup* returns the markup for displaying the actions matching the source and the code for the chosen trigger action. The execution of the code is triggered through clicking. The method *getOnClickMarkup* returns the markup associated with the C2A attribute.

So these methods enable users to integrate C2A actions with actions derived from an external source or to implement filtering/augmenting of the action list that is associated with the C2A property. You can basically construct a custom C2A user interface paradigm instead of using a menu but you have to be careful because other portlets may not support your user interface paradigm.

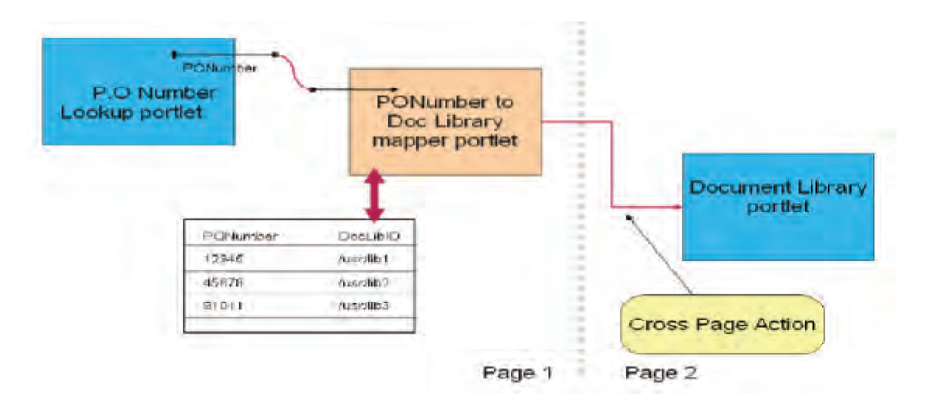
Portlet Menus

One of the biggest misconceptions of WebSphere Portal, is that menu entries can only be added to the navigation tree by adding pages. However a portlet based on the IBM portlet API can add menu entries to the navigation using either a dynamic or static method.

The static method allows portlets to declare menu entries in the deployment descriptor, **portlet.xml**. These can later be changed via the dynamic method.

In the deployment descriptor you add a configuration parameter that specifies the file path of the menu tree descriptor relative to the /WEB-INF folder. For instance, in the following example menu.xml is the file with the tree descriptor.

```
<config-param>
       <param-name>XMIFilePath</param-name>
       <param-value>menu.xml</param-value>
 </config-param>
```



PONumber	DocLibID
12346	/usr/lib1
45678	/usr/lib2
91011	/usr/lib3

Figure 3: Our helper portlet

In menu.xml, you define which nodes you will be populating. Then in your portlet add the appropriate import statements, change your class declaration and extend *XMIMenuTreePortlet*, and ensure that the method *actionPerformed() of the ActionListener* interface is implemented.

The dynamic method allows a portlet developer to add portlet menus dynamically. At each request, the portlet is queried for menu information so that the portlet can return stateful information in the menu entry.

For dynamic the method, you are basically overwriting the method *getMenuTree* of the superclass. You call *createEmptyMenuTree()* which uses the *MemoryMenuService* to create the tree. When the portlet is selected for the first time, the menu tree is created and stored in a session attribute. This will obviously impact your session size and can possibly impact performance. You create your controllers and then through your code add nodes and associate actions with them.

An excellent explanation on how to do static and dynamic menus is in the WebSphere Portal Javadoc.

Summary

In this article we tried to stimulate your imagination and show you the power of Composite application and WebSphere Portal. WebSphere Portal provides the developer with functionality to provide very powerful, modular, and flexible applications. You can create portlets which are loosely coupled but can dynamically manipulate information as filters, adapters, lookup helpers etc. The power of the applications is only limited by the creativity of the developer.

The authors would like to thank Amber Roy-Chowdry, the Master of Co-operative Portlets, for his assistance in this article.

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

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

24/7

FREE NEWSLETTERS

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

SYS-CON.TV

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

BLOG-N-PLAY!

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

WEBCAST

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

EDUCATION

The world's leading online i-Technology university

RESEARCH

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

MAGAZINES

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

INTERNATIONAL SITES

Get all the news and information happening in other countries worldwide

JUMP TO THE LEADING i-TECHNOLOGY WEBSITES:

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

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

Part One of a Two-Part Article

Building SOA Solutions with Service Component Architecture

BY ROLAND BARCIA AND JEFF BRENT

IntroductionYou may be thinking, oh great, not another programming model. What about Web services? What happened to Enterprise JavaBeans? Well, Service Component Architecture (SCA) is not meant to replace or compete with any existing programming models. Instead, SCA gives you a model to define interfaces, implementations, and references in a technology neutral way, letting you then bind these elements to whichever technology specific implementations you choose.

ABOUT THE AUTHORS

Roland Barcia is a Consulting IT Specialist for IBM Software Services for WebSphere in the New York/New Jersey Metro area. He is a co-author of IBM WebSphere: Deployment and Advanced Configuration..

Jeff Brent is an Advisory Software Engineer for the WebSphere Business Integration Competency Center (WBICC). His responsibilities include helping independent software vendors develop integration strategies for WebSphere products.

For example, we can define our interface in Java, and our implementation can be applied as a BPEL process, or we can have my interface as a WSDL document and our implementation could be a Java™ class. Figure 1 illustrates how SCA is used in IBM WebSphere Process Server. Out of the box, WebSphere Process Server lets integration developers have various different implementation technologies and enables you to expose the interfaces using open standards, such as with Java or WSDL. Additionally, SCA provides a client programming model that lets clients access service components. Developers are encouraged to keep developing using the various technologies they are familiar with, such as Enterprise JavaBean (EJBs), and use SCA to glue the components together.

Another major element of SCA is that it also defines a standard model for defining dependencies between components. As such, dependencies are defined by wiring SCA components together using references.

Finally, SCA defines a standard deployment model for packaging components into a service module. SCA components with their associated dependencies can be defined and packaged together into deployable units.

An SCA module is not just another type of package. In WebSphere Process Server, an SCA **service module** is equivalent to a J2EE EAR file and several other J2EE submodules. J2EE elements, such as a WAR file, can be packaged along with the SCA module. Non-SCA artifacts (JSPs, and others) can also be packaged together with an SCA service module, enabling them to invoke SCA services through the SCA client programming model using a special type of reference called a **standalone reference**.

SCA is closely tied to integration; we described references above as the way of defining dependencies between SCA components. For example, we can define a reference to another SCA component within the same module. When invoking service components within a module using the reference, the data is passed by-reference. SCA defines a way for components to be invoked by, or to invoke other SCA services, that live in other SCA modules.The mechanisms used for module to module and module to external service invocation are called **imports** and **exports**.

Imports and exports are expressed from the perspective of the module. The **module** is a self-contained bundle of components that perform a specific business function.

When the module wishes to provide the ability for another entity (external service or other module) to invoke a business function it **exports** this capability. Exports also provide the ability to make this service available over a number of different transport protocols. The export is associated to a particular component within the module.

When the module wishes to leverage the ability of another entity (external service or module) the module will **import** this function. Imports also provide the ability to interact with service providers across a variety of transport protocols.

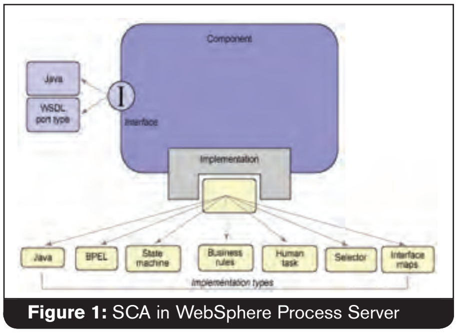


Figure 1: SCA in WebSphere Process Server

(This article was first published on developerWorks WebSphere at http://www.ibm.com/developerWorks/websphere.)

Figure 5 illustrates these concepts.

Imports and exports are abstract concepts. They need to be binded to a specific technology. The types of bindings provided in WebSphere Process Server V6.0 are:

- SCA (used for SCA module to module)
- Web service
- JMS
- Stateless session bean.

When invoking SCA components through imports and exports, parameters are passed by-value.

The wires that glue SCA components together abstract out most of the infrastructure level code. For example, we can define wires to be synchronous or asynchronous, operations to be one way or request-response, or we can define the transactional behavior between components. SCA handles the infrastructure details for you. We will elaborate more on these details throughout this article series.

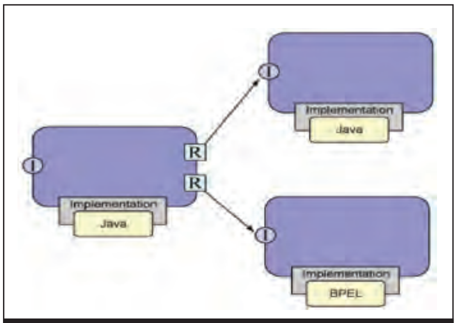


Figure 2: Define dependencies between components

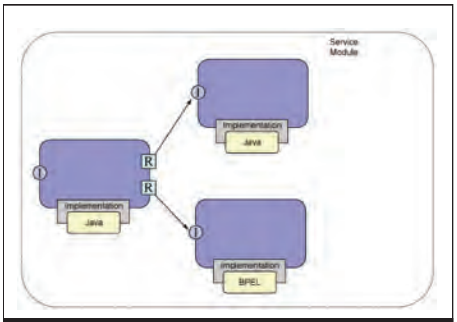


Figure 3: Components packaged with dependencies

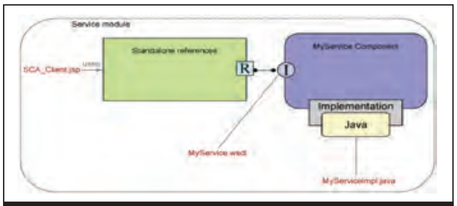


Figure 4: Standalone reference for non-SAC artifacts

Imports and exports can also bound to other technologies such as JMS, Enterprise JavaBeans, or Web services. This enables a Web service client to invoke an SCA module, or an SCA module to invoke an existing Enterprise JavaBean using the SCA programming model.

We will discuss imports and exports in an article later in this series. What about my data?SCA gives us a universal model to define business services. The **Service Data Object (SDO)** provides the technology to represent a universal model for data. SCA components can be composed and can exchange data with each other in a neutral fashion by passing SDOs. The fundamental concept in the SDO architecture is the **data object**, a data structure that holds primitive typed data and/or other data objects. The data object also holds references to metadata that provides information about the data included in the data object.

In the SDO programming model, data objects are represented by the commonj.sdo.DataObject Java interface definition. This interface includes method definitions that enable clients to get and set the properties associated with the DataObject. Another important concept in the SDO architecture is the **data graph**, a structure that encapsulates a set of data objects. From the top level data object contained in the graph, all children data objects are reachable by traversing the references from the root data object. Another important feature included in the data graph is a **change summary** that is used to log information about what data objects and properties in the graph have changed during processing. (See Resources for more details on SDO.)

WebSphere Process Server implements the SDO specification by way of **business objects**. SCA components can exchange data by passing around business objects as shown in Figure 8. Much like an SDO is wrapped in a DataGraph, a **business graph** is used to wrap a top level business object and provide additional information that is used to supplement the data that is included the graph. In particular, the business graph includes the change summary for the data in the graph (similar to the SDO change summary information), event summary, and verb information (used for data synchronization between EIS systems). The business graph is very similar to the concept of the data graph in the SDO

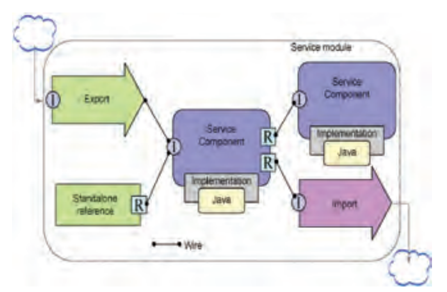


Figure 5: SCA imports and exports

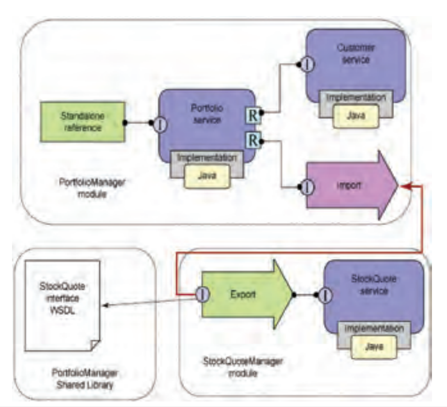


Figure 6: SCA imports and exports

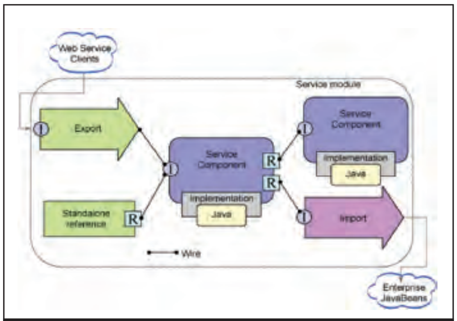


Figure 7: Imports and exports with other technologies

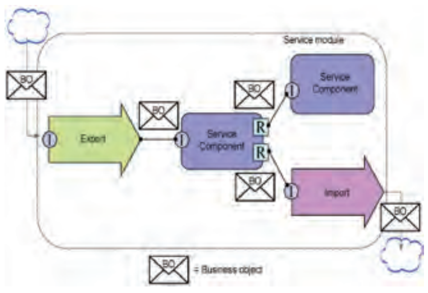


Figure 8: WebSphere Process Server business objects

architecture. However, the event summary and the verb portion of the enhanced information were not included with the SDO data graph concept.

SCA 101Now that we've looked at SCA from a 10,000 foot view, we will begin to discuss some of the details using a sample. During the course of building the sample, we will give you an overview of IBM WebSphere Integration Developer, a tool that you will use to visually build and integrate SCA components.

OverviewTo demonstrate the aspects of SCA and business objects, we will discuss these concepts related to a business case. As with all development efforts, the cycle must begin with requirements. The requirements in this scenario involve the creation of a credit approval service, which will receive information about an applicant (CreditApplication) and respond with a credit rating (CreditRating).

The simplest element in SCA is a **service component**. As mentioned earlier, a service component is made up of an interface and an implementation. The technology used to "code" these artifacts can vary; in WebSphere Process Server, interfaces can be Java or WSDL. A Java interface can be a Plain Old Java Interface (POJI) as shown below:

Listing 1:

```
public interface CreditRequest
{ public DataObject calulateCreditScore(DataObject
creditApp)
       throws ServiceBusinessException;

}
```

If you use a Java implementation, you can create a simple POJO. Below is an example of a POJO that acts as an SCA implementation:

Listing 2:

```
public class CreditApprovalImpl implements
CreditRequest {
          public DataObject calulateCreditScore(Da
taObject creditApp) {

          ServiceManager serviceManager = new
ServiceManager();

          BOFactory bof = (BOFactory)serviceManag
er.locateService("com/ibm/websphere/bo/BOFactory");
          DataObject creditRating = bof.
create("http://CreditApproval", "CreditRating");
                              creditRating.
setString("customerId", creditApp.getString("custom
erId"));

          creditRating.setInt("creditScore", 750);
          creditRating.setDouble("creditLimit",
10000d);

          return creditRating;
  }

}
```

In this implementation class, we use the CreditApplication input to create a simple CreditRating business object that we then return to the caller.

A service component is defined in a **Service Component Definition Language (SCDL)** file. A component file created with SCDL is roughly analogous to an EJB deployment descriptor, in that it defines the interface, implementation, and several qualities of service requirements of an SCA component. The SCDL file for the Java classes above would look like this:

Listing 3:

```
<?xml version="1.0" encoding="UTF-8"?>
<scdl:component xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
xmlns:java="http://www.ibm.com/xmlns/prod/web-
sphere/scdl/java/6.0.0"
xmlns:ns1="http://CreditApproval/CreditRequest"
xmlns:scdl="http://www.ibm.com/xmlns/prod/web-
sphere/scdl/6.0.0"
xmlns:wsdl="http://www.ibm.com/xmlns/prod/web-
sphere/scdl/wsdl/6.0.0"
displayName="CreditApproval" name="CreditApproval">
  <interfaces>
    <interface xsi:type="java:JavaInterface"
interface="CreditRequest " >
      <method name="calulateCreditScore"/>
    </interface>
  </interfaces>
  <implementation xsi:type="java:
JavaImplementation" class=" CreditApprovalImpl"/>
</scdl:component>
```

Similarly, we could have used WSDL to represent the interface as shown here:

Listing 4:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:bons1="http://
CreditApproval"
xmlns:tns="http://CreditApproval/CreditRequest"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
name="CreditRequest"
targetNamespace="http://CreditApproval/
CreditRequest">
  <wsdl:types>
    <xsd:schema targetNamespace="http://
CreditApproval/CreditRequest"
   xmlns:bons1="http://CreditApproval"
   xmlns:tns="http://CreditApproval/CreditRequest"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:import namespace="http://
CreditApproval"
schemaLocation="xsd-includes/http.CreditApproval.
xsd"/>
      <xsd:element name="calulateCreditScore">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="CreditApp"
nillable="true"
type="bons1:CreditApp"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="calulateCreditScoreResp
onse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="CreditRating"
nillable="true"
type="bons1:CreditRating"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
    <wsdl:message name="calulateCreditScoreReque
stMsg">
    <wsdl:part element="tns:calulateCreditScore"
name="calulateCreditScoreParameters"/>
  </wsdl:message>
    <wsdl:message name="calulateCreditScoreRespo
nseMsg">
    <wsdl:part element="tns:calulateCreditScoreRe-
sponse"
name="calulateCreditScoreResult"/>
  </wsdl:message>
    <wsdl:portType name="CreditRequest">
    <wsdl:operation name="calulateCreditScore">
      <wsdl:input message="tns:calulateCred-
itScoreRequestMsg"
name="calulateCreditScoreRequest"/>
      <wsdl:output message="tns:calulateCred-
itScoreResponseMsg"
name="calulateCreditScoreResponse"/>
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

The implementation class could look the same (except it would obviously not implement any Java interface since it is defined as WSDL).

Our SCDL file would bind the interface to a WSDL document:

Listing 5:

```
<?xml version="1.0" encoding="UTF-8"?>
<scdl:component xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
xmlns:java="http://www.ibm.com/xmlns/prod/web-
sphere/scdl/java/6.0.0"
xmlns:ns1="http://CreditApproval/CreditRequest"
xmlns:scdl="http://www.ibm.com/xmlns/prod/web-
sphere/scdl/6.0.0"
xmlns:wsdl="http://www.ibm.com/xmlns/prod/web-
sphere/scdl/wsdl/6.0.0"
displayName="CreditApproval" name="CreditApproval">
  <interfaces>
    <interface xsi:type="wsdl:WSDLPortType"
portType="ns1:CreditRequest">
      <method name="calulateCreditScore"/>
    </interface>
  </interfaces>
  <implementation xsi:type="java:
JavaImplementation" class="CreditApprovalImpl"/>
</scdl:component>
```

SCDL is used to describe SCA elements such as modules, references, imports, and exports. You will see examples of these throughout the series. SCDL definitions are organized across several files. For example, we store the SCDL for the interface and implementation in a file called CreditApproval.component. References can be included in the CreditApproval.component file (in-line) or in a separate sca.references file located in the Module root. Any standalone reference will be placed in the sca.references file, shown below. As we mentioned, standalone references can be used by non-SCA artifacts (JSP) within the same SCA module to invoke the SCA component.

Listing 6:

```
<?xml version="1.0" encoding="UTF-8"?>
<scdl:references xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
```

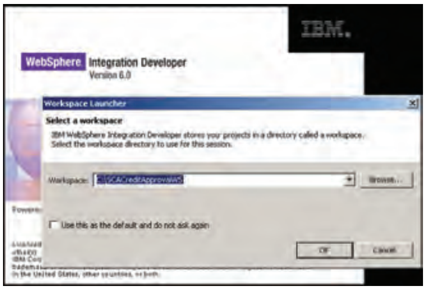
WebSphere JOURNAL **Advertiser Index...**

COMPANY	URL	PHONE	PG
AJAX Seminar	http://www.ajaxseminar.com/	201-802-3022	39
IBM	www.ibm.com/Middleware/apps	800-426-4968	52
Instantiations	www.instantiations.com/analytix	800-808-3737	2
ISSJ	http://issj.sys-con.com/	888-303-5282	40
IT Solutions guide	http://itsolutions.sys-con.com/	201-802-3021	35
ITKO	www.itko.com	877-BUY-ITKO	51
Java developer's journal	http://jdj.sys-con.com/		47
Jinfonet	www.jinfonet.com	301-838-5560	17
Motive	www.motive.com/within1	512-531-2527	9
Nastel	www.nastel.com/uncomplicate	800-580-2344	13
Prolifics	www.prolifics.com	800-675-5419	3
sys-con media website	www.sys-con.com	888-303-5282	19
Tangosol	www.tangosol.com	617-623-5782	11
webservices edge/open source	www2.sys-con.com/events	201-802-3066	19
websphere journal	www.sys-con.com	888-303-5282	47
Wily	www.wilytech.com	888-GET-WILY	5

WebSphere JOURNAL **Coming Next Month...**

Advance Authentication in WAS

Building SOA Solutions with Service Oriented Components Architecture



IBM.

WebSphere Integration Developer
Version 6.0

Workspace Launcher

Select a workspace

IBM WebSphere Integration Developer stores your projects in a directory called a workspace.
Select the workspace directory to use for this session.

Workspace: C:\SCACreditApprovalWS [Browse...]

☐ Use this as the default and do not ask again

[OK] [Cancel]

in the United States, other countries, or both.

Figure 9: Open a blank workspace in WebSphere Integration Develope

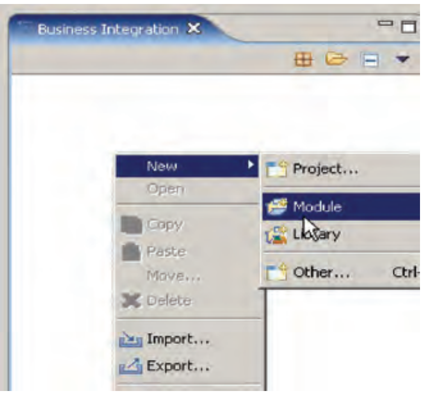


Figure 10: Create new module

```
xmlns:java="http://www.ibm.com/xmlns/prod/web-
sphere/scdl/java/6.0.0"
xmlns:scdl="http://www.ibm.com/xmlns/prod/web-
sphere/scdl/6.0.0">
  <reference name="CreditRequestPartner">
    <interface xsi:type="java:JavaInterface"
interface="approval.credit.credit.request.
CreditRequest">
      <method name="calulateCreditScore"/>
    </interface>
    <wire target="CreditApproval"/>
  </reference>
</scdl:references>
```

In our example, a third file we will define is the sca.module. This SCDL file will contain the definition for the module:

Listing 7:

```
<?xml version="1.0" encoding="UTF-8"?>
<scdl:module xmlns:scdl="http://www.ibm.com/xmlns/
prod/websphere/scdl/6.0.0" name="CreditApproval"/>
```

Although we defined a WSDL interface for our component, notice that we are able to add another Java interface to the standalone reference and wire it to our target component. In this manner, Java clients can invoke SCA components using a simple POJI. WebSphere Integration Developer will provide the option to translate the call between the Java and WSDL interface. You will see an example of this shortly when you create your first SCA component.

We will see examples of how SCDL defines other SCA artifacts, such as service references, imports, and exports, throughout this article series. The naming conventions for the SCDL files above are used by WebSphere Integration Developer, and were actually by using WebSphere Integration Developer. In our example, you will use WebSphere Integration Developer when building your first SCA component. However, SCA applications can also be built using a notepad process as well.

WebSphere Integration Developer and WebSphere Process ServerIn this article, you will use WebSphere Integration Developer and WebSphere Process Server to build and run your SCA components.

WebSphere Integration Developer is used to develop applications that run on WebSphere Process Server V6. WebSphere Integration Developer, based on the Eclipse 3.0 platform, delivers role-based development for integration projects. You can use WebSphere Integration Developer and IBM WebSphere Business Modeler V6 together with IBM Rational® Application Developer V6 or IBM Rational Software Architect V6 as an integration development platform. J2EE developers, business analysts, or integration developers can use the tooling perspective based on their unique roles, so that each developer can focus on the editors and tools needed for those roles, thereby maximizing productivity.

WebSphere Process Server is a comprehensive integration platform which is based on WebSphere Application Server V6. WebSphere Process Server is used to execute component-based business integration applications in a service-oriented architecture. Because it is based on J2EE 1.4 infrastructure and platform services provided by WebSphere Application Server, WebSphere Process Server includes capabilities such as business process automation.

See Resources for more details on both WebSphere Integration Developer and WebSphere Process Server.

Build your first SCA projectYou are now ready to build your first SCA project. In this example, you will build a simple credit approval component. The component will take in a data object with an ID and name, and then return another data object containing a credit score and credit limit. Download the zip file included with this article and extract it to your hard drive. The instructions assume you extract the zip file to your C: drive.

WebSphere Integration Developer V6.0 is a flexible development environment that provides tools and wizards for developing SCA applications from the bottom-up or from the top-down.

For the creation of the credit approval service, we will explore the aspects of top-down development. The general high level steps in our top-down development process are:

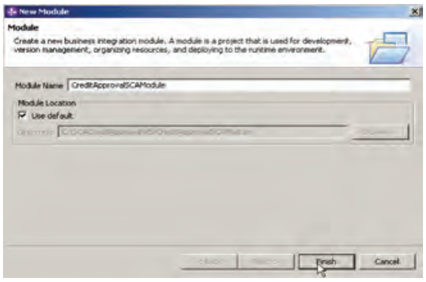
1. Create the SCA module.
2. Create the business objects.
3. Define the service interface.
4. Generate the component and provide an implementation.
5. Unit test the SCA component.
6. Provide a standalone reference.
7. Test the service using a simple JSP client.

Create an SCA moduleFirst, you need to create an SCA module. As stated earlier, an SCA module is the packaging construct of SCA components.

1. Open WebSphere Integration Developer to a blank workspace. (Figure 9)
2. Close the Welcome screen.
3. Next you will create a new module.
   a. Your WebSphere Integration Developer workbench should be open to the Business Integration perspective. Find the Business Integration view.
   b. Right-click inside the Business Integration view and select New => Module. (Figure 10)
   c. The New Module wizard should

display (Figure 11). Name the Module CreditApprovalSCAModule.

d. You should now have a new project displaying in the Business Integration view. SCA modules are described as a SCDL file. The Business Integration view shows you a logical view of your SCA module. You can open the Physical Resource view to see the physical resources contained in the SCA module, as previously discussed. The SCDL for the SCA module is contained in the file called sca.module. If you open the file with a text editor, you should see something like this:


New Module

Module

Create a new business integration module. A module is a project that is used for development, version management, organizing resources, and deploying to the runtime environment.

Module Name: CreditApprovalSCAModule

Module Location

☑ Use default

Figure 11: Create new module

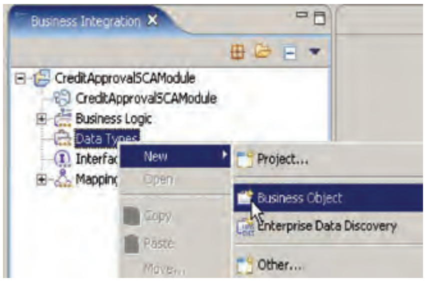
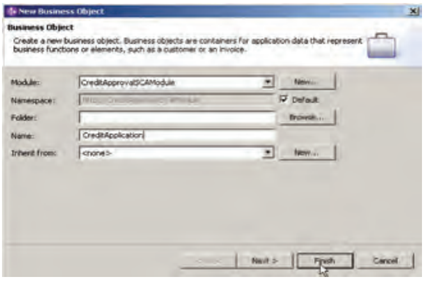


Figure 12: Create a new business object


New Business Object

Business Object

Create a new business object. Business objects are containers for application data that represent business functions or elements, such as a customer or an invoice.

Module: CreditApprovalSCAModule

Namespace:

☑ Default

Folder:

Name: CreditApplication

Inherit from: <none>

Figure 13: Define business object

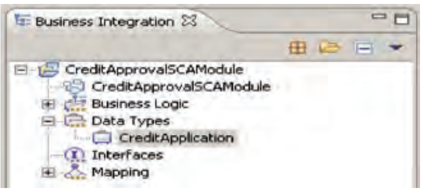


Figure 14: Business Integration menu

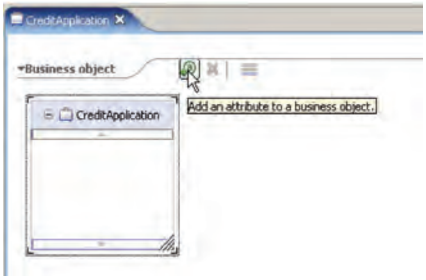
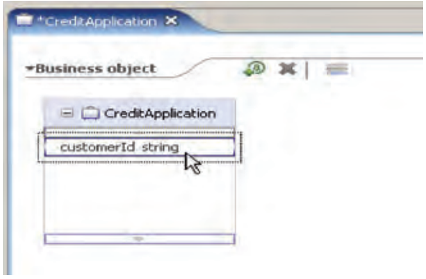


Figure 15: Add Attribute icon



customerId string

Figure 16: Define attribute

Listing 8:

```
<?xml version="1.0" encoding="UTF-8"?>
<scdl:module xmlns:scdl="http://www.ibm.com/
xmlns/prod/websphere/scdl/6.0.0" name="CreditApp
rovalSCAModule"/>
```

Create the business objects. There is no predefined sequence of tasks for creating interfaces or business objects; you could have just as easily have created the business objects when creating the interface. Here we chose to create business objects from the Business Integration view.

When creating a business object, you are actually creating an XML schema. Business objects are stored as standard XML schema. Applications can access the data using the SDO programming model and XPath.

The Business Object Editor enables us to create the business object definitions that will be used by the credit approval service. There are two business objects that are needed for our service.

CreditApplication: Contains the information about the credit applicant necessary to calculate a credit rating. The credit application business object (CreditApplication) consists of three attributes:

- **Customer Id:** unique identifier of the customer
- **First Name:** customer's first name
- **Last Name:** customer's last name

CreditRating: Contains credit rating information such as credit score and credit limit. The credit rating business object (CreditRating) consists of three attributes:

- Credit Score: customer's credit score based on prior history
- Credit Limit: total amount that the customer may owe.

1. We begin by creating our Request business object, which will be used as an input parameter to our SCA component.
   a. Make sure that the CreditApprovalSCAModule is expanded. Right click on Data Types and select New => Business Object. (Figure 12)
   b. Name your business object CreditApplication as shown in Figure 13.
   c. The business object should open in the Business Object Editor. You should now have the CreditApplication under the Data Types menu. (Figure 14)

2. We will now define the attributes of the CreditApplication business object.
   a. You can add an attribute by selecting the Add Attribute icon as shown in Figure 15.
   b. Enter customerId as the attribute name and accept the default String type. (To change the type, you could just select the type field and select it from the drop-down box. You will do this for the response object.)

      Figure 16. Define attribute
   c. Add two more String attributes and name them firstName and lastName. Both of them should be of type string as shown below.

TO BE CONTINUED IN THE NEXT ISSUE

The obstacles to the adoption and successful realization of Service-Oriented Architecture

SOA Antipatterns

BY JENNY ANG, LUBA CHERBAKOV, DR. MAMDOUH IBRAHIM

Explore different Service-Oriented Architecture (SOA) antipatterns, which are descriptions of commonly occurring situations or solutions that generate decidedly negative consequences. With more businesses taking major steps to move from Web services to SOA, barriers to the introduction, adoption, and successful implementations of SOA are becoming more evident. Some of these barriers are similar to those that caused past essential initiatives to fail; others are specific to SOA.

ABOUT THE AUTHORS

Jenny Ang (jang@us.ibm.com), Executive IT Architect, IBM

Luba Cherbakov (lubacher@us.ibm.com), IBM Distinguished Engineer, IBM

Dr. Mamdouh Ibrahim (mibrahim@us.ibm.com), Senior Certified Executive IT Architect, IBM

Documenting such barriers and worst practices will help consultants, architects and specialists not to repeat the same mistakes and learn how to avoid them instead. The antipatterns compiled and described here were identified by the authors through personal experiences as IBM architects, examination of past and current SOA engagements, and by soliciting input from practitioners who were involved in customer SOA engagements.

Patterns versus antipatterns

"*Example isn't another way to teach*; it's the only way" – Albert Einstein

Patterns and pattern languages capture and formally codify good designs and best experience-based practices in a way that it is possible for others to reuse them. They successfully convey insight into common problems and their solutions. After all, common concepts, a vocabulary to describe them and a language to connect them together are the underpinnings for all disciplines and communities that practice them.

Christopher Alexander's research on buildings and town design is often considered the pioneering work on pattern-based thinking (*see Resources*). He coined the term "Pattern Language" to express his conviction that people's ability to design is as natural as their ability to use a language.

Patterns and pattern languages have been used by many disciplines, ranging from physiology and processes to project management and software engineering. Software design patterns became well accepted and used after the publication of the book *Design Patterns: Elements of Reusable Object-Oriented Software* by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides (frequently referred to as the Gang of Four).

The software community is using patterns to resolve recurring problems encountered throughout the software lifecycle, ranging from software architecture and design to, more recently, software development processes and topologies. These patterns collectively capture the body of knowledge that represents our understanding of structures and mechanisms leading to well-architected software solutions.

A pattern is often defined as a "generalized, named problem-to-solution mapping." It captures a successful solution to a repeating problem in a particular context. Often, software patterns are documented using a template similar to one depicted in Table 1, Pattern template.

Software patterns provide a mechanism to capture knowledge and experience among architects and designers. They provide a common language and facilitate reuse of approaches that have been successful elsewhere and, thus, contribute towards the following aspects of a software project: reduced risk, increased quality and improved delivery time.

Antipatterns, on the other hand, document things that went wrong. Various surveys of hundreds of software development projects undeniably illustrate that things can – and often do – go wrong with software development. Studies show that five out of six projects fail: delivered far over the expected budget, significantly late, or are canceled. This suggests that it might be (at least) as worthwhile to examine causes for frequent failures as the rare instances of success (Noted author of Bitter Java, Bruce Tate, demonstrated in his developerWorks article how and why antipatterns are a necessary and complementary companion to design patterns (see the Resources section for more information).

These repeated failed software development projects or "negative solutions" should be mined to harvest useful knowledge of "what went wrong and why." Obviously, just categorizing the causes of failure is not as useful as also examining how to avoid them and how to recover when one is encountered. When codified, this collective knowledge makes a valuable extension to software patterns and classified as antipatterns.

Antipattern is a frequently used, but largely ineffective solution to a problem. The term was originally used to refer to a design pattern gone wrong. Similarly to patterns, use of antipatterns has extended to all software development phases and beyond, to other domains. Antipatterns document commonly recurring solutions that have counterproductive effects. They typically capture refactored solution descriptions, showing how to change the antipattern into a healthier solution. Antipatterns are usually described in a template that identifies symptoms, consequences, root causes, and potential solutions. Although antipatterns are not as widely studied and written about as patterns, some of them, which have colorful monikers such as analysis-paralysis, blob, spaghetti code, and stovepipe systems, are readily recognizable by the software community. Table 2 provides an overview of some of these examples sourced from Brown et. al.'s antipatterns book (see the Resources section for more information). Why are antipatterns important?

Antipatterns are tools to prevent problems by helping to identify a problem before it becomes a problem, and by providing knowledge on how to prevent that from happening. Formal codification of failure causes allows us to intelligibly understand them. Once problems occur, antipatterns can help by explaining how to recover from them.

To summarize, antipatterns have the following elements:

- Documentation of what does not work
- A common vocabulary
- Detailed remedy
- Awareness of situation and alternative solutions
- Today's hot solution that can be tomorrow's antipattern

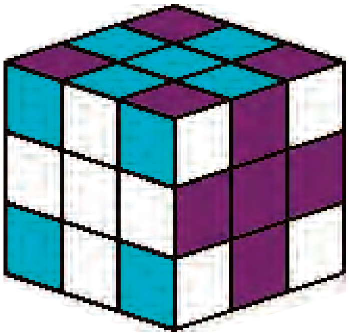


Figure 1 expresses the difference between patterns and antipatterns. A pattern starts with a problem that you are trying to solve and documents a repeatable successful solution to it. The solution generates some benefits, consequences and possible problems. An antipattern demonstrates a frequently used solution to a problem that has counterproductive effect. It describes the causes that led to it and also shows how to prevent or correct the solution.

SOA: brief introduction

In the last five years, much has been written about service-orientation, SOA and more recently, service-oriented everything. But what do we mean by services and service orientation?

There are many different definitions -- and the definitions have changed reflecting maturity of the industry and SOA practices. We provide here some base-line definitions to be used in this article. The definitions are reflected in Figure 2.

1. First of all, what is a service? A service is a repeatable logical manifestation of a business task. It is important to stress that we're talking about a part of a business process here -- not about software or IT.

   When realized through technology, the term service applies to a software resource (discoverable) with an externalized specification. This service specification is available for searching, binding, and invocation by a service consumer. The service provider realizes the service specification implementation and also delivers the quality of service requirements to the service consumer. Services shall be governed by declarative policies and thus support a dynamically re-configurable architectural style.
2. Second, what is service orientation? Building on our definition of a service, service orientation is a way of integrating a business as a group of linked services. We are still not talking about technology; we are talking about a new way of looking at business and how it operates.
3. What is SOA? SOA is an architectural style that supports the service orientation. SOA is an enterprise-scale IT architecture for linking resources on demand. These resources are represented as business-aligned services which can participate and be composed in a value-net, enterprise, or line of business to fulfill business needs.
4. And finally, what is a composite application? It's a set of integrated services. Composite applications are the actual running services that have been assembled and strung together to support what a business does. The primary structuring

Content	Description
Name:	A name used for identification
Problem:	A repeating problem that occurs in a domain
Solution:	Best practice solution to that problem
Consequences:	Advantages and disadvantages of the recommended solution
Examples	A few examples where the recommended solution has already been applied

Table 1: Pattern templateContent

Mistake category	Antipattern	Description
Design	Blob	A large class with too many attributes and is the "heart" of the system
Design	Poltergeists	Unnecessary classes and too much abstraction
Structure	Spaghetti code	Program code that has no structure (lots of goto statements)
Structure	Stovepipe systems	Applications are unique and isolated
Technology	Wolf ticket	A technology that claims openness, but has no conformance standard test
Technology	Continuous obsolescence	The attempt to use the latest release
Reuse	Cut-and-paste	Software errors are duplicated
Reuse	Golden hammer	Forcing everything to fit a selected tool

Table 2: JExamples of known antipatterns

element for SOA applications is a service as opposed to subsystems, systems, or components.

Approach to identifying SOA antipatterns

SOA antipatterns were identified using the following approach:

- Surveyed literature for published antipatterns
- Documented antipatterns discovered by the authors on engagements in client meetings
- Surveyed SOA CoE and CoP members for antipatterns identified from their experiences
- Used recognized antipatterns template/ language
- Antipatterns included in this presentation have been agreed upon by the three authors

Antipatterns template/language

The antipatterns are described using the following template / language:

- Name: A succinct name to convey the essence of the antipattern
- Problem / Bad solution: The commonly occurring mistake or bad solution that relates to the antipattern
- Symptoms : The indications or signs of the problem
- Consequences: The results of applying this antipattern
- Root cause: This provides the context for the antipattern, that is, where a pattern was applied incorrectly and resulting in a problem or failed solution
- Suggested solution(s): Refactored solution that solves the problem and ensures more benefits

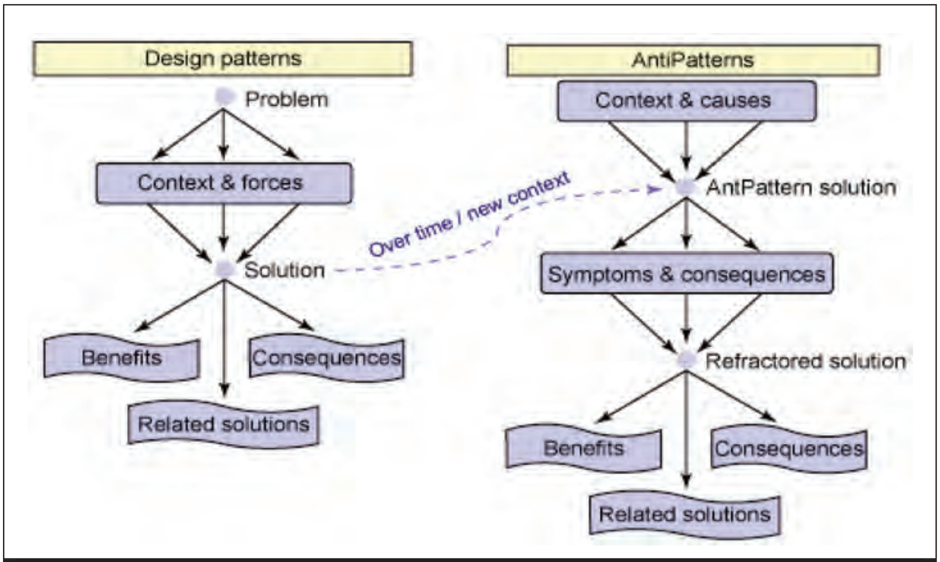
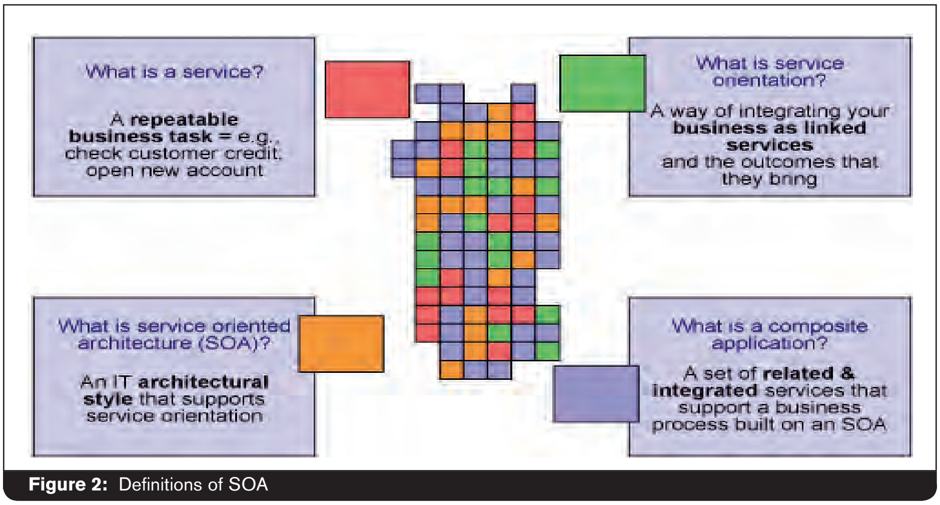


Figure 1: Patterns versus antipatterns (Source: Brown et. al., "AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis")



What is a service?

A **repeatable business task** = e.g., check customer credit; open new account

What is service orientation?

A way of integrating your **business as linked services** and the outcomes that they bring

What is service oriented architecture (SOA)?

An IT **architectural style** that supports service orientation

What is a composite application?

A set of **related & integrated** services that support a business process built on an SOA

Figure 2: Definitions of SOA

SOA antipatterns

The identified antipatterns are classified into three categories:

1. SOA Adoption antipatterns: These are antipatterns that hinder or delay SOA adoption by customers and businesses. Antipatterns discussed in this category are:
   - A1. Technology Bandwagon
   - A2. So, What's New?
   - A3. The Big Bang

2. Service identification & design antipatterns: These are antipatterns encountered while practitioners work on identifying and designing services as part of a SOA initiative. Antipatterns discussed in this category are:
   - I1. Web service = SOA
   - I2. The Silo Approach
   - I3. Misbehaving Registries

3. Service realization antipatterns: These antipatterns capture worst practices for realizing services. Many of these antipatterns are focused on Web services implementation; the most common realization of SOA. In this paper we identified a partial list of SOA realization antipatterns that are not focused on Web services since such antipatterns are better discussed in a forum devoted to Web services. Antipatterns discussed in this category are:
   - R1. Chatty Services
   - R2. Point-to-point Services
   - R3. Component-less Services

As SOA matures and more engagements are conducted it is expected that more antipatterns will be identified.

SOA adoption antipatterns

These are antipatterns that hinder or delay

ENGAGE AND EXPLORE...

The Technologies, Solutions and Applications that are Driving Today's Initiatives and Strategies...

CALL FOR PAPERS NOW OPEN!

SOA 10th International WebServices Edge conference+expo Edge 06

June 2006 | New York, NY

The Sixth Annual SOA Web Services Edge 2006 East - International Web Services Conference & Expo, to be held June 2006, announces that its Call for Papers is now open. Topics include all aspects of Web services and Service-Oriented Architecture

Suggested topics...

- > *Transitioning Successfully to SOA*
- > *Federated Web services*
- > *ebXML*
- > *Orchestration*
- > *Discovery*
- > *The Business Case for SOA*
- > *Interop & Standards*
- > *Web Services Management*
- > *Messaging Buses and SOA*
- > *Enterprise Service Buses*
- > *SOBAs (Service-Oriented Business Apps)*
- > *Delivering ROI with SOA*
- > *Java Web Services*
- > *XML Web Services*
- > *Security*
- > *Professional Open Source*
- > *Systems Integration*
- > *Sarbanes-Oxley*
- > *Grid Computing*
- > *Business Process Management*
- > *Web Services Choreography*

CALL FOR PAPERS NOW OPEN!

2006 ENTERPRISE> OPENSOURCE CONFERENCE+EXPO

June 2006 | New York, NY

The first annual Enterprise Open Source Conference & Expo announces that its Call for Papers is now open. Topics include all aspects of Open Source technology. The Enterprise Open Source Conference & Expo is a software development and management conference addressing the emerging technologies, tools and strategies surrounding the development of open source software. We invite you to submit a proposal to present in the following topics. Case studies, tools, best practices, development, security, deployment, performance, challenges, application management, strategies and integration.

Suggested topics...

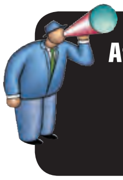
- > *Open Source Licenses*
- > *Open Source & E-Mail*
- > *Databases*
- > *ROI Case Studies*
- > *Open Source ERP & CRM*
- > *Open-Source SIP*
- > *Testing*
- > *LAMP Technologies*
- > *Open Source on the Desktop*
- > *Open Source & Sarbanes-Oxley*
- > *IP Management*

Submit Your Topic Today! events.sys-con.com

Sponsored by

WebServices Journal · XML Journal · .NET Developer's Journal · eclipse) developer's journal · WebSphere Journal · information Storage+Security journal · wldj · JDJ · LinuxWorld Magazine · MX developer's journal · asp.netPRO · SD Times · CoDe · Software Test & Performance

*Call for Papers email: events@sys-con.com



Attention Exhibitors:

An Exhibit-Forum will display leading Web services and OpenSource products, services, and solutions

For Exhibit and Sponsorship Information ▸ Call 201 802-3023

Produced by SYS-CON EVENTS

SOA adoption by customers and businesses.

Antipattern name: Technology bandwidth (see also : Web service = SOA)

- ***Problem:*** We observed that many corporations embark on a SOA effort leading from an IT perspective instead of a Business one. Implementations might be technically feasible and sometimes successful, but the impact on the business may not be realized since it was never considered in the first place.
- ***Context:*** This antipattern is mostly observed in large corporations with well established IT departments that employ highly technical staff supported by strong and influential technically oriented leadership.
- ***Symptoms:*** A common symptom of this antipattern is the inability of the sponsors to articulate the value proposition of SOA adoption. Also, lack of business alignment with the projects targeted for SOA implementation could be another symptom.
- ***Consequences:*** As a result of this antipattern, the cost of IT rises without realizing any return on investment (ROI). In addition, the corporation may lose the opportunity to infuse flexibility into the IT portfolio.
- ***Root Cause:*** In most cases, the root cause of this antipattern stems from the pressure on a corporation to match or keep up with announcements from competitors who may claim leadership through the adoption of this technology. As a result, the corporation might find it easier to dive into a technology-based effort to adopt SOA without taking the time and the effort to align it with the business needs.
- ***Solution:*** The best approach to deal with this antipattern is to establish hype-free SOA value propositions which can be accomplished by identifying and describing how SOA addresses client-specific pain points or business challenges. This approach can be complemented by the development of a roadmap for proper introduction of technology in support of the business.
- ***Solution example:*** Develop a SOA platform based on hype-free SOA value propositions

A global car rental company understood the value propositions of a SOA solution that would support its key business drivers:

- Provide a flexible business model to increase speed and flexibility in delivering new business services
- Drive down costs by streamlining processes to reduce operating costs
- Reduce cycle time and costs for external business processes by providing first-to-market innovative services for its customers
- Integrate across the enterprise by enabling easy and flexible integration to support multiple delivery channels

A2: Antipattern name: So, What's New?

- ***Problem:*** This antipattern describes a situation where skeptics in a corporation claim that SOA is just a name for same old techniques and that SOA doesn't offer any

Category - ID	Antipattern	Description
Adoption - A1	Technology Bandwagon	A noted common trend where corporations decide to jump on the new technology bandwagon without careful consideration whether this technology have any contribution to the business.
Adoption - A2	So, What's New?	Lack of understanding of the differences between SOA and previous computing paradigms drives skeptical to claim that SOA is just a name for same old techniques. The result is opposition to the adoption of SOA even if such adoption benefits the business.
Adoption - A3	The Big Bang	This antipattern is also known as "Bite more than you can chew." It is observed when SOA is viewed as a panacea, leading to a push to change all the enterprise systems and architecture at once. Such a big bang adoption could result in failures that are then unjustly blamed on SOA.
Identification & Design - I1	Web service = SOA	When architects equate SOA with Web services they run the risk of replacing existing APIs with Web services without proper architecture. This will result in identifying many services that are not business aligned.
Identification & Design - I2	The Silo Approach	In this antipattern, services are identified based on isolated applications, thus same services are identified by different groups with different names. As a result, no common services or service sharing are realized.
Identification & Design - I3	Misbehaving Registries	Duplicate service registries and overlapping, unclear ownerships result in governance nightmare and runtime confusion, potential bad performance, and unplanned costs due to duplication.
Realization - R1	Chatty Services	This antipattern describes a common mistake developers usually make when they realize a service by implementing a number of Web Services where each communicates a tiny piece of data. This will result in the implementation of a large number of services leading to degradation in performance and costly development.
Realization - R2	Point-to-point Services	Using XML or SOAP over HTTP between applications replacing middleware by point to point web services results in the emergence of point-to-point integration as the defacto integration pattern.
Realization - R3	Component-less Services	By jumping into development and implementation of Web services without having a clear association with owning components results in unstructured and undisciplined architecture (non strict layering). This leads to inflexibility behind the service interface and in preserving the legacy limitations of existing applications and packages.

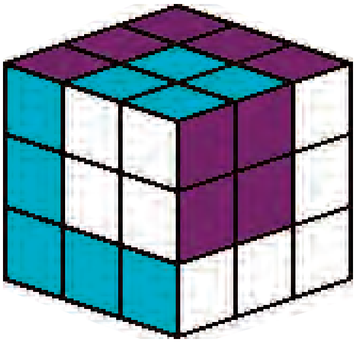
Table 3: List of antipatterns

thing new that they haven't been doing already. On the surface, that may appear to be so, but the emergence of Web services and XML, amongst other related standards is a major differentiator of what was done compared to what can be done with the appropriate implementation of SOA.

- ***Context:*** This antipattern is mostly adopted by IT personnel who are comfortable with the technology they have been using for an extended period of time and are reluctant to introduce or consider changes. It also appears in situations where IT departments have gone through a painful technology transformation or that the new technology didn't live up to its hype.
- ***Symptoms:*** The most obvious symptom is the strong opposition of some technical managers in a corporation to consider SOA as a serious approach to address legitimate business problems. The opposition could be in the form of strong and explicit arguments against the adoption of SOA, or it could be implicit and passive by ignoring SOA altogether while discussing approaches to solutions for business problems.
- ***Consequences:*** This antipattern most likely will foster lack of support for SOA that will result in missed opportunities to realize the SOA value propositions that will support business pain points.
- ***Root Cause:*** Though SOA builds on the same principles introduced and supported by other computing paradigms (for example, Object-Oriented and Component Based Development), many experienced IT teams lack real understanding of the differences between SOA and these other computing paradigms. This lack of understanding is one of the basic root causes of this antipattern. Another root cause is a direct consequence of IT teams having had bad experiences in implementing too many "Paradigm Shifts" and as such they are not willing to try a new one.
- ***Solution:*** One way to deal with this antipattern is to emphasize how SOA is different from earlier solutions. For example, the differences between an APIs and Services should be explored, and dependency on Open Standards and their differentiating attributes should be explained. Another major differentiator is the emergence of the Enterprise Service Bus (ESB) as an essential component of SOA. The facilities provided by an ESB such as Transport Services, Mediation Services, and Event Services, are examples of new capabilities made available by adopting SOA. However, the most effective solution is to provide successful examples that will highlight the differences and demonstrate the success and feasibility of implementing a SOA solution.
- ***Solution Example:*** SOA Education Educate both business and IT on what SOA is, its value propositions and the benefits it provides to deliver IT flexibility that is required to support business goals. Provide an understanding of the importance of the Web services and XML standards and emerging standards in the implementation of SOA that distinguishes it from past paradigm shifts.

A3: Antipattern name: The Big Bang (also known as: Bite more than you can chew)

- ***Problem:*** This antipattern can be thought of as the extreme opposite of the "So, What's New" antipattern. Here the problem is that the advocate view SOA as a panacea, leading to a push to change all the enterprise systems and architecture at once.
- ***Context:*** This antipattern shares the same context as the Technology Bandwagon antipattern. In particular those enterprises where major stake holders are very technically oriented and are more influential than their business counterparts, you can expect them to try to "bite more than they can chew" regardless of the effect on the business.



- ***Symptoms:*** Overwhelming concerns from business units about the proposed changes to adopt SOA is a clear indication that this antipattern has been exercised.
- ***Consequences:*** A result of applying this antipattern is serious failure to deliver on the promised benefits of SOA, which may lead to a return to existing architecture thus delaying or even eliminating any potential benefit from SOA. Another consequence of exercising this antipattern is to blame any failure to deliver on SOA rather than finding the root cause of the problem.
- ***Root cause:*** Zealous technologists in any enterprise can be the root cause of such an antipattern, particularly if these technologists are in a position to make IT decisions override business concerns.
- ***Solution:*** To eliminate this antipattern, one solution is to develop a business supported roadmap to move towards SOA incrementally. Instead of trying to solve all the enterprise problems through sweeping changes using SOA, it would be better to adopt an incremental approach by building a business case for a pilot first, then when the pilot is successful, use the roadmap to select the next target business area for which SOA would be beneficial. It will also be helpful if education on SOA entry points is made available to those in the organization that are responsible for the push to SOA.
- ***Solution example:*** Develop a roadmap to implement SOA Given its business strategy, a major bank determined that a SOA solution was the appropriate architectural direction that would enable it to meet its business goals. It first conducted an assessment of its maturity level in terms of service integration. Then a roadmap was developed to enable the bank to incrementally migrate to a SOA environment so as to minimize risks while continuing to provide the necessary business functionality. A pilot project was selected to validate the SOA architectural style and to work out any technology and organizational issues while moving to this SOA paradigm.

SOA identification & design antipatterns

These are antipatterns encountered while practitioners work on identifying and designing services as part of a SOA initiative.

Antipattern name: Web service = SOA (also known as Service Proliferation Syndrome)

- ***Problem:*** To many, SOA is just another name for Web services. Though implementing Web services is a legitimate entry point towards SOA adoption, enterprises should not equate Web services with SOA.
- ***Context:*** Most of today's production Web services systems are not SOAs. They're simple remote procedure calls or point-to-point messaging through SOAP or well structured integration architectures. Organizations find it easier to claim a move to SOA by only implementing Web services.
- ***Symptoms:*** Replacing existing APIs with Web services without proper architecture as well as implementation of services that are not business aligned are obvious symptoms of this antipattern.
- ***Consequences:*** Proliferation of Web services is a direct consequence of the application of this antipattern. This is a result of implementing any interface as a Web service regardless of whether these services are part of a SOA value proposition or not. Also, it is likely that the resulting systems will not have interfaces that reflect the requirements of the service requestor.
- ***Root cause:*** Root causes for this antipattern fall in two categories. The first is haste, where the intent is to take shortcuts and do this SOA stuff quickly and cheaply. The second is ignorance of the difference between SOA and Web Services and lack of understanding the need for good service modeling techniques.
- ***Solution:*** An approach toward addressing this antipattern is to develop a viable SOA transition plan which requires organizations to define their SOA value proposition. Achieving the SOA value propositions requires both SOA and Web services. This will require education on SOA versus Web services as well as applying good service modeling method such as SOMA (Service-Oriented Modeling and Architecture).
- ***Solution example:*** Apply service modeling to identify coarse-grained services and leverage Web services as the implementation technology for a SOA solution Adopt and use a good service modeling technique to determine appropriate coarse-grained services for a business solution and leverage Web services to implement a SOA solution. The right level of service granularity would help minimize the proliferation of services that are too fine-grained. Achieving the value propositions of SOA requires both SOA and Web services.

Name: The Silo Approach

- ***Problem:*** The problem presented by this antipattern is a result of the way services are identified. When services are identified based on isolated applications rather than focusing on the enterprise strategy, the benefits expected from SOA implementation never materialize.
- ***Context:*** Enterprises that are organized based on a functional model with towers that are working in isolation are prime candidates for this antipattern. The autonomy each functional tower enjoys lead to the working in a silo approach.
- ***Symptoms:*** Observing that same services are identified by different groups with different names is a symptom of this antipattern. Failure to identify common services or lack of realizing service sharing is also a clear indication that this antipattern has being applied.
- ***Consequences:*** Since services are identified and implemented multiple times as a result of the silo approach, costly development of unnecessary or duplicate services negates the benefits promised from adopting SOA. Furthermore, reuse between stake-holders of different applications diminishes or is completely eliminated as a result of this antipattern.
- ***Root cause:*** In most cases, organizational boundaries and constraints are the root cause of this antipattern. In some enterprises, the lack of a disciplined approach to "service exposure decisions" is attributed as the root cause.
- ***Solution:*** The solution to this antipattern is to establish a governance framework that will ensure cross tower service identification and communications. Also, a focused effort on identifying common services should be initiated and enforced. This can be accomplished using a good service modeling method such as SOMA.
- ***Solution example:*** Identifying common services. Like most corporations, a major bank was organized by business towers which provided similar business functionality to different types of bank clients. SOMA was applied to develop a service model to facilitate and understand opportunities of reuse across the business towers. At a high level, about 90 percent of the business services that were needed to support business functionality were found to be similar and were potential areas of reuse.

Antipattern name: Misbehaving Registries

- ***Problem:*** UDDI adoption without an agreed upon SOA governance model leads to process inefficiencies as well as design inefficiency. In particular duplicate registries can cause inefficiencies resulting in bad performance as well as creating security and compliance holes.
- ***Context:*** Enterprises that do not have an agreement on the registry governance but attempt to establish UDDI usage will often encounter this antipattern.
- ***Symptoms:*** Given that UDDI is a standard and not a product, its improper use can lead to the establishment of duplicate service registries and overlapping, unclear ownerships. This will manifest itself in the form of incorrect, misleading service interface information particularly during the discovery of composite service scenarios within a shared services model. Despite individual service interfaces having the correct description in the registry, duplicate entries will result in the discovery of incorrect whole services. Furthermore, lack of a governance model could result in services that are registered along various stages of life cycle process but become abandoned when there is no one to one ownership transfer. For example: a developer may add three services in to the internal UDDI. Then the service administrators may pick only two of them to be moved to the production UDDI. This results in orphaned services. A similar situation may arise with the service components.
- ***Consequences:*** Duplicate registries will create governance nightmare and

WHY 'ECLIPSE DEVELOPER'S JOURNAL' AND WHY NOW? PG 2

eclipse) developer's journal

PREMIER 2005 VOL 1 ISSUE 1

http://eclipse.sys-con.com

What's New in Eclipse 3.1

Polishing the apple ...4

PLUS...

WEB TOOLS PLATFORM: J2EE DEVELOPMENT THE ECLIPSE WAY ...12

'BEING A BETTER PLATFORM ...22 INTERVIEW WITH MIKE MILINKOVICH, EXECUTIVE DIRECTOR, ECLIPSE FOUNDATION

MYECLIPSE 4.0 UP ON DECK: EXCLUSIVE INTERVIEW WITH MAHER MASRI, PRESIDENT, GENUITEC ...36

ECLIPSE IDE FOR STUDENTS: USEFUL TIPS AND TRICKS ...28

runtime confusion in trying to identify which service in which registry that is causing failure to meet SLAs. Additionally, potential bad performance and security and compliance holes could result from such duplication. Unplanned costs due to duplication are also another consequence of this antipattern.

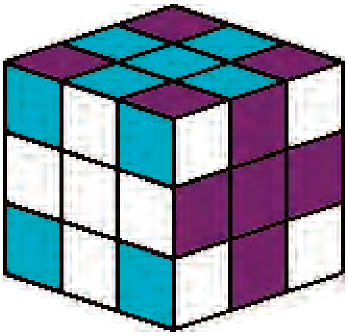
- ***Root cause:*** Lack of design-time coherence (SOA governance model) is the leading cause of this antipattern. In particular, unclear service registry ownership and lack of establishing and enforcing enterprise level SOA reference architecture are at the core of the problem.
- ***Solution:*** To avoid this antipattern, a corporation must establish a SOA governance model that defines and adopts service registry best practices, ownership, and communications across related teams. This would eliminate duplicate and incorrect service interface information and minimize the consequence of having orphaned services. Runtime confusion would also be eliminated since there is only one source of service descriptions which include the associated SLAs and policies. Establishing and enforcing compliance with a SOA enterprise level reference architecture should be the first step towards the institutionalization of a SOA governance model.

SOA realization antipatterns

These antipatterns capture worst practices for realizing services.

Antipattern name: Chatty Services

- ***Problem:*** This antipattern describes situations where developers realize a service by implementing a number of Web services where each communicates a tiny piece of data. Another flavor of the same antipattern is when the implementation of a service ends up in a chatty dialog communicating tiny pieces of information rather than composing the data in a comprehensive document-like form.
- ***Context:*** In organizations where developers are eager to start implementing without proper modeling, they end up being victims of this antipattern. In some situations, where developers are asked to start replacing APIs with Web services and without proper understanding of how to best benefit from SOA, it usually leads to this antipattern.
- ***Symptoms;*** The need to implement a large number of too finely grained services is an indication that this antipattern is being applied.
- ***Consequences:*** Degradation in performance and costly development are the major consequences of this antipattern. Additionally, consumers have to expend extra effort to aggregate these too finely grained services to realize any benefit as well as have the knowledge of how to use these services together.
- ***Root cause:*** As a result of not knowing any better, the approach many developers take is to mimic API implementation in the form of Web services. This is very similar to the situation we encountered at the beginning of the object-oriented paradigm shift, where developers used object-oriented languages to develop procedural programs. Also, excitement to dive into this new technology could cause everything to become a web service with no benefit and excessive cost.
- ***Solution:*** To avoid this antipattern, a focused effort should be on refactoring the design to combine individual pieces of data in one document thus eliminating the chatty services. Education on differences between API and service with focus on appropriate service granularity would also help. However, the most effective approach to avoid this antipattern is to define services that map back to business goals. This can be accomplished by adopting and using a good service modeling technique to determine appropriate coarse-grained services for a business solution. This minimizes the chatty behavior of services since it has been identified at the correct level of granularity appropriate in a SOA. Applying a Service Litmus Test (SLT) would also help determine the right level of services to expose. An example of a SLT is to consider whether a service provides a required unit of business functionality that supports business processes and goals.



Antipattern name: Point-to-point Services

- ***Problem:*** The basic problem is the attempt of developers to replace a middleware solution with point to point web services as an integration approach regardless of the usage context.
- ***Context:*** This pattern is observed in organizations where there is lack of long term systems integration vision with emphasis on short term results.
- ***Symptoms:*** An indication that this antipattern is being exercised is the use of XML or SOAP over HTTP between internal applications.
- ***Consequences:*** As a result of using this antipattern, the point-to-point integration solution emerges as the defacto integration pattern for the enterprise. This will negate any potential for achieving the full advantages of proper SOA implementation.
- Root cause: A lack of consideration for the long-term maintenance and evolution of the overall system, potentially as a result of focusing on short term solutions, is the principle root cause of this antipattern. In some cases, excitement about using services everywhere could also lead to the proliferation of such point to point services approach.
- ***Solution:*** To avoid the consequences of adopting this antipattern, the use of an intelligent connector such as a Service Bus should be seriously considered as an integration approach. A Service Bus enables applications to work together simply and efficiently to support business needs while maintaining loose coupling between collaborating systems and applications.
- ***Known exceptions:*** There are some known exceptions for which the antipattern solution is acceptable. For example,

a quick, short-lived integration scenario is required to solve immediate business problems and one case where point to point services could be used. However, there is a tendency for these solutions to stay in production for a long time. Thus applying this antipattern should be carefully monitored and controls should be put in place to prevent its long term adoption.

Antipattern name: Component-less Services (also known as Logical Layering is Obsolete)

- ***Problem:*** Good practices for service modeling promote associating identified services with components that own them. The problem presented by this antipattern is the tendency of developers to jump into developing and implementing Web services without having a clear association with owning components.
- ***Context:*** This antipattern appears in environments where architecture patterns are not applied or considered, for example, the layering architecture pattern. This lack of architectural discipline provides an environment that fosters the application of this antipattern.
- ***Symptoms:*** Examination of services' implementation will reveal that direct reach of any part of the system is allowed without adhering to any architectural structure. In these cases, Web services are developed without any regard to the concepts of layering and separation of concerns.
- ***Consequences:*** The most notable consequence is the inflexibility beyond the service interface since modular design, information hiding and logical structuring principles are all violated. This will lead to preservation of legacy limitations of existing applications and packages, which may prevent potential reengineering in the future.
- ***Root cause:*** As with any other situation where best practices are violated, the root cause of this antipattern is the lack of good design.
- ***Solution:*** The true potential of components and SOA will only be realized when you have both of them. Coherent service interfaces supported by flexible component-based applications is the solution. This will require developers to continue to leverage J2EE and general EAD best practices and layering patterns as a mean to overcome the shortfalls of this antipattern.
- ***Solution example:*** Understand the value of components with SOA
  Services are optimally realized with components. Without components, there is inflexibility behind the service interfaces, with potential concerns about scalability and portability of implementation. Existing applications and packages preserve their legacy limitations and this would minimize the ability to provide the flexibility needed to support changing business needs. Components provide the scalability and flexibility to support the service interfaces with its loose coupling and reuse features.

Summary

In this article, we reviewed some of the SOA antipatterns gleaned from observations and SOA projects that affects the adoption, identification & design, and realization of SOA.

Acknowledgements
The authors would like to thank the following colleagues for their contributions to this paper:

- JeanPierre Paillet
- Kerrie Holley
- Kyle Brown
- Mike Ellis
- Olaf Zimmermann
- Prathap Gangula
- Rick Robinson
- Rolando Franco
- Sri Ramanathan

Resources

Learn

- For more information on antipatterns, check out the following books:
- The Timeless Way of Building, by C. Alexander
- Notes on the synthesis of form, by C. Alexander
- The Oregon Experiment, by C. Alexander
- A Pattern Language: Towns, Buildings, Constructions, by C. Alexander
- Antipatterns: Refactoring Software, Architecture, and Projects in Crisis, by W. Brown, R. Malveau, H. McCormick III, and T. Mowbray
- J2EE Antipatterns, by B. Dudney, S. Asbury, J. Krozak, K.Wittkopf

- Web Services Architecture & Best Practices (IBM WebSphere Developer Technical Journal, Oct 2003) : Book excerpt that covers some of the architectural challenges posed by Web services, examines how to use (and not to use) Web services, and describes some best practices in applying Web services for solving tough architectural problems.
- " A taste of "Bitter Java"" (developerWorks, March 2002) : How antipatterns can improve your programming.
- SOA and Web services – hosts hundreds of informative articles and introductory, intermediate, and advanced tutorials on how to develop Web services applications.
- Architecture: Build for the future – visit the new Architecture area on developerWorks and get the resources you need to advance your skills in the architecture arena.

Discuss

- developerWorks blogs: Get involved in the developerWorks community.

> **"Software patterns provide a mechanism to capture knowledge and experience among architects and designers"**

IBM Supports Open Document Format as Way to Energize Emerging Economies

IBM has extended its support of what it calls open standards to the desktop by announcing that the upcoming version of the IBM Workplace Managed Client will support the newly ratified OpenDocument Format for Office Applications (ODF) spec. IBM customers can now help protect an organization's investment in corporate data by ensuring consistency, reliability and accessibility of their documents, the company said.

This announcement "is particularly relevant in emerging markets such as India, since acceptance of open standards, such as ODF, will help governments achieve their goal of truly bridging the digital divide and reaching out to citizens by deploying open, accessible standards -- not proprietary software," the company said in an official statement.

M. Moni, Deputy Director General, National Informatics Centre (NIC) said, ''As a premier government organization undertaking strategic e-Government/e-Governance projects at national and state levels, NIC has been promoting open standards, open source, and products and technology with clear R&D roadmaps to reach public services to unreached and grassroots development.

"They are the key factors underpinning our endeavors to leverage Information Technology as an enabling catalyst to develop and run products and services more effectively and in tune with India's growing digital based economy. NIC has received a mandate from the Central Department of Information Technology to work in the areas of standards to facilitate implementation of the National e-Governance Program in the country. The choice, flexibility and reliability inherent in open standards like ODF are critical in our efforts to drive the e-Governance / e-Government momentum in the country. I am happy to commend IBM's initiatives in this technology domain."

Open Standards Essential

As an increasing amount of corporate data is captured, stored and exchanged electronically, customers are becoming aware that the accessibility and maintenance of documents is critical to the long-term success of their organization, IBM says. Proprietary file formats force businesses into a perpetual cycle of royalty and licensing fees, which can restrict growth and innovation by requiring that companies stay locked into their original software choices just to ensure future access to their documents. "The ODF standard is a key development in the management of corporate data and documents -- organizations should not have to pay to access their own data, and the ODF standard ensures that key information like financial records, government contracts, payroll data and other corporate information is usable and accessible, regardless of your software platform," said Bob Suttor, Vice President of Standards and Open Source at IBM. "IBM's commitment to open standards based platforms, particularly the IBM Workplace portfolio of products and solutions, has provided businesses an alternative to proprietary systems, and addresses the market's demand for greater choice and flexibility in their software needs."

"India is fast emerging as a key country in the global technology landscape," says Harsh Gram, Vice President, IBM India Software Lab. "With the growth of open standards across the globe, Indian governments and businesses are realizing the need to embrace and contribute to open technology momentum while maintaining their freedom of choice. We have repeatedly seen how open standards increase customer choice in the marketplace and tear down barriers to integration. This announcement is a significant step towards bridging the digital divide by driving down prices and driving up access and interoperability and really demonstrates IBM's worldwide leadership and commitment to open standards."

New Support

The upcoming release of Workplace Managed Client 2.6 will include support for version 1.0 of the ODF standard. ODF is an XML based standard recently ratified by Organization for the Advancement of Structured Information Standards (OASIS) that pertains to office applications. OASIS is a not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards. IBM is a member of the OASIS Technical Committee that developed the standard.

The productivity editors within the upcoming version of the Workplace Managed Client, including word processing, presentation graphics and spreadsheets, will support the ability to import, export and rewrite files saved in the ODF standard. This support for industry standards will allow IBM customers to rapidly create, manage, and exchange documents within their business, as well as with their partners and customers. The support of ODF in IBM's Workplace Managed Client helps businesses of all sizes and industries take advantage of the cost and time effectiveness of open standards while leveraging their existing IT investments to help protect the reliability and efficiency of access to documents in the future.

IBM, Oracle, BEA, Jobs, Microsoft, and Sun Come Out Gunning at SYS-CON Application Server Shoot-Out

SYS-CON Media's Group Publisher Jeremy Gamelan and Java Developer Journal's Enterprise Editor Yukon Fain led the discussion at SYS-Can's second annual Application Server Shoot-Out - web cast live around the world from the famous 4th floor Reuters television studio in New York City's Times Square on Friday.

Executives from IBM, BEA, JBoss, Microsoft, Oracle, and Sun debated diverse aspects of their application server offerings. A discussion of price points, from the JBoss open-source model which emphasizes service revenues, to the more traditional

business models from most of the other vendors was followed by a discussion of the technical merits of each platform and how each company is uniquely striving to serve what it perceives as its customers' needs.

The six industry experts involved in the Shoot-Out this year were Mark Heid (IBM), Gary McBride (BEA), Shaun Connolly (JBoss), Dino Chiesa (Microsoft), Mike Lehmann (Oracle), and Rich Sharples (Sun).

The panel predictably discussed open source, but in ways that might have been unexpected. One of the more interesting discussion points, posed by Yakov Fain, for example, involved platform modifications (with J2SE, for example) and risk mitigation.

Chiesa said Microsoft's customers always have concerns that companies might disappear, in the manner of "Wang or SGI," so derive confidence in Redmond's approach to provide the features they demand while mitigating their risk by assuring customers that the company is here to stay. "I agree, risk mitigation is what this is really all about," echoed Mike Lehmann. "Oracle layers in the features that our customers find to be the most useful and help them drive and maintain competitive advantage."

"(And there is) risk around vendor lock-in," noted IBM's Heid. "But IBM has been a leader...for a very long time." He also noted that the company and its customers "can't be hogtied by the standards-setting process in cases where our customers want to move faster."

BEA's McBride picked up on Heid's point, saying, "Where there no standards available, BEA will innovate, and where there are (valuable) standards, BEA will adopt. We contribute greatly to the open standards community."

"At JBoss our mantra is open source, open standards," Connolly added. "Simplicity (is) as a key piece of our vision (and) we must remain portable. The benefit is that being an open source company really keeps each group building the application sever innovating at the highest levels."

Sun's Sharples noted that the key is to have software that is "integrated in that the stuff all works...and integratable (in that) when you are trying to replace enterprise middleware, you can't just rip things apart, you need standards. Java EE is one of them, for example." He added that this discussion is ongoing at Sun, among Sun's customers, and in the broader community, noting, "one important thing to think about is that the software ecosystem is bigger than Java (Enterprise). How would people like to see (it) grow? Should it grow up the stack, for example?"

Geelan commented that there seems to be "an embarrassment of riches" among customer offerings by these companies today, and asked them to focus on a single key aspect of what they offer to customers. The answers ranged from "simplicity" to "independence" to "completeness" to "flexibility" to "productivity," reflecting the diverse views held by the panel members and the companies they represent.

The hour allotted for this event passed quickly, with each panel member given a fair chance to answer several penetrating questions posed by Fain and Geelan. The entire webcast is available at www.sys-con.com.IBM Taps Boom in Linux Growth by Expanding Commitment to Partners, Linux and Open Source

Building upon the company's commitment to Linux and the Open Source Community, IBM announced Novell and Red Hat have been elevated to IBM's Strategic Alliance program -- IBM's highest tier partner status. The move will make it simpler for clients to acquire open standards-based Linux hardware, software, and services through integrated and streamlined sales, distribution and services channels.

- Opening significant new channels and access to IBM innovation centers to Novell and Red Hat, including those in emerging countries like Brazil, Russia, India, China and Korea to drive more open source deals in those booming markets.
- Enhanced New subscription models combining IBM offerings with Novell and Red Hat solutions.
- Support by Novell and Red Hat of IBM's open platforms including the Java-based Apache Geronimo web server and Apache Derby database, and continued support of the Eclipse development platform.

With 's news, the companies are reinforcing their commitment to the Java community, which comprises more than six million developers worldwide. Additionally, the companies will help customers begin to deploy Service Oriented Architectures (SOA) based on a J2EE application environment. SOA can give businesses the flexibility to easily integrate information with custom third-party business process applications regardless of the underlying technology, freeing clients from relying on any single information technology vendor.

The announcement represents the most significant elevation of IBM's strategic partnerships with its key Linux Distribution Partners se it embraced Linux six years ago, a testament to a Linux market that continues to experience strong growth. According to IT industry firm IDC, the overall Linux marketplace is growing annually at 26 percent and is expected to reach $35.7 billion by 2008. IBM now ranks number one in overall Linux-based server revenue worldwide with 29.7 percent of the revenue, up 32 percent year-to-year, according to Gartner, an industry research firm.

IBM Business Partners contribute over one third of IBM's Linux revenue. The ecosystem around these two alliances contributes to IBM's growth and represents the core of IBM's strategy of offering innovative open standard solutions that combine open source technology, IBM commercial products, and IBM's industry expertise. The alliances are timed to tap the boom in IBM's Linux growth expanding its base of 12,000 enterprise deployments worldwide in key industries like government, retail, health care and financial services, as well as emerging global markets.

WAS and ESB

Connect Non-SOAP HTTP Requesters and Providers to WebSphere Application Server V6 Enterprise Service Bus

BY GREG FLURRY

This article shows you how to connect non-SOAP HTTP service requesters and providers to the IBM® WebSphere® Application Server V6 Service Integration Bus. This lets requesters and providers leverage the integration capabilities of an enterprise service bus.

ABOUT THE AUTHORS

Greg Flurry is a Senior Technical Staff Member in IBM's Enterprise Integration Solutions group. His responsibilities include working with customers on service oriented solutions and advancing IBM's service oriented products.

IBM's WebSphere Application Server V6 (hereafter called Application Server) provides a platform for building an Enterprise Service Bus (ESB). Building an Enterprise Service Bus with WebSphere Application Server V6, Part 1: Introduction to WebSphere V6 Messaging Resources is the first part of a series that provides a wealth of information on WebSphere Messaging Resources, also known as the Service Integration Bus, or SIBus, available in Version 6. The SIBus supports many of the integration features required of an ESB. The SIBus ships with mechanisms to connect service requesters and providers that use SOAP/HTTP and Java™ Messaging Service (JMS).

However, there are many sources of service requesters and providers that don't use SOAP/HTTP or JMS. In this article, we'll talk about requesters and providers that use HTTP as a transport, but don't use SOAP, and may not even use XML as the payload. In some cases, such requesters and providers were developed before SOAP became widespread; in other cases, even for recently developed applications, SOAP is considered too heavyweight. See the Resources section for links to XML-RPC, REST, eBay-REST, eBay-XML and Yahoo to get a look at some non-SOAP mechanisms and applications that use them.

Non-SOAP HTTP requesters and providers can be an important part of a service-oriented environment. In this article, you'll learn how to connect such entities to the SIBus. This allows non-SOAP HTTP requesters and providers to benefit from the same integration logic available for SOAP/HTTP and JMS requesters and providers.

Our goal

Figure 1 shows a scenario in which a service requester interacts directly with service a provider via XML/HTTP. This means that the requester uses an XML-encoded payload for the request message sent to the provider and expects an XML-encoded payload as the response message. The requester uses HTTP to deliver the request and receive any response.

Figure 2 shows the scenario we want to achieve, where a requester interacts with a service provider through an ESB that can, for example, log requests, modify requests, or even route requests to a different provider. You'll see later that the payload does not have to be XML.

Figure 3 shows some additional scenarios possible with an ESB. Integration logic in the ESB can transform requests so that XML/HTTP requesters can interact with SOAP/HTTP providers and SOAP/HTTP requesters can interact with XML/HTTP

BY NOW THERE ISN'T A SOFTWARE DEVELOPER ON EARTH WHO ISN'T AWARE OF THE COLLECTION OF PROGRAMMING TECHNOLOGIES KNOWN AS AJAX!

REAL-WORLD AJAX

ONE DAY SEMINAR

www.ajaxseminar.com

March 13, 2006

Marriott

Marriott Marquis Times Square
New York City

For more information
Call 201-802-3022 or
email events@sys-con.com

How, in concrete terms, can you take advantage in your own projects of this newly popular way of delivering online content to users without reloading an entire page?

How soon can you be monetizing AJAX?

This "Real-World AJAX" one-day seminar aims to answer these exact questions...

Led by "The Father of AJAX" himself, the charismatic Jesse James Garrett of Adaptive Path, "Real-World AJAX" has one overriding organizing principle: its aim is to make sure that delegates fortunate enough to secure themselves a place – before all seats are sold out – will leave the seminar with a sufficient grasp of Asynchronous JavaScript and XML to enable them to build their first AJAX application on their own when they get back to their offices.

HURRY!
LIMITED SEATING
THIS SEMINAR WILL SELL-OUT
CALL 201-802-3022 TO REGISTER!

Jeremy Geelan

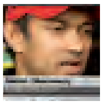
Jeremy Geelan
Conference Chair, Real-World AJAX
jeremy@sys-con.com

Jesse James Garrett
Father of AJAX

Scott Dietzen
Creator of WebLogic, Ph.D., President and CTO, Zimbra

Bill Scott
AJAX Evangelist of Yahoo!

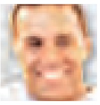
David Heinemeier Hansson
Creator of Ruby on Rails



Satish Dharmaraj
Father of JSP, Co-Founder & CEO, Zimbra

Rob Gonda
Best-Selling AJAX Author, CTO, iChameleon Group

Dion Hinchcliffe
Co-founder & CTO, Sphere of Influence Inc.



Ross Dargahi
Well-known AJAX Evangelist & Co-founder and VP of Engineering, Zimbra

Early Bird (Before January 31, 2006)	$995
Discounted Price (Before February 28, 2006)	$1,195
Seminar Price (After February 28, 2006, and if any seat is available)	$1,295

MEDIA SPONSOR
ajax developer's journal

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

PRODUCED BY SYS-CON EVENTS

NOTE: SPEAKER LINE-UP SUBJECT TO CHANGE WITHOUT NOTICE

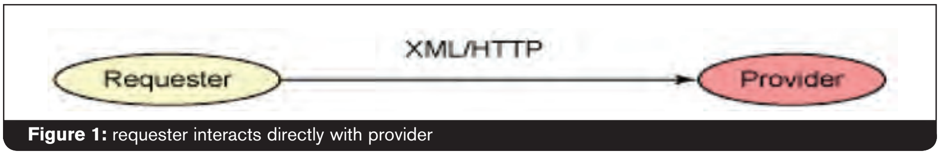


Figure 1: requester interacts directly with provider

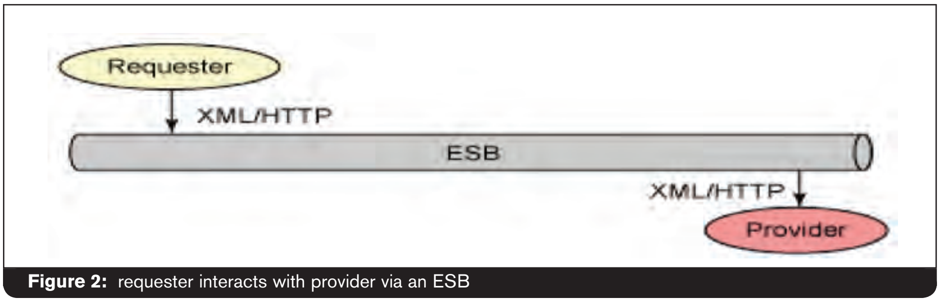


Figure 2: requester interacts with provider via an ESB

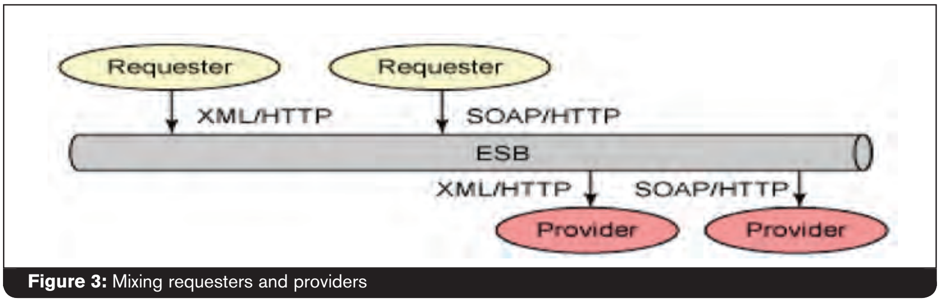


Figure 3: Mixing requesters and providers

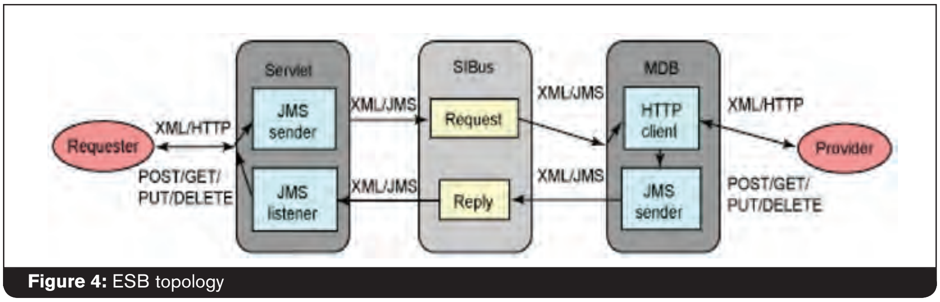


Figure 4: ESB topology

providers. However, in this article, we'll only deal with the scenario shown in Figure 2.

An analysis of the XML/HTTP <==> XML/HTTP scenario shows that the ESB must do the following:

- Receive the XML/HTTP request from the XML/HTTP requester. (At this point, integration logic in a mediation can run to process the request.)
- Send the request to the XML/HTTP provider
- Receive the response from the XML/HTTP provider. (At this point, integration logic in a mediation can run to process the response.)
- Return the response to the XML/HTTP requester

Our solution

Figure 4 shows an ESB topology that performs the steps described above. The figure shows a servlet that acts as an XML/HTTP *on-ramp* for the SIBus, in a manner very similar to the SOAP/HTTP on-ramp for Web services, called the *inbound Web service*, that ships with the SIBus. The servlet inserts the request into the SIBus, via the JMS sender, where it can be mediated if necessary, and returns the response to the requester, via the JMS listener. A Message-Driven Bean (MDB) acts as an XML/HTTP *off-ramp* for the SIBus, in a manner very similar to the SOAP/HTTP off-ramp for Web services, called the *outbound Web service*, that ships with the SIBus. The MDB sends the request to the XML/HTTP provider, via the HTTP client and inserts the response into the SIBus, via the JMS sender, where it can be mediated if necessary.

Note that to support the spectrum of possible non-SOAP request types, the ESB must preserve the fidelity of the HTTP request and response. In particular, the servlet must deal specifically with the following HTTP methods:

- POST, where the requester sends body content in the request and expects body content in the response.
- GET, where the requester only expects body content in the response.
- PUT, where the requester sends body content in the request, but expects no body content in the response.
- DELETE, where the requester sends no body content in the request, and expects no body content in the response.

In addition, the servlet's JMS sender function inserts the HTTP method, the HTTP path info (part of the URL past the root), the HTTP parameters and the HTTP headers (including cookies) from the HTTP request into a JMS message inserted into the SIBus. For the POST and PUT methods, the JMS sender also inserts the request payload or body (which might be XML) into the JMS message. The JMS sender then sends the message to an SIBus queue destination (the yellow box labeled Request in Figure 4). The servlet then invokes its JMS listener to listen for the response.

The MDB listens for messages arriving on the Request queue. Upon receiving a message, the HTTP client function of the MDB constructs an HTTP request using all the information placed in the request JMS message by the servlet. There is one significant semantic difference between the newly constructed HTTP request and the original HTTP request received by the servlet: the root URL is different so that when the HTTP client sends the request, it goes to the expected provider. The MDB HTTP client waits for the synchronous

HTTP response from the provider.

The MDB JMS sender inserts any response payload, the HTTP headers (including Set-Cookie headers) and status code from the HTTP response into a JMS message. The JMS sender also sets the correlation ID of the JMS response message to the message ID of the JMS request message, and then sends the JMS response message to an SIBus queue destination (the yellow box labeled Reply in Figure 4).

The servlet's JMS listener listens for an incoming message on the Reply queue with a JMS selector set to match a response correlated with the request; the correlation ID of the response must be equal to the JMS request message ID. Upon receiving the correlated response message, the JMS listener inserts only the response status code into the HTTP response if the response is not OK. If the response is OK, the JMS listener inserts the response (XML) payload and any headers from the JMS message into the HTTP response. The JMS listener then returns the HTTP response to the original requester.

Implementation

Listing 1 shows the servlet skeleton used for the on-ramp. The init() method gets the JMS connection factory and queues, and starts a connection. The connection is held open for the lifetime of the servlet, and is not closed until the servlet's destroy() method gets called. Configuration information, such as JNDI names, is described in detail in the next section.

Listing 1. Servlet skeleton

```
public class XMLHTTP2SIB extends javax.servlet.
http.HttpServlet
implements javax.servlet.Servlet {

  // connection factory and queue resources
  private final static String

  JMSCF_JNDI_NAME = “java:comp/env/MDBCF”;
  private final static String JMSRequest_JNDI_NAME
=

  “java:comp/env/MRequest”;
  private final static String

  JMSReply_JNDI_NAME = “java:comp/env/MReply”;
  // session and connection
  private Session session = null;
  private Connection connection = null;
  // request and reply queues
  private Destination requestQueue = null;
  private Destination responseQueue = null;


  public void init() throws ServletException
{

  try {

  InitialContext context = new InitialContext();

  ConnectionFactory factory = (ConnectionFactory)

  context.lookup(JMSCF_JNDI_NAME);
            requestQueue = (Destination)

  context.lookup(JMSRequest_JNDI_NAME);
            responseQueue = (Destination)

  context.lookup(JMSReply_JNDI_NAME);

  connection = factory.createConnection();

  session = connection.createSession(

  false,

  Session.AUTO_ACKNOWLEDGE);


  connection.start();

  } catch (NamingException e) {

  …

  } catch (JMSException e) {

  …

  }
  }

  public void destroy() {

  try {

  connection.close();

  } catch (JMSException e) {
    …

  }
  }
}
```

Before describing the servlet request and response processing logic, let's discuss the utility class HTTPOutput, partially shown in Listing 2 below. This class is used to convey information from the HTTP response. It has a returnCode field that carries the HTTP response status. The header field contains any response headers, where each member in the Vector is a two-element array of String; the first member of the array is the header name and the second is the header value. The utility class also contains the body in the response field. Note that only the returnCode field is guaranteed to be meaningful; the others can be null.

Listing 2. HTTPOutput class

```
public class HTTPOutput {

  private int returnCode;
  private byte[] response;
  private Vector header = null;

  // setters and getters not shown
}
```

Listing 3 shows the servlet's doPost() method. The method first uses the private sendJMSMessage() method to map the HTTP request into a JMS message and insert that message into the SIBus, as shown below. Note that sendJMSMessage() returns the ID of the message to use in a JMS selector; this helps correlate the response to the request. doPost() then listens for the response on a JMS queue by calling the private getJMSResponse() method; as shown below. If doPost() finds that the status code from the response indicates that request was unsuccessful, it simply returns the status code; otherwise it returns the entire response body and headers. This ensures the response returned to the original requester semantically matches that returned by the actual provider.

Note that header names require a bit of special processing using the utility function XMLHTTPUtil.unfixName(), because JMS properties are used to send the HTTP headers in the response JMS message, and Java does not allow the dash character to appear in JMS property names. The use of the dash is common in HTTP header names, and a companion utility function, XMLHTTPUtil.fixName() used in the MDB that sends the response, replaces any dash characters in a name with underscore characters, as you'll see later.

Listing 3.

```
doPost() method of XMLHTTP2SIB servlet
protected void doPost(HttpServletRequest request,
HttpServletResponse
response) throws ServletException, IOException {
   // send request
   String selector = sendJMSMessage(request.get-
Method(), request);
   // listen for response
   HTTPOutput output = getJMSResponse(selector,
response);
   // return response
   if (output == null) {
   throw new IOException(“bad response from HTTP
provider”);
   } else {

   if(output.getReturnCode() !=
(HttpServletResponse.SC_OK)) {

   response.setStatus(output.getReturnCode());

   } else {

   // do headers

   Vector header = output.getHeader();

   if ((header != null) && (header.size() > 0)) {

   for (int i = 0; i < header.size(); i++) {

   String[] entry = (String[])header.get(i);

   response.setHeader(XMLHTTPUtil.
                 unfixName(entry[0]), entry[1]);

   }
     }
   if (output.getResponse() !=null) {

   response.getOutputStream().

   write(output.getResponse());

   response.getOutputStream().close();

   }

   }
   }
}
```

The servlet's doGet(), doPut() and doDelete() methods all simply call the doPost() method described above. The sendJMSMessage() method deals with the differences in the request for the various HTTP methods (POST and PUT have body content, GET and DELETE don't), and as you can see, the doPost() method itself deals with the differences in the response for the various HTTP methods (POST and GET have body content, PUT and DELETE don't).

– continued from page 12

Strategies for Software Development Project Success

"Talking" marketing

"Talking" marketing refers to how you communicate technical features through the media. Sometimes the target audience for a marketing effort is very technical, but more often, such efforts are pitched at business decision makers who are not technical at all. The best way to help your marketing organization transform feature details into meaningful information for a non-technical audience is to fully understand the benefits and value of a new product or enhancement, and be prepared to help express them in metrics such as dollar amounts.

Be ready to provide clear answers to questions such as:

- "Why is this new product or enhancement important?"
- "How does it support the key initiatives in our business strategy?"
- "Who will benefit from the new capabilities?"
- "How does the product work?"

In my experience, there is always a way to explain even the most complex technical solutions in a relatively simple manner that summarizes the motivation behind the solution and its benefits. The challenge is to find a model that most of the people in your audience can understand.

Conclusion

In the face of the complex business and technology demands that globalization has brought about, today's organizations have to concentrate both non-technical and technical resources on creating a successful product and a strong market presence. Although there is no silver bullet that can protect your software development organization from all the challenges it might encounter in the course of a project, recognizing the importance of synchronization among all the different departments and functions within your organization will better prepare you for meeting every challenge that comes your way.

In part two, next month, I'll explore ensuring effective testing and supporting marketing efforts for software development project success.

Notes:

Standards for writing code can be industry specific and are often mandatory requirements.

e-BUSINESS SOLUTIONS: TAKING BUSINESS TO THE NET
SPECIAL: PREMIER ISSUE!
WebSphere DEVELOPER'S JOURNAL
The World's Leading Independent WebSphere Developer Resource
PREMIER VOLUME 1 ISSUE 1
INTRODUCTORY CHARTER SUBSCRIPTION SAVE $30 Today! - SEE PAGE 5 FOR DETAILS -
Iterative Development with VisualAge for Java
Visiting and revisiting your code throughout the cycle
Best Practices in WebSphere Application Packaging and Deployment
Building an Application Server Data Center
IBM WebSphere Studio v3.5 Advanced
It's You, Not the Software
The World's Leading Independent WebSphere Developer Resource

Extending JAAS

Advanced Authentication in WAS PART 3

BY KEYS BOTZUM, BILL HINES,
PAUL ILECHKO, MESSAOUD BENANTAR

The advanced authentication features in IBM WebSphere Application Server V6 support a more flexible authentication model with a new highly customizable authentication framework that's based on – and extends – Java Authentication and Authorization Service (JAAS).

ABOUT THE AUTHORS

Keys Botzum is a Senior Technical Staff Member with IBM Software Services for WebSphere. He holds a Masters degree in Computer Science from Stanford University and a B.S. in Applied Mathematics/Computer Science from Carnegie Mellon University.

Bill Hines is a Senior Certified Consulting IT Specialist with IBM Software Services for WebSphere. His expertise includes installation, configuration, tuning, dynacache, security, troubleshooting, and design/architecture of enterprise J2EE WebSphere applications.

Paul Ilechko is a Senior Solutions Architect with IBM Software Services for WebSphere. He has been involved with WebSphere and J2EE technology almost since their inception. He has a B.Sc. in Mathematics from the University of London.

Messaoud Benantar is a senior software engineer with IBM Software Services for WebSphere. He spent most of his career as a software developer working in IBM Large Systems Division, Software Group/Tivoli and IBM Global Services. His focus is on middleware and computing security.

Figure 15 shows the pane that is displayed after the module has been added. Notice that the default is to add it at the end of the stack, third in order as shown. We will need to use the highlighted Set Order button to fix this.
Figure 16 shows the pane that is presented when using the Set Order control to change the order of the login modules. In this figure, we have already selected the radio button for our login module and pressed the Move Up button twice to move it to be at the top of the stack.

Pressing OK in the Set Order pane presents Figure 17. It may be disconcerting at first to see the new login module still at the bottom of the order, but this is only due to the default alphabetical ordering of modules. Looking at the Module order column confirms that our module is indeed the first in sequence.

- We start our EJB client using launchClient. The EJB client connects to WebSphere Application Server and looks up our EJB and then calls it. As you might expect, the standard WebSphere Application Server authentication process applies so we will be challenged to provide our user ID and password (not shown). Listing 7 shows the launchClient process. If you want to run this example yourself, you will need to install the provided ReverseEJBEAR.ear application, which contains the components that are used by launchClient.

Listing 7:

```
>launchclient
"c:\home\presentationsPapers\AdvancedAuthPaper\
ReverseClientEAR.ear"
-CCBootstrapPort=2810 hello
IBM WebSphere Application Server, Release 6.
J2EE Application Client Tool
Copyright IBM Corp., 1997-2004
WSCL0012I: Processing command line arguments.
WSCL0013I: Initializing the J2EE Application
Client Environment.
[6/20/05 9:47:04:923 EDT] 0000000a  W UOW=null
source=com.ibm.ws.util.ImplFactory
org=IBM prod=WebSphere component=Application Server
thread=[P=223911:O=0:CT]
          WSVR0072W: Ignoring undeclared override
of interface,
com.ibm.websphere.cluster.topography.
DescriptionManager, with implementation,
com.ibm.ws.cluster.topography.
ClientDescriptionManager
WSCL0035I: Initialization of the J2EE Application
Client Environment has completed.
WSCL0014I: Invoking the Application Client class
Main
String provided: hello
String in reverse: olleh
```

- After the authentication challenge takes place, the RMI_INBOUND login module configuration is called. Our custom login module gets control first. In the initialize() method, the references to the subject and callbackhandler are stored.

Listing 8:

```
public void initialize( Subject subject,
CallbackHandler callbackHandler,
Map arg2, Map arg3 ) {

System.out.println( "MyBeforeLTPALoginModule ini-
tialize()" );
this.subject = subject;
this.callbackHandler = callbackHandler;
}
```

- Next, the login() method is called by the WebSphere Application Server security runtime. The code shown below is executed in the login() method. Much of this should look familiar, as it is similar to what we did in the TAI example. One area of code that is new, and of particular interest, is the callbackHandler section (discussed earlier). In this particular context, our login module calls out to the JAAS callback handlers provided by WebSphere Application Server to retrieve the username and password that were entered on the basic authentication screen. Also, notice that we used the code described in the login types section to check for the propagation login scenario using the WSTokenHolderCallback. The string arguments to the username and password callbacks may seem superfluous, but providing empty strings results in an IllegalArgumentException.

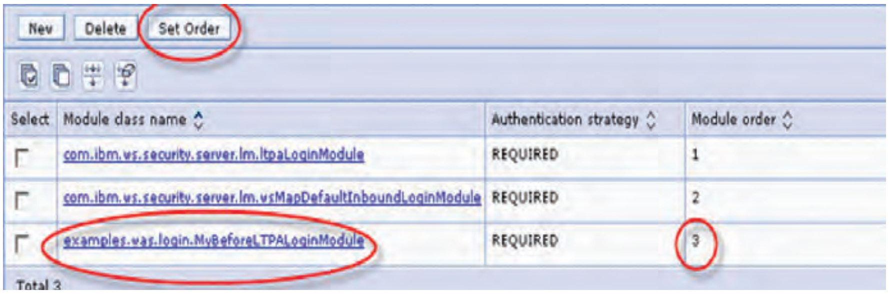
Listing 9:

```
public boolean login() throws LoginException  {
  System.out.println( "MyBeforeLTPALoginModule
login()" );

  Callback callbacks[] = new Callback[ 3 ];
  try {
    callbacks[ 0 ] = new WSTokenHolderCallback(
"" );
    callbacks[ 1 ] = new NameCallback( "User:" );
    callbacks[ 2 ] = new PasswordCallback(
"Password:", false );
    callbackHandler.handle( callbacks );
  } catch ( Exception e ) {
    System.out.println( "Login Module failed: " +
e );
  e.printStackTrace( System.out );
    throw new LoginException( e.getMessage() );
  }

  boolean requiresLogin = ( (WSTokenHolderCallback
)callbacks[ 0 ] ).getRequiresLogin();
  if ( requiresLogin ) {
    System.out.println( "MyBeforeLTPALoginModule:
Need to do stuff for an initial login" );
    String username = ((NameCallback)callbacks[ 1
]).getName();
    String password = new String( ((PasswordCallb
ack)callbacks[ 2 ]).getPassword() );
      ((PasswordCallback)callbacks[ 2 ]).clear-
Password();
    Hashtable hashtable = new Hashtable();
    String uniqueid = null;
    try {
      InitialContext ctx = new InitialContext();
      UserRegistry reg = (UserRegistry)ctx.look-
up( "UserRegistry" );
      uniqueid = reg.getUniqueUserId( username );
      reg.checkPassword( username, password );
    } catch ( Exception e1 ) {
System.out.println( "Login Module failed: " + e1
);
      e1.printStackTrace( System.out );
      throw new LoginException( e1.getMessage()
);
    }
    System.out.println( "uniqueid = " + uniqueid
);
    hashtable.put( AttributeNameConstants.
WSCREDENTIAL_UNIQUEID, uniqueid );
    hashtable.put( AttributeNameConstants.
WSCREDENTIAL_SECURITYNAME, username );

// Assert the user as belonging to only this
group. This is done only
// as an example of how to manipulate the user's
groups dynamically.
// We could also have looked up the user's groups
in the registry with
// reg.getUniqueGroupIds() and added this one, or
subtracted others
// from that list.
ArrayList groups = new ArrayList();
groups.add( 0, "nomemb" );
hashtable.put( AttributeNameConstants.WSCREDENTIAL_
GROUPS, groups );

  // Set unique cache key to prevent cache prob-
lems (losing custom
// info, etc) since the info is always the same
for the user, I
// just prefix this with
  // the login module name and userid.
// But, if the info was unique to *this login* I'd
need a more
    // unique value...
  System.out.println( "Using new cache key" );
  hashtable.put( AttributeNameConstants.
WSCREDENTIAL_CACHE_KEY,
 uniqueid + "ExampleBeforeLTPALoginModule" );
  subject.getPublicCredentials().add( hashtable );
  return true;
  } else{
    System.out.println( "MyBeforeLTPALoginModule:
This is a repeat login, nothing to do." );
    return true;
  }
}
```

- The only other significant code in this module is to return true from the commit() method. Listing 10 shows the edited output from the server side EJB. The EJB itself does nothing of interest in our example other than print out information about the current user. Notice that the user has been placed in to the group we asserted, rather than whatever groups might have been found in the WebSphere Application Server registry.

Listing 10:

```
[6/20/05 10:14:19:503 EDT] 00000057 SystemOut
O MyBeforeLTPALoginModule
initialize()
[6/20/05 10:14:19:503 EDT] 00000057 SystemOut
O MyBeforeLTPALoginModule login()
[6/20/05 10:14:19:503 EDT] 00000057 SystemOut
O MyBeforeLTPALoginModule: Need to do
stuff for an initial login
[6/20/05 10:14:19:503 EDT] 00000057 SystemOut
O uniqueid = user:customRealm/3
[6/20/05 10:14:19:503 EDT] 00000057 SystemOut
O Using new cache key
[6/20/05 10:14:19:533 EDT] 00000057 SystemOut
O MYAfterwsMapLoginModule being
loaded
[6/20/05 10:14:19:533 EDT] 00000057 SystemOut
O MYAfterwsMapLoginModule
initialize()
[6/20/05 10:14:19:533 EDT] 00000057 SystemOut
O MyAfterWsMapLoginModule login()
[6/20/05 10:14:19:533 EDT] 00000057 SystemOut
O MyBeforeLTPALoginModule commit()
 [6/20/05 10:14:20:755 EDT] 00000056 SystemOut
O In reverseString
[6/20/05 10:14:20:765 EDT] 00000056 SystemOut
```



Select	Module class name	Authentication strategy	Module order
□	com.ibm.ws.security.server.lm.ltpaLoginModule	REQUIRED	1
□	com.ibm.ws.security.server.lm.wsMapDefaultInboundLoginModule	REQUIRED	2
□	examples.was.login.MyBeforeLTPALoginModule	REQUIRED	3

Total 3

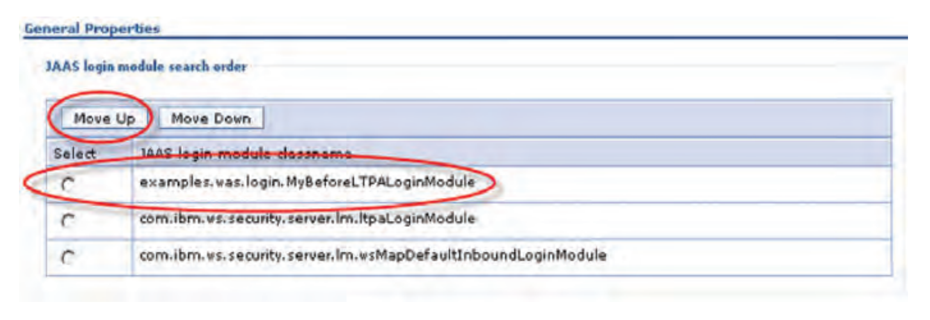
Figure 1: MyBeforeLTPALoginModule added to the WEB_INBOUND stack

```
O The WebSphere Application Server
Subject layer thinks you are keys
[6/20/05 10:14:20:765 EDT] 00000056 SystemOut
O Looking into your Subject I see
these groups:
[6/20/05 10:14:20:765 EDT] 00000056 SystemOut
O Group ID: group:customRealm/nomemb
[6/20/05 10:14:20:765 EDT] 00000056 SystemOut
O looking into Subject for custom
stuff
```

A more sophisticated login module could have asserted a different, arbitrary user ID and group information. The key is that the login module, because it is asserting the user information to WebSphere Application Server, takes on the responsibility for authentication. In our simple example, we used the WebSphere Application Server registry, but you could easily develop your own much more customized authentication approach.

Example: Subject modification login module

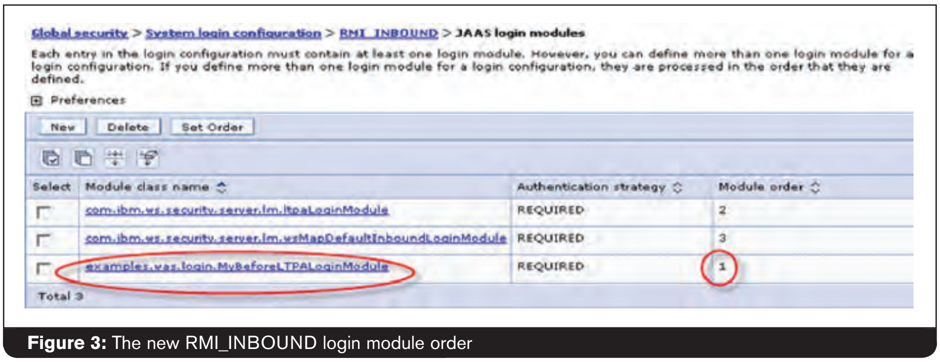
Our next example shows how a login module can be used to alter the subject after it has been created by WebSphere Application Server. To accomplish this, we will place our login module after the wsMapDefaultInboundLoginModule per our earlier discussion. Figure 18 shows the login module being added to the RMI_INBOUND configuration.


General Properties

JAAS login module search order

Select	JAAS login module classname
○	examples.was.login.MyBeforeLTPALoginModule
○	com.ibm.ws.security.server.lm.ltpaLoginModule
○	com.ibm.ws.security.server.lm.wsMapDefaultInboundLoginModule

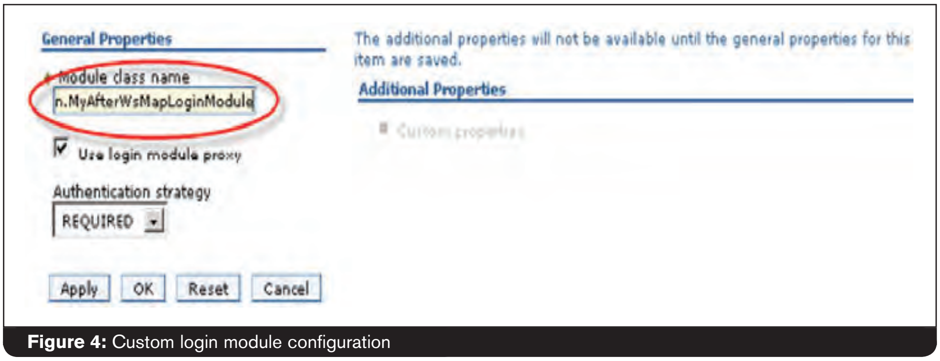
Figure 2: The Set Order admin console pane


Global security > System login configuration > RMI_INBOUND > JAAS login modules

Each entry in the login configuration must contain at least one login module. However, you can define more than one login module for a login configuration. If you define more than one login module for a login configuration, they are processed in the order that they are defined.

Select	Module class name	Authentication strategy	Module order
□	com.ibm.ws.security.server.lm.ltpaLoginModule	REQUIRED	2
□	com.ibm.ws.security.server.lm.wsMapDefaultInboundLoginModule	REQUIRED	3
□	examples.was.login.MyBeforeLTPALoginModule	REQUIRED	1
Total 3			

Figure 3: The new RMI_INBOUND login module order


General Properties

* Module class name: n.MyAfterWsMapLoginModule

☑ Use login module proxy

Authentication strategy: REQUIRED

The additional properties will not be available until the general properties for this item are saved.

Additional Properties

Figure 4: Custom login module configuration

Figure 19 shows our custom login module in place with the two standard WebSphere Application Server system modules for RMI_INBOUND. Our login module is placed in the appropriate place to modify the subject after WebSphere Application Server has created it, which is after the wsMapDefaultInboundLoginModule. There is no need to change the order as we did in the last example, since the default places it where it needs to be, at the end of the stack.

We will use the same session EJB as in the last example, which will result in the same basic authentication process as before:

- Login again as the same user. When the user is authenticated, the login modules are executed on the RMI_INBOUND configuration. The two standard WebSphere Application Server modules (seen in Figure 19) are executed, and then control reaches our module. The initialize() method shown below, in which we store off the reference to the subject for later use, is executed.

Listing 11:

```
public void initialize( Subject subject,
CallbackHandler arg1,
Map sharedState, Map arg3 ) {

System.out.println( “MYAfterWsMapLoginModule ini-
tialize()” );
  this.subject = subject;
}
```

Next, the login() method is called:

Listing 12:

```
public boolean login() throws LoginException{
System.out.println( “MyAfterWsMapLoginModule
login()” );
   return true;
}•
```

This would be the point where, in some cases, we might perform authentication of the user, such as in our earlier example, where we assert the user's identity to

OUT OF MEMORY! PAGE 36
Magazine in the World
JDJ
IN THIS ISSUE...
Leveraging Open Source Solutions to Improve Productivity While Using EJBs
Spring Web Flow
Intelligent Web Applications with AJAX
Navigating the Global Enterprise
JDJ.SYS-CON.COM
ARE PORTALS THE MAGIC BULLET OF WEB APPLICATION DEVELOPMENT?
The many advantages to utilizing portal software
PLUS...
The Performance of EJB 3.0
Dealing with Open Source Software
A Blueprint for Developing Language Tools
THE WORLD'S LEADING i-TECHNOLOGY MAGAZINE JDJ.SYS-CON.COM

WebSphere Application Server. In this case, since we are letting WebSphere Application Server create the Subject, authentication is not necessary and we can simply return true. The subject is not fully formed at this point, so we will not attempt to alter it here.

- The next method to gain control in our login module is commit() and, in our case, this is where the real action happens:

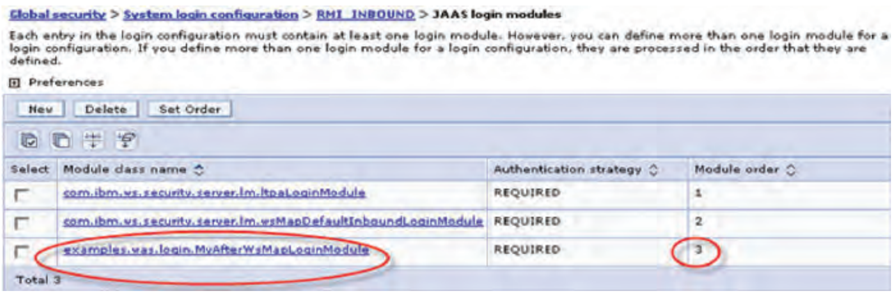
Listing 13:

```
public boolean commit() throws LoginException {
  String id = null;
  System.out.println( "MyAfterWsMapLoginModule
commit() " );
  Set stuffset = subject.getPublicCredentials(
CustomJAASStuff.class );
  if ( stuffset != null && stuffset.isEmpty() ) {
    System.out.println( "MyAfterWsMapLoginModule
never been called before." );
    CustomJAASStuff stuff = new CustomJAASStuff();
    Set principals = subject.getPrincipals();
    if ( ( principals != null ) && ( !principals.
isEmpty() ) ) {
      Iterator iter = principals.iterator();
Principal p = (Principal)iter.next();
      id = p.getName();
    }
    StringBuffer id2 = null;
    id2 = new StringBuffer( id );
    stuff.setWord1( "Caller principal in caps = "
+ id.toUpperCase() );
    stuff.setWord2( "Caller principal in reverse
= " +
      id2.reverse().toString() );
    System.out.println( "MyAfterWsMapLoginModule:
adding to subject" );
    subject.getPublicCredentials().add( stuff );
} else {
System.out.println( "Found existing
CustomJAASStuff. Must be a reinvocation." );
    return true;
  }
}
```

- For this example, we have created a plain old Java object (POJO) class that contains a few string objects (word1, word2) and a date object. First, we ensure the class does not already exist, and then we instantiate one for our use. The code retrieves the principal from the subject, and the user ID is retrieved from that. The two String objects in CustomJAASStuff are populated with the user ID, converted to all capitals and reversed. The custom object is then added to the public credentials of the subject, and our commit() method returns true, which commits our changes to the subject.
- The following code has been added to the EJB to check for our modifications when the subject is passed to it:

Listing 14:

```
System.out.println("looking into Subject for cus-
tom stuff");
Set set = WSSubject.getCallerSubject().getPublic-
Credentials(
CustomJAASStuff.class);
if ((set != null) && (!set.isEmpty())) {
System.out.println("your subject appears to have a
CustomJAASStuff in it");
  iter = set.iterator();
  int i = 1;
while (iter.hasNext()) {
    System.out.print("#" + i);
    CustomJAASStuff stuff = (CustomJAASStuff)
iter.next();
    System.out.print(": And in it I find word1 =
"
      + stuff.getWord1());
    System.out.println(". And the date is "
      + stuff.getDate());
i++;
  }
```



Global security > System login configuration > RMI_INBOUND > JAAS login modules

Each entry in the login configuration must contain at least one login module. However, you can define more than one login module for a login configuration. If you define more than one login module for a login configuration, they are processed in the order that they are defined.

⊞ Preferences

Select	Module class name	Authentication strategy	Module order
□	com.ibm.ws.security.server.lm.ltpaLoginModule	REQUIRED	1
□	com.ibm.ws.security.server.lm.wsMapDefaultInboundLoginModule	REQUIRED	2
□	examples.was.login.MyAfterWsMapLoginModule	REQUIRED	3

Total 3

Figure 5: Custom login module placement

- The results are shown in Listing 15.

Listing 15:

```
[6/20/05 9:47:17:301 EDT] 00000057 SystemOut     O
MYAfterwsMapLoginModule being loaded
[6/20/05 9:47:17:301 EDT] 00000057 SystemOut     O
MYAfterwsMapLoginModule initialize()
[6/20/05 9:47:17:301 EDT] 00000057 SystemOut     O
MyAfterWsMapLoginModule login()
[6/20/05 9:47:17:301 EDT] 00000057 SystemOut     O
MyAfterWsMapLoginModule commit()
[6/20/05 9:47:17:311 EDT] 00000057 SystemOut     O
MyAfterWsMapLoginModule never been
called before.
[6/20/05 9:47:17:311 EDT] 00000057 SystemOut     O
MyAfterWsMapLoginModule: adding to
subject
 [6/20/05 9:47:19:013 EDT] 00000056 SystemOut
O In reverseString
[6/20/05 9:47:19:033 EDT] 00000056 SystemOut     O
The WebSphere Application Server
Subject layer thinks you are keys
[6/20/05 9:47:19:033 EDT] 00000056 SystemOut     O
Looking into your Subject I see
these groups:
[6/20/05 9:47:19:033 EDT] 00000056 SystemOut     O
Group ID: group:customRealm/2
[6/20/05 9:47:19:033 EDT] 00000056 SystemOut     O
looking into Subject for custom
stuff
[6/20/05 9:47:19:033 EDT] 00000056 SystemOut     O
your subject appears to have a
CustomJAASStuff in it
[6/20/05 9:47:19:033 EDT] 00000056 SystemOut     O
#1: And in it I find word1 = Caller
principal in caps = CUSTOMREALM/KEYS. And the date
is Mon Jun 20 09:47:17 EDT 2005
```

- Notice that our login module is called in the "initial login" state. After our login module completes, the reverseString EJB method executes and simply prints out the subject, which, you may notice, contains the custom information we placed in it (CustomJAASStuff). Also, the user group here is different from the previous example even though we did authenticate as the same user. That is because rather than assert user or group information in this login module, we let WebSphere Application Server get it from the user registry.

Propagation

Overview

Earlier, we discussed requirements for integrating custom logic with the authentication model supported by WebSphere Application Server, making use of both the TAI interface and the ability to add custom JAAS login modules to the supported login configurations. What this custom logic does is add information to the JAAS subject that affects the security attributes of the application; that is, the custom information that was added to the subject may be used within the application to make additional, finer grained, authorization decisions. This can include the use of calculated or dynamic group information that is not stored in any registry.

It is possible for custom JAAS login modules to add information to the subject during a system login. This might include generic Java objects which may or may not be serializable. However, it is important to remember that you are probably not running on a single, standalone server, so there are implications as to exactly what you add to the subject for it work as expected in a distributed environment. The problem with attempting to recreate custom data is that it is not always possible to do so. In some cases, the custom data generated at the first server may have been dynamically derived. For example, one can derive a location identifier based on the originating browser's IP address, derive a specialized role from a form login attribute, or gather authorization attributes specific to the time-of-day that the login first occurred. These are examples of dynamic authorization attributes that could be lost as soon as a user moves to a second server. To be useful in a distributed, clustered environment, it must be possible to propagate this custom subject. There are two distinct types of propagation that we will address:

- ***Horizontal propagation*** involves making the subject available to all application servers that are accessed by Web clients.
- ***Downstream propagation*** is the ability to pass custom credentials via the RMI/IIOP layer.

Tokens

For propagation to occur, subject information needs to be transmitted over the network. WebSphere Application Server uses Java serialization for custom objects that have been added to the subject; it is therefore essential to use good serialization practices, as defined by Sun™, to support version-to-version compatibility. WebSphere Application Server itself does not depend on Java serialization for transmitting its own information. Instead, WebSphere Application Server relies on a token framework, where tokens represent various aspects of a user's identity or the subject. The token types used by WebSphere Application Server are:

- ***Authentication token*** - Represents a user identity (not groups, just base identity) and flows downstream over RMI/IIOP connections; equivalent to the older (pre-WebSphere Application Server V5.1.1) version of the LTPA token.
- ***Single sign-on (SSO) token*** - Used for horizontal propagation of user identity, this token is converted to a cookie and sent back to the browser, and is analogous to the older version of the LTPA cookie. The SSO token is essentially another version of an Authentication token that is suitable for use with a Web browser.
- ***Authorization token*** **-** A new token type that contains most of the user information from the subject, including all group information.

These tokens represent aspects of the subject. A fourth token, the propagation token, is not user specific, but it represents the thread context for downstream propagation. The propagation token will not be discussed here.

TO BE CONTINUED IN THE NEXT ISSUE

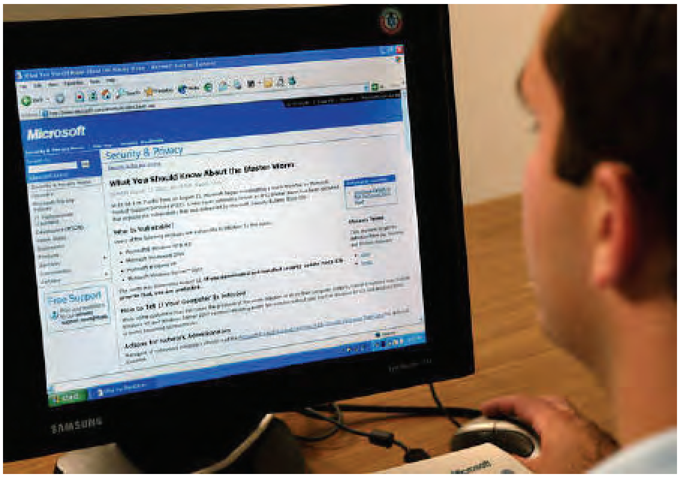
How to Keep Your Job

BY ROGER STRUKHOFF

So it appears that the power of the pen is omnipotent. In this very space just a couple of months ago, the writer was heard complaining about certain aspects of a college football team, and a few short weeks later the coach of said team lost his job. Wow.

This is either a metaphorical example of the "butterfly in motion effects all events in the universe" theory, or a completely meaningless and random connection that has nothing to do with anything.

But, given that the latter scenario is most likely the correct one, it does hit on a very profound and universal truth: we can all be fired. Nobody ever got fired for buying IBM has been an industry cliché for decades, and in most cases, it probably still holds true.



But is it the primary reason for the success of IBM WebSphere in the application space? To refer back to the college football column, in which an IBM Global Services salesperson was wondering whether his company was "standing still" or not, is there anything more going on at Big Blue and among its customers other than inertia?

The answer to this question is "yes," of course, and this writer won't bother to provide the IBM PR department spin as to why. Everyone knows why IBM thinks it has been successful, and everyone also know why *they* think the company and its technology is working for them.

Because the truth of the matter is you *can* be fired for buying IBM, or certainly if you think that all you *need* to do is buy and let it fly. There are serious software development issues within any organization, and in a day of multiplying specifications, "standards," languages, and approaches, these issues are often apparent in an environment that requires everything to be done today, perhaps with teams dispersed throughout the world, perhaps without the unambiguous communications procedures and management controls in place that are absolutely required if one is to make any sort technology actually work.

The unlucky college football coach had most likely done everything in his mind that could have been done. One could go through all the aspects of the organization—recruiting, training, offensive and defensive strategies and tactics, special teams, nice uniforms and everything—and find that everything looked fine. But the team simply did not perform on the field as it should have for too many years' running, and it was time "to move in a new direction."

For those of us in the business world, the timeline is much shorter than several years. If our efforts do not perform as expected for one quarter (let alone a full game or many seasons), we are in danger of losing our jobs, no matter how much IBM-branded stuff is lying around the shop.

The essence of successful development is knowing how to use the tools, how to focus them onto the business problems that are at hand, and at the end of the day being sure that our competitors are not moving more quickly and more intelligently than us.

One can surmise that IBM's continued success is deeply related to the company's support of its technology, something that my new Global Services friend is focused on every day as he continues to worry that his company is not doing enough. This is the sort of spirit that will enable him to keep his job, it seems.

This is the gist of the "Does IT Matter?" argument. Certainly, it matters, but since everyone has access to the same tools it is increasingly difficult to seize and maintain that competitive advantage that is crucial to company growth and our own personal job maintenance!

ABOUT THE AUTHOR

Roger Strukhoff, editor-in-chief of *WebSphere Journal*, is West Coast Bureau Chief for the SYS-CON News Desk, and President of www.wdva.com. He spent 15 years with Miller Freeman Publications and The International Data Group (IDG), then co-founded CoverOne Media, a custom publishing agency that he sold in 2004. His work has won awards from the American Business Media, Western Press Association, Illinois Press Association, and the Magazine Publishers' Association. Read his blog at http://www.rssblog.linuxworld.com. roger@sys-con.com

WebSphere.
IBM.