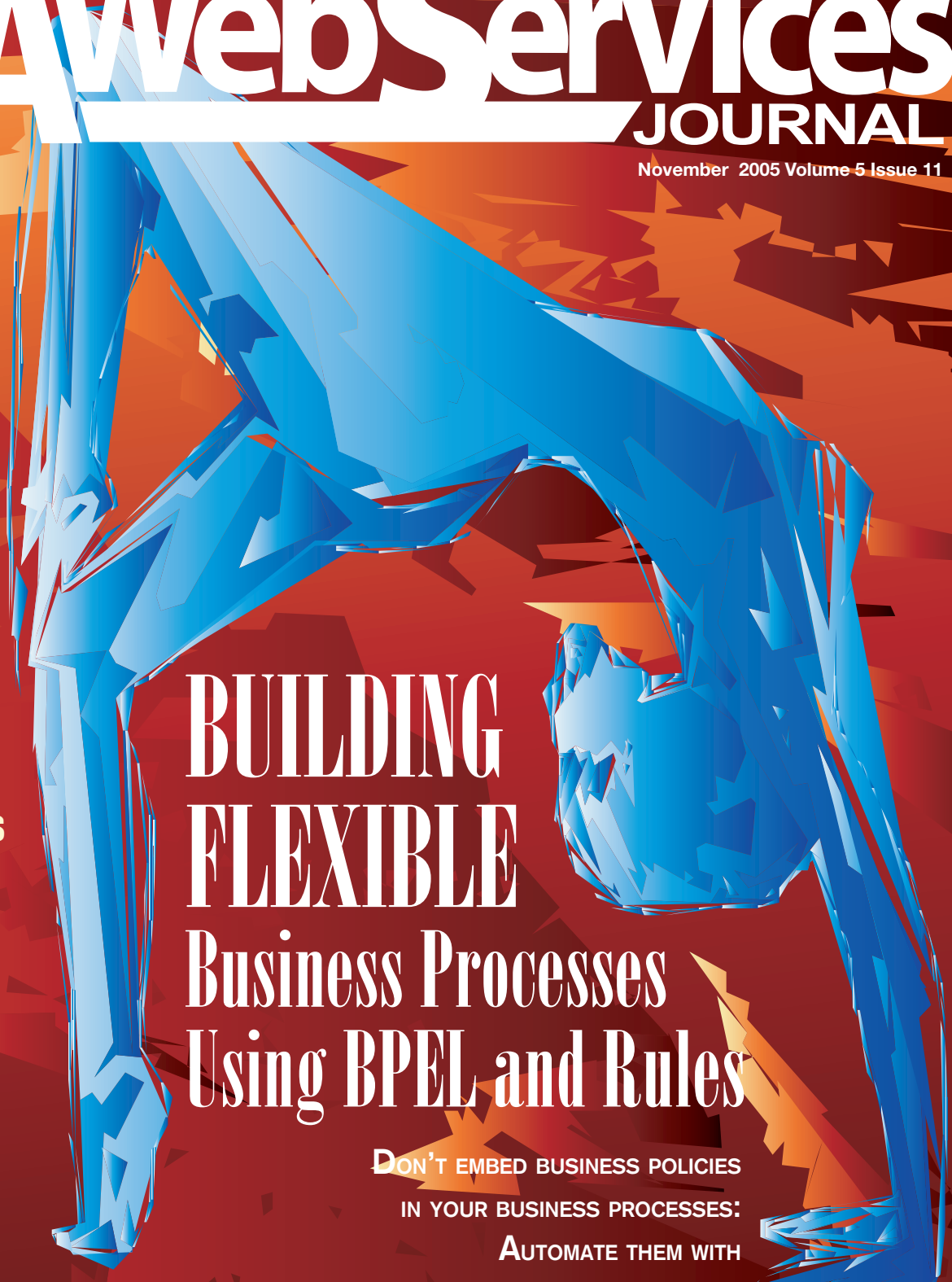
SYS·CON

NOW INCORPORATING XML JOURNAL

SOA Web Services JOURNAL

November 2005 Volume 5 Issue 11



BPEL4WS 1.1 to WS-BPEL 2.0

A migration path

pg.10

Ten Things Your EII Vendor May Not Tell You

The hidden complexities of implementing an EII solution

pg.22

Business Process Orchestration

BPEL supports time-critical decision making

pg.28

BUILDING FLEXIBLE Business Processes Using BPEL and Rules

DON'T EMBED BUSINESS POLICIES IN YOUR BUSINESS PROCESSES: AUTOMATE THEM WITH A BUSINESS RULES ENGINE!

PG.36

RETAILERS PLEASE DISPLAY UNTIL JANUARY 31, 2006

$6.99US $7.99CAN

11>

71486 03420 9

Ensure Interoperability.

Validate functionality.

Eliminate security vulnerabilities.

Test performance & scalability.

Confirm compliance.

Collaborate thru reuse.

Ensure Secure, Reliable Compliant Web Services

PARASOFT®

SOA Test™

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

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

If you are building serious web services, you need SOA Test.

Download a copy of SOA Test for a free evaluation today at www.parasoft.com/SOA_WSJ

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

PARASOFT®

We make software work.™

Automated Software Error Prevention™

Forum XWall™ Web Services Firewall - Reinventing Security

Security should never be an inhibitor to new opportunity: Forum XWall™ Web Services Firewall has been enabling Fortune 1000 companies to move forward with XML Web services confidently. Forum XWall regulates the flow of XML data, prevents unwanted intrusions and controls access to critical Web services.

Visit us at www.forumsys.com to learn more about how you can take your next leap forward without increasing the risks to your business.

FORUM SYSTEMS™ – THE LEADER IN WEB SERVICES SECURITY

SOA Web Services Journal

NOVEMBER 2005 Volume 5 Issue 11

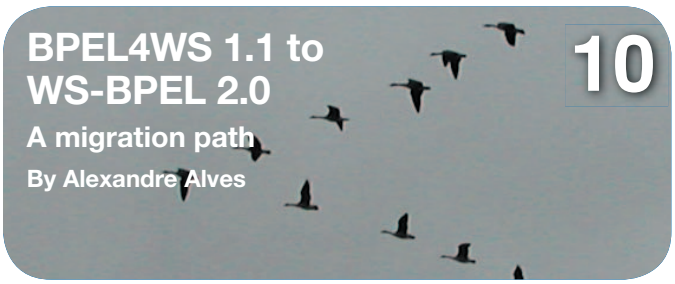
XML Journal

Inside WSJ

Building Flexible Business Processes Using BPEL and Rules

Don't embed business policies in your business processes: Automate them with a business rules engine!

By Mohamad Afshar & Bhagat Nainani



BPEL4WS 1.1 to WS-BPEL 2.0

A migration path

By Alexandre Alves

Ten Things Your EII Vendor May Not Tell You

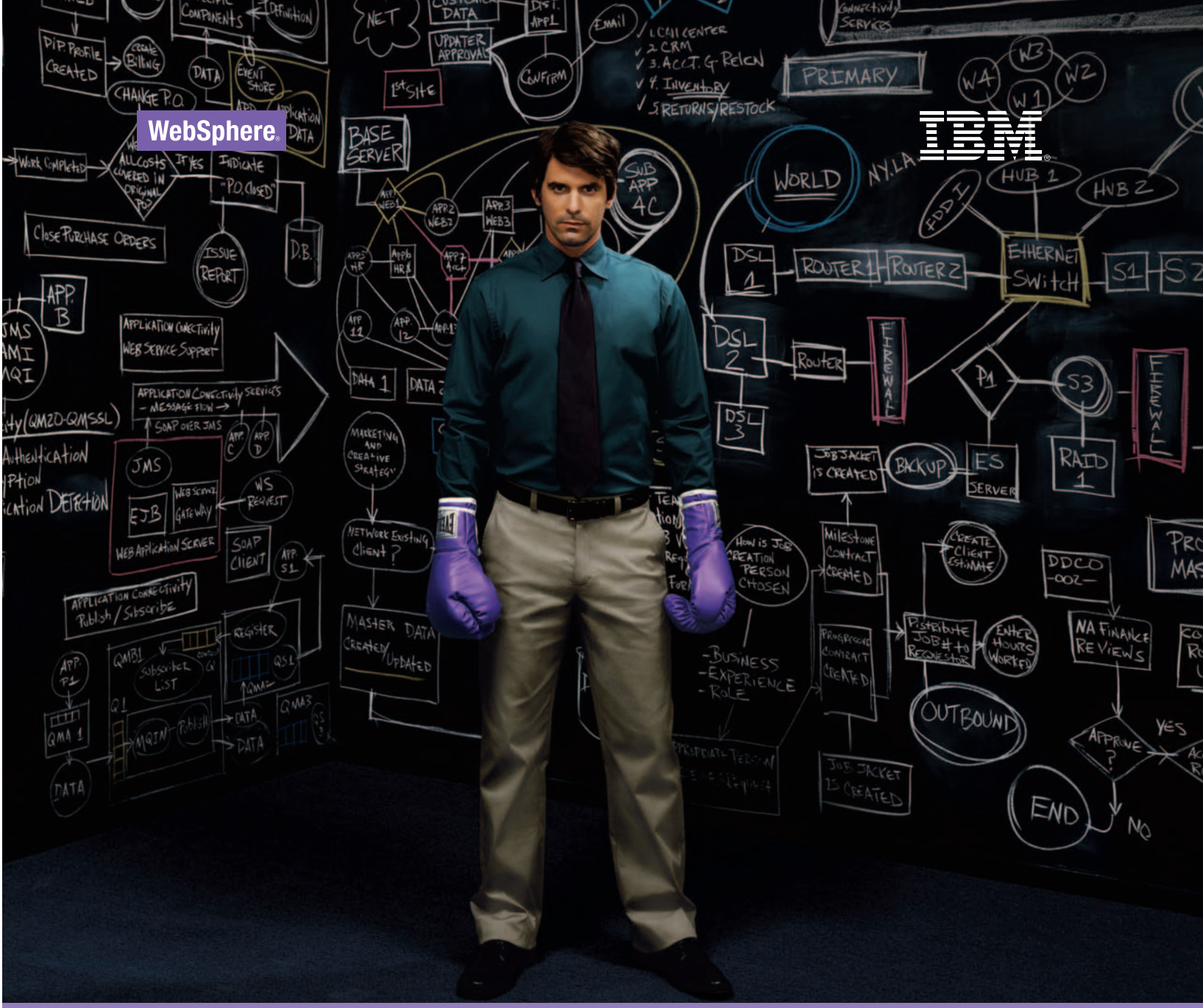
The hidden complexities of implementing an EII solution

By Tieu Luu

Business Process Orchestration

BPEL supports time-critical decision making

By Matthew Zager



IBM WEBSPHERE PRESENTS

YOU VS REWRITE, REVISE RE-EVERYTHING

INFLEXIBILITY: MEET SOA

PLUS: TANGIBLE BUSINESS BENEFITS ★ BEST PRACTICES FOR BEST RESULTS

OVER 10 YEARS OF WORLD-CLASS INTEGRATION EXPERTISE

FEATURING:

A FASTER, EASIER WAY TO IMPLEMENT TRUE SOA	SUPPORT FOR OVER 80 O.S. CONFIGURATIONS	UNPARALLELED INDUSTRY KNOWLEDGE & PROCESS SKILL

IBM MIDDLEWARE. POWERFUL. PROVEN.

FIGHT BACK AT WWW.IBM.COM/MIDDLEWARE/SOA. THIS IS A RIP-AND-REPLACE-FREE EVENT.

Drown your competition with Intermedia.NET

Looking for a hosting company with the most cutting-edge technology? With **Intermedia.NET** you get much more than today's hottest web & Exchange hosting tools – you get a decade of experience and an unmatched reputation for customer satisfaction.

Thousands of companies across the globe count on us for reliable, secure hosting solutions...and so can you.

In celebration of 10 successful years in the business, we're offering a promotional plan for just $1. Visit our website www.intermedia.net to find out more.

Our premier hosting services include:

- Windows 2003 with ASP.NET
- ColdFusion MX with Security sandboxes
- Linux with MySQL databases
- E-Commerce with Miva Merchant Store Builder
- Beneficial Reseller Programs
- ...and much more!

Unprecedented power, unmatched reputation... Intermedia.NET is your hosting solution.

INTERMEDIA.NET

Call us at: 1.888.379.7729
e-mail us at: sales@intermedia.NET
Visit us at: www.intermedia.NET

Web Services JOURNAL
.NET J2EE XML

EDITORIAL OFFICES
SYS-CON MEDIA
135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9637
WEB SERVICES JOURNAL (ISSN# 1535-6906)
Is published monthly (12 times a year)
By SYS-CON Publications, Inc.
Periodicals postage pending

Montvale, NJ 07645 and additional mailing offices
POSTMASTER: Send address changes to:
WEB SERVICES JOURNAL, SYS-CON Publications, Inc.
135 Chestnut Ridge Road, Montvale, NJ 07645

What Don't You Get?

WRITTEN BY
SEAN RHODY

SOA – service-oriented architecture – seems to be on everyone's radar. It's rare to walk into an IT meeting where someone hasn't bombarded the audience with the current buzzwords, and where someone isn't extolling the virtues of an SOA. Somehow, even though it's not really about a single application, even application designers are "building SOA in."

The funny thing though is, as with sex, everyone is talking about it but no one is doing it. Well, maybe not no one, but there's an awful lot of fence-sitting going on. In part, I can see it as very similar to the Internet. When it first came out, everyone had to be on it, but not that many were, and when they did get there, it took them a while to figure out what to do. Similarly, SOA is being paraded as the latest, greatest cure for the common cold. It slices, dices, changes your oil, and rotates your tires.

Now, SOA really DOES do a lot for an organization. In fact, it's such a good idea that once you have it explained to you, your first reaction is likely to be "well, yeah, of course." SOA is a natural idea for organizing systems and maximizing the return on investment in IT. However, that same obvious perspective hides another SOA truism – measuring the return on investment is very difficult.

It may seem intuitively obvious to reorganize systems so that they can communicate with one another without barriers or without needing specialized software adapters. However, putting a price tag on interoperability can be very difficult. In the end, have you returned a significant business value if you invest millions to reorganize, but don't gain any functionality in the process? Maybe, especially if you can reduce your operating and maintenance costs, but those are typically soft numbers, ones that don't have the same emotional impact on a corporation as increasing sales by 10 percent, or lowering production costs by a factor of two. One of the most significant challenges in achieving SOA is finding the business value that you can bring to the table during the process, which is why many SOA projects are funded by using SOA as the underpinning to achieve a business goal, such as consolidating redundant systems, while adding new functionality.

Another aspect of SOA that holds back adoption is the question of who will go first. SOA often adds cost to the first few projects; instead, it's the later projects that reap the benefits of SOA architecture. In large companies this can add to the delay in moving to SOA, as different departments play a game of musical chairs to see who will foot the bill for additional infrastructure and retooling of the development group for an SOA approach. Much as the children trying to get Mikey to eat Life cereal, departments say, "Not me, let Mikey do it."

A third problem in the adoption of SOA is that a services approach really asks a company to look at the way they do business. There's a governance issue, as well as the greater issue of who owns what. Obviously, this is heavy political territory, and it's not easy to get someone to give up their "turf" just because it may be in the best interests of the company. In the end, many organizations will need to redefine the ways they incent people in order to adopt an SOA approach, even if it appears straightforward and obvious on the surface.

Slowly though, the technologies that support SOA, namely Web services, are making their way, finally, into the corporate mainstream. Thankfully, there is enough of a vendor groundswell beneath Web services that the eventual, obvious move to SOA will ultimately take place for most organizations.

About the Author

Sean Rhody is the editor-in-chief of ***Web Services Journal***. He is a respected industry expert and a consultant with a leading consulting services company.

sean@sys-con.com

WebServices JOURNAL .NET J2EE XML

PRESIDENT AND CEO
Fuat Kircaali fuat@sys-con.com
VP, BUSINESS DEVELOPMENT
Grisha Davida grisha@sys-con.com
GROUP PUBLISHER
Jeremy Geelan jeremy@sys-con.com

ADVERTISING
SENIOR VP, SALES & MARKETING
Carmen Gonzalez carmen@sys-con.com
VP, SALES & MARKETING
Miles Silverman miles@sys-con.com
ADVERTISING DIRECTOR
Robyn Forma robyn@sys-con.com
NATIONAL SALES & MARKETING MANAGER
Dennis Leavey dennis@sys-con.com
ADVERTISING MANAGER
Megan Mussa megan@sys-con.com
ASSOCIATE SALES MANAGERS
Kerry Mealia kerry@sys-con.com

SYS-CON EVENTS
PRESIDENT, SYS-CON EVENTS
Grisha Davida grisha@sys-con.com
NATIONAL SALES MANAGER
Jim Hanchrow jimh@sys-con.com

CUSTOMER RELATIONS/JDJ STORE
CIRCULATION SERVICE COORDINATORS
Edna Earle Russell edna@sys-con.com

sys-con.com
CONSULTANT, INFORMATION SYSTEMS
Robert Diamond robert@sys-con.com
WEB DESIGNERS
Stephen Kilmurray stephen@sys-con.com
Vincent Santaiti vincent@sys-con.com
Shawn Slaney shawn@sys-con.com

ACCOUNTING
FINANCIAL ANALYST
Joan LaRose joan@sys-con.com
ACCOUNTS PAYABLE
Betty White betty@sys-con.com
ACCOUNTS RECEIVABLE
Gail Naples gailn@sys-con.com

SUBSCRIPTIONS
SUBSCRIBE@SYS-CON.COM
1-201-802-3012
1-888-303-5282
For subscriptions and requests for bulk orders, please send your letters to Subscription Department
Cover Price: $6.99/issue
Domestic: $69.99/yr (12 issues)
Canada/Mexico: $89.99/yr
All other countries: $99.99/yr
(U.S. Banks or Money Orders)

Worldwide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ
Newsstand Distribution Consultant:
Brian J. Gregory / 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

Openly Managing Web Services

WRITTEN BY
AJIT SAGAR

Last month I talked to a couple of vendors who are making new inroads in the services arena through open source offerings. Open source support in Web services is definitely very heartening. While the frameworks and utilities for implementing Web services in enterprise applications have matured, the standards of critical functions of promoting the "service bus" concept, which decouples Web services from the realization of an SOA, and the effective deployment and management of services, are still evolving. ESB as a concept has caught on very well in architecture discussions and vision, but I haven't seen too many examples of where large enterprises have actually implemented the design in their applications.

One of the main reasons for this is the fact that although ESB is offered by the prevalent applications server, messaging, and other "plumbing" vendors, it is only recently that implementations from the community have made their way into the industry. The way a technology usually makes inroads into an organization (at least medium sized, where commercial vendors don't have a monopoly) is through open source offerings. For example, frameworks such as Struts and Axis are not just the de facto standards, but are also the most widely adopted standards. Every app server vendor who has any presence in the market supports them.

Web services have matured to a level where their build and maintenance are becoming a part of the traditional SDLC. Going into the build and project management space, Maven is one of the prime examples of how the open source offerings were packaged to enable the process instead of just the technology. Maven utilizes Ant for build management, JUnit for unit testing, Jalopy for formatting source code, Checkstyle for validating Java source against a coding standard, and Sun's javadoc for standard JDK code documentation. One of the vendors I mentioned at the beginning of this document, Mergere, has leveraged the Maven and Continuum project to create a software build and orchestration platform for Web services. Mergere created an offering to leverage best practices in the form of tools that create a build automation dashboard for services.

The other vendor that I talked to, LogicBlaze, is focused on a technology layer in the services stack – ESB. While ESB implementations are available from a few vendors, as I mentioned earlier, the application of the bus in enterprise applications is still rare. One of the main challenges is the lack of standard APIs to enable a protocol-agnostic interface to the bus, without which the concept of a ubiquitous enterprise bus is hard to digest. The JBI spec, announced in June at JavaOne 2005, attempts to address this problem. The latest release of ServiceMix (2.0), the open source product from LogicBlaze, leverages JBI in its ESB offering.

While open source offerings from vendors facilitate the adoption of products that enable service-orientation in enterprise applications, their acceptance in large organizations is still lagging. This is because of the same reason that has plagued adoption of open source from the start – support. Although the products are adequately supported through a community, dedicated professional services and support groups that businesses can bank their business on are not generally available. As a result, open source products are leveraged in organization through commercial vendor toolsets or for non-production environments.

Partnerships are needed between vendors with open source offerings and system integrators that offer professional services to large clients. System integrators can offer the missing link to enable adoption of economical options to expensive vendor offerings in large organizations.

About the Author

Ajit Sagar is a principal architect with Infosys Technologies, Ltd., a global consulting and IT services company. Ajit has been working with Java since 1997, and has more than 15 years experience in the IT industry. During this tenure, he has been a programmer, lead architect, director of engineering, and product manager for companies from 15 to 25,000 people in size. Ajit has served as JDJ's J2EE editor, was the founding editor of *XML Journal*, and has been a frequent speaker at SYS-CON's Web Services Edge series of conferences. He has published more than 100 articles.

ajitsagar@sys-con.com

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

These days, with mission-critical web services running on Java, keeping your enterprise applications available can be pretty stressful.

Unless of course, you've discovered the power of Wily. No other software offers our level of insight. Which means you'll be able to monitor web services transactions in real time, find the root cause of problems fast and optimize performance across the entire SOA application environment — end-to-end.

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

Get Wily.™

1 888 GET WILY | wilytech.com

wily technology

BPEL4WS 1.1 to WS-BPEL 2.0

A migration path

BPEL4WS V1.1 is a public draft release of the "Business Process Execution Language for Web Services" specification dated May 3, 2003. BPEL4WS V1.1 is arguably the de facto standard for Business Process Management (BPM); however, because it's a draft release, BPEL4WS V1.1 has several shortcomings that will be addressed by the next release of the specification (named WS-BPEL 2.0), which is targeted to be released either toward the end of this year or during the beginning of 2006.

WRITTEN BY
ALEXANDRE ALVES

WS-BPEL 2.0, henceforth referenced as BPEL 2.0, is considerably different from the previous V1.1 draft release. The article will address these changes and demonstrate how to attempt to migrate a V1.1 business process to be compatible with a BPEL 2.0 engine. Sometimes this migration is simple and can be accomplished by means of syntactic changes to the process; sometimes the migration is not so easy, and mostly results in the rewrite of the process or process fragment. We will start with the simple cases and move toward the more complicated ones.

It is not the intention of this article to explain BPEL 1.1 or to explain the new features of BPEL 2.0, so it is highly recommended that the reader have a good familiarity with the BPEL language.

BPEL 1.1 Features That No Longer Exist

We will first address those features that have been removed from BPEL 1.1.

The concept of "partner" is no longer available for BPEL 2.0. A "partner" groups several "partnerLinks," and in doing so represents a common endpoint. Aside from being descriptive, the "partner" concept did not have any executable property, so it was decided that the language did not need this concept.

The XML element "compensationHandler" and the XML attribute "enableInstanceCompensation" in the top-level "process" element have been removed. Instance (process) level compensation handlers never had any mechanism for being invoked; therefore, because they could not be used instance level compensation handling is no longer supported.

Since it is very unlikely that any BPEL 1.1 engine made use of either of these concepts, it is generally safe enough to just remove them from the process definition when migrating to BPEL 2.0.

Syntactic Changes

The following changes are just syntactic. You can simply do a simple find-and-replace to migrate to BPEL 2.0:

- Replace the XML attribute "variableAccessSerializable" with "isolated"
- Replace the XML tag "terminate" with "exit"
- Replace the XML attribute "onMessage" of event handlers with "onEvent"
- Move the XML attribute "joinCondition" that is present in BPEL activities to be a child element of "targets," as in the following XML fragment:

```
<invoke name="settleTrade">
    <targets>
        <joinCondition>
            $buyToSettle and
```

Mindreef® SOAPscope®

Take Control

of your

Web Services

Developers • Testers • Operations • Support

Web services and SOA have changed the rules with the introduction of a new XML abstraction layer.

Though most development organizations have tools and processes for managing code objects, they have little or no help for testing, diagnosing, or supporting the XML portion of their Web services.

Mindreef SOAPscope completes your Web services development toolkit by combining XML-aware tools for developers, testers, support, and operations. This flexible combination gives you the right tool for any diagnostic task, whether working alone or collaborating with other team members.

Start taking control of your Web services by downloading a FREE trial version of Mindreef SOAPscope today!

Mindreef SOAPscope: Named best Web services development tool in the 2005 InfoWorld Technology of the Year awards

Test *No coding required!*
Test service contracts and underlying service code with an easy-to-use, forms-based interface

Diagnose *No more wading through angle brackets!*
Find the cause of a problem with powerful message collection, analysis, and diagnostics tools

Collaborate *No more emailing XML snippets!*
Share test or problem data with others, regardless of their roles or system requirements

Support *No more finger pointing!*
Web services require support at the API level – a real burden for most organizations. Mindreef SOAPscope makes it easy by allowing customers and support staff to share Mindreef package files that contain complete problem data

Download a free version of Mindreef SOAPscope at www.Mindreef.com/tryout

```
$sellToSettle
        </joinCondition>
        <target
linkName="buyToSettle"/>
    </targets>
</invoke>
```

- Replace the XML attribute value "rendezvous" of the attribute "initiate" with the attribute value "join"
- The schema type "tRole" no longer has a child element representing the port type; instead, the port type is now specified as an attribute directly in the role itself, as demonstrated in the following fragment:

```
<plnk:partnerLinkType
name="shippingLT">
    <plnk:role name="shippingServic
e"          portType="shippingSer
vicePT"/>
</plnk:partnerLinkType>
```

- The attribute "portType" of messaging activities such as "receive," "invoke," "reply," "pick," and "onEvent" is no longer mandatory and can be omitted
- The URI used to specify XPath 1.0 as the expression/query language of choice has been changed, so replace the attribute value "http://www.w3.org/TR/1999/REC-xpath-19991116" of the attributes "expressionLanguage" and "queryLanguage" with the attribute value "urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"

In addition to these, the syntax for assignments has been changed, but we will discuss this in a separate section.

Extensibility of Expression/Query Languages

In BPEL 1.1 expressions are used by switch conditions, while conditions and assignments such as XPath expressions are constrained as being an XML attribute value. Although this is not generally a problem for XPath 1.0 expressions, it is awkward for more complex languages such as XPath 2.0 or XQuery 1.0. XML attributes do not provide enough "real estate" for complicated expressions and also do not allow for the use of other XML features such as CDATA, or to write XML itself as the expression.

Hence, to allow for better extensibility of BPEL using external languages, the authoring of expressions and queries are now realized within XML tags (elements) instead of attributes.

In practice this means that the XML attributes "for," "until," "joinCondition," "transitionCondition," "expression," "query," and "condition" must all be changed to be XML elements, which would then contain the expressions (the expressions are the former attribute values). The following snippet shows an example of this conversion for XPath 1.0. Note that the attribute "expressionLanguage" is optional.

```
<while>
    <condition
 expressionLanguage="urn:oasis:names:tc:
wsbpel:2.0:sublang:xpath1.0">
        $itemsShipped &lt; bpws:get
VariableProperty('shipRequest','sns:
itemsTotal')
    </condition>
    <sequence>
        <!— do something -->
    </sequence>
</while>
```

Listing 1 shows an example of a nonstandard usage of XQuery 1.0 as the expression language. BPEL 2.0 has thus far only standardized the usage of XPath 1.0.

Links

Links are used to specify synchronization dependencies between nested activities within a flow. In BPEL 1.1, links could not cross the boundary of structured activities such as "while," "isolated scope," "event handler," and "compensation handler." In BPEL 2.0, this restriction has been made stronger. Links that create a reentrant control path in scopes are no longer permitted. The reason for this tightening is to simplify the semantic of compensation handling. Figure 1 illustrates this banned scenario.

Messaging

Several aspects of messaging for BPEL 1.1 are unspecified. For example, there is no defined behavior for a process that receives a message for a request-response operation and finishes without replying. In BPEL 2.0, such a scenario would have trigged a new BPEL standard fault called "missingReply" fault.

Another example has to do with event handlers. In BPEL 1.1 global event handlers are enabled as soon as the process starts, but the specification does not go into details to define exactly what that entails. Would an event handler instance be executed concurrently with the execution of the start activity that created the process instance? If so, what if the event handler instance is accessing some variable that would have been initialized by the start activity? All of these issues are left for the vendor implementation to answer. In contrast, BPEL 2.0 explicitly states that the global event handler is only enabled after the start activity has completed its

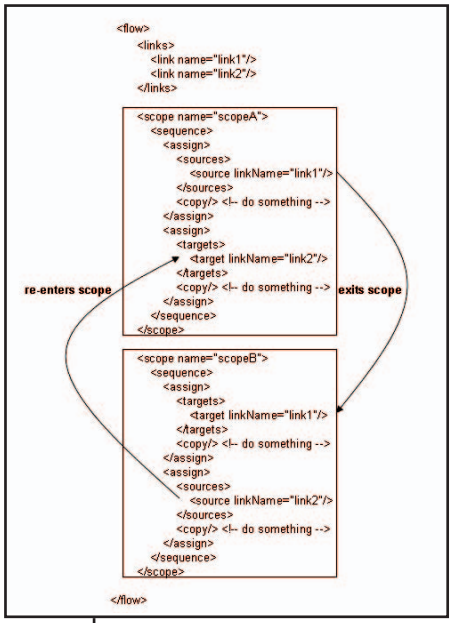


FIGURE 1 **Illegal scenario – reentrant control path**

execution, thus decreasing any chances for race conditions. However, what if there are multiple event handler instances? Do they operate on the same input variable? In BPEL 2.0, an "onEvent" activity declares its own input variable, hence each event handler instance works with its own local copy of the input data. The following snippet provides an example of a BPEL 2.0 event handler declaration:

```
<onEvent partnerLink="eventLink"
operation="event1" variable="inputVar"
messageType="tns:eventMessage">
    <correlations>
        <correlation set="id"
initiate="no"/>
    </correlations>
```

```
    <sequence>
        <!—do something -->
    </sequence>
</onEvent>
```

BPEL does not allow a process instance to have multiple outstanding replies for the same request-response operation. This restriction prevents concurrent event handler instances for the same request-response operations from being supported. BPEL 2.0 solves this problem by allowing "correlationSets" to be declared in an "onEvent," thereby disambiguating the event handler instances. However, because this is a common user scenario, it is expected that vendors solved this specification shortcoming of BPEL 1.1 by providing their own vendor-specific mechanism that will likely differ from the BPEL 2.0 solution.

There is no automatic way of migrating a business process to address these issues. People will have to understand how the vendor-specific behavior of the BPEL 1.1 engine being used is different from the BPEL 2.0–compliant behavior and take the appropriate actions. For example, consider the issue of the new BPEL fault "missingReply." It may be that one vendor-specific behavior prior to BPEL 2.0 was to raise a vendor-specific fault, i.e., "vendor:myFault," in which case the migration consists of replacing the vendor-specific fault with the new BPEL standard fault "missingReply."

Compensation Handling

Compensation handling is one of the distinguishing features of BPEL. It allows business processes to support long-running activities.

In BPEL 1.1, a compensation handler is executed with a frozen snapshot of the BPEL variables made at the time that the compensation handler was installed. In contrast, compensation handlers in BPEL 2.0 work off live data; that is, they use the current value of the BPEL variables. One way of mitigating this difference by guaranteeing that the BPEL variables are not changed is to declare them local to the scope that may be compensated.

Compensation handlers declared within isolated scopes are not themselves isolated in BPEL 1.1. This is changed in BPEL 2.0: compensation handlers within isolated scopes also have isolated behavior. Note however that the scope and the compensation handler do not share the same isolation domain.

In BPEL 2.0, attempts to invoke the same compensation handler more than once result in a no-op operation, as a replacement for raising the BPEL standard fault "repeatedCompensation," which has been removed. This makes sense because it seems unlikely that a business process would have a need to handle such a fault.

Data Manipulation

Data manipulation has changed significantly in BPEL 2.0. There are two parts to understanding these changes. First, BPEL 2.0 defines a data model to represent the BPEL variables. Second, BPEL 2.0 defines rules for mapping this data model to the different languages, such as XPath 1.0.

The BPEL 2.0 data model is based upon XML infoset. Each BPEL variable is conceptually a separate set of XML documents. In the case of BPEL variables of WSDL message types, each WSDL message part is mapped into a separate XML document of the set.

BPEL 2.0 data model mapping to XPath 1.0 is simplified, the BPEL XPath extended functions "getLinkStatus()" and "getVariableData()" were removed, and propagation of application data is done using XPath variables.

Let's look at the first case. Instead of using "getLinkStatus(linkName)" to access the status of a link, one can just reference the link directly as an XPath variable, that is, by adding the "$" character to the link name, as shown in the following fragment:

```
<joinCondition>
    $buyToSettle
</joinCondition>
```

The same idea applies to accessing BPEL variables. Instead of using "getVariableData(varName, partName)," one would reference the variable by using the "$varName" convention. However, in case of variables of WSDL messages, a part is referenced by appending the "." character and the part name to the variable name, as illustrated in the following snippet:

```
<assign>
  <copy>
      <from variable="value"/>
      <to>$outputVar.value</to>
  </copy>
</assign>
```

Queries on property aliases also make use of the same concept. In this case, a BPEL predefined variable named "source" must be used to reference the BPEL variable in question, as demonstrated below:

```
<bpws:propertyAlias propertyName="tns:
phoneNumber"    messageType="tns:out-
putMessage">
    <bpws:query>$source.value</bpws:
query>
</bpws:propertyAlias>
```

In BPEL 1.1 it is not clear what the result of "getVariableData()" is. Is it an XML document fragment? Is it just character data? BPEL 2.0 tightens the definition by specifying that the XPath variable referencing a BPEL variable resolves to the document element of the appropriate XML document. Thus it becomes clear what can or cannot be accessed in the resulting document. For example, you are able to access the XML attributes and child elements of a BPEL variable using XPath 1.0 expressions, however you will not be able to navigate to a parent node to try to access a different WSDL message part using XPath 1.0, since we have seen that each WSDL message part is contained in a separate XML document. There is one caveat to this rule: when used in the context of an expression (in contrast to being used in the context of a query), BPEL variables of simple type return the simple type content itself (character data) instead of the document element. This is done to simplify common usage scenarios. Listing 2 shows XPath 1.0 expressions being used to access multiple BPEL variables of different Schema types.

Finally, the BPEL "assign" construct itself has changed. Aside from the changes to the "expression" and "query" attributes that we have seen previously, literal assignments must now be authored within a "literal" element, as shown below:

```
<assign>
  <copy>
      <from>
          <literal>
              <addr:phoneNumber
                type="work">
                  <addr:country
                    Code>1</addr:
                    countryCode>
```

```
                    <addr:areaCode>408</
                     addr:areaCode>
                    <addr:number>570-
                     8000</addr:number>
                </addr:phoneNumber>
            </literal>
        </from>
        <to variable="phoneNumber" />
        </copy>
</assign>
```

Conclusion

The issues that have been presented here are not final. Specifically, issues related to data manipulation are still undergoing changes in the BPEL 2.0 specification to be released. Neither is this article exhaustive in its listings of migration issues.

Furthermore, not all issues are likely to be addressed and corrected in time for the upcoming release. It is expected that issues, such as filtering of EPRs, inclusion of Business Process Transaction Management (BTM) constructions, and accessibility to transport (i.e., SOAP) headers will be handled in a future version of the specification.

For all but the simplest business processes, migration from BPEL 1.1 to BPEL 2.0 is not an easy task. Some of the syntactic changes can be automated as shown in this article, however the semantic differences, especially when dealing with links, messaging, compensation handling, and data manipulation, will demand a comprehensive and time-consuming process.

Nonetheless, BPEL 2.0 is a huge step forward from BPEL 1.1. Whereas BPEL 1.1 provided a framework for BPM, BPEL 2.0 is moving towards allowing for the portability of execution of business processes across vendors, and as such, is an important hallmark in the industry.

References

- *Business Process Execution Language for Web Services V1.1:* http://dev2dev.bea.com/webservices/BPEL4WS.html
- *Web Services Business Process Execution Language:* www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
- *XML Information Set:* www.w3.org/TR/xml-infoset/
- *XML Path Language (XPath) Version 1.0:* www.w3.org/TR/xpath
- *XQuery 1.0: An XML Query Language:* www.w3.org/TR/xquery/

About the Author

Alexandre Alves currently works at BEA Systems, in the WebLogic Integration group. He has worked with integration technologies for over 10 years, focusing on CORBA, J2EE, and Web services.

aalves@bea.com

Listing 1

```
<assign>
    <copy>
        <from expressionLanguage="urn:bea:wli:bpm:
          wsbpel:2.0:sublang:xquery1.0">
            <![CDATA[
                <pons:simplePO>
                {
                    for $s in (1,2,3)
                    return
                    <pons:Item>
                        <pons:id>{$s}</pons:id>
                        <pons:quantity>100</pons:
                          quantity>
                    </pons:Item>
                }
                </pons:simplePO>
            ]]>
        </from>
        <to>$respVar.PO</to>
     </copy>
</assign>
```

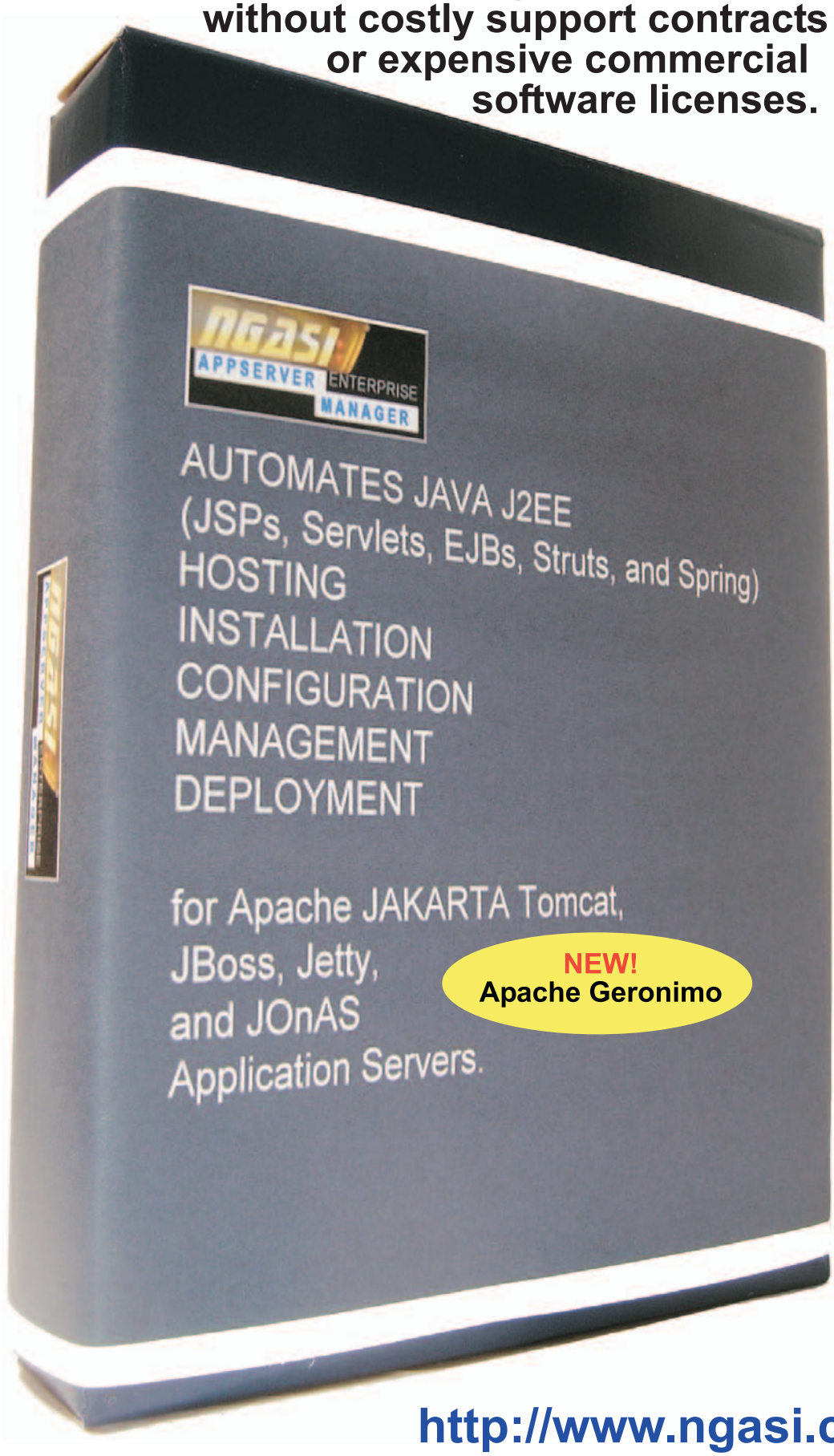
Listing 2

```
<!—BPEL 2.0 process definition -->
<variables>
    <variable name="booleanVar" type="xsd:boolean"/>
    <variable name="contactInfoVar" element="tns:
      contactDetails"/>
    <variable name="addressVar" element="tns:address"/>
</variables>

<sequence>
    <switch>
        <case>
            <condition>$booleanVar</condition>
            <assign>
                <copy>
  <from>$contactInfoVar/tns:address</from>
  <to>$addressVar</to>
                </copy>
            </assign>
        </case>
    </switch>
</sequence>

<!—Referenced Schema types -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/
XMLSchema" elementFormDefault="qualified"
    targetNamespace="uri:sample:assign" xmlns="uri:
     sample:assign">
    <xsd:complexType name="tPhoneNumber">
        <xsd:sequence>
            <xsd:element name="countryCode" type="xsd:
              string"/>
            <xsd:element name="areaCode" type="xsd:
              string"/>
            <xsd:element name="number" type="xsd:
              string"/>
        </xsd:sequence>
        <xsd:attribute name="type" type="xsd:string"
          use="required"/>
    </xsd:complexType>
    <xsd:complexType name="tAddress">
        <xsd:sequence>
            <xsd:element name="addressLine1"
type="xsd:string"/>
            <xsd:element name="city" type="xsd:
              string"/>
            <xsd:element name="state" type="xsd:
              string"/>
            <xsd:element name="zip" type="xsd:
              string"/>
            <xsd:element name="country" type="xsd:
              string"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="tContactDetails">
        <xsd:sequence>
            <xsd:element name="address"
              type="tAddress"/>
            <xsd:element name="phoneNumber"
              type="tPhoneNumber"
maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="address" type="tAddress"/>
    <xsd:element name="contactDetails"
type="tContactDetails"/>
</xsd:schema>
```

JAVA J2EE Hosting Automation Software without costly support contracts or expensive commercial software licenses.



FREE DOWNLOAD

nGASI APPSERVER ENTERPRISE MANAGER

http://www.ngasi.com/jdj.jsp

1.866.256.7973

An Architect's Guide to DSLs

Adding domain-specific languages to applications

You are an architect on your company's new flagship application. The app encompasses several business and technical domains that are, in your opinion, well suited to domain-specific languages (DSLs). In years past, you would have turned to XML as the solution for all your DSL needs. With or without schemas, you would specify configuration files and scripts in files foul with angle brackets. Thankfully, for the discerning .NET architect, DSL implementation options are now plentiful.

WRITTEN BY
DAN MASSEY

For each DSL you create, you need to identify the target user for that DSL. Is she a non-developer subject matter expert, a particular kind of developer, or an end user? This choice will drive the kinds of tools you make available to support the DSL. It will also influence choices such as whether your DSL is interpreted or when it is compiled.

Before selecting DSL implementation strategies or tools, you need to decide the application layer in which each DSL will live. Some are intended to support run-time configuration and initialization. Others may be part of well-defined and reasonably confined components. Still other DSLs might be used to script together your application at the macro scenario level. Even particular idiomatic coding styles are essentially DSLs. Martin Fowler calls these "internal DSLs." These choices influence other architectural decisions. We'll begin our examination of the implication of some of these decisions with a brief description of DSLs in general. Then we'll look at different techniques for including DSLs in a larger architecture.

DSLs have several advantages over general-purpose programming languages (GPLs). DSLs directly support high-level concepts from the business domain with program constructs (services, classes, interfaces, and libraries, whatever is appropriate to the target technology and the business requirement). The language takes responsibility for transforming, compiling, or interpreting those business concepts into a working system. When the business domain and technology expertise are encapsulated in a DSL, users of the language boost productivity while also improving quality.

It's useful to point out that the boundaries between GPLs and DSLs are fuzzy, if they exist at all. For example, C was originally created for the purpose of system programming. Perl was originally created for text processing (which is still its best domain). Java was an environment meant for set top boxes and then Web applications. Lisp was originally created for AI research. Now it's used in an operating system masquerading as a text editor. Even COBOL and Fortran are DSLs, although not Business DSLs (BDSLs). Many people say DSL when they mean BDSL, where business is used in the same sense as business logic. For purposes of this article, I would like to include business domain and non-business DSLs. In many IT shops, the non-business DSLs outnumber the BDSLs.

Whether business-specific or not, each DSL should provide the ability to say useful things over a well-defined domain. This implies the ability to describe a domain and the ability to specify the grammar and semantics of statements or expressions ranging over the selected domain. A BDSL should be created only where an organization has a clearly defined domain with well-understood semantics, and then it should mimic the normal forms of communication used when discussing the domain. From an architectural perspective, what we're really thinking about is providing points in an application architecture where domain concepts are expressed in more natural forms.

When it comes to including DSLs in an architecture, how you implement a DSL and where it fits in your architecture are closely related. There are many approaches to implementing DSLs, from literate (sometimes called intentional, although there are other meanings for intentional) programming to fully

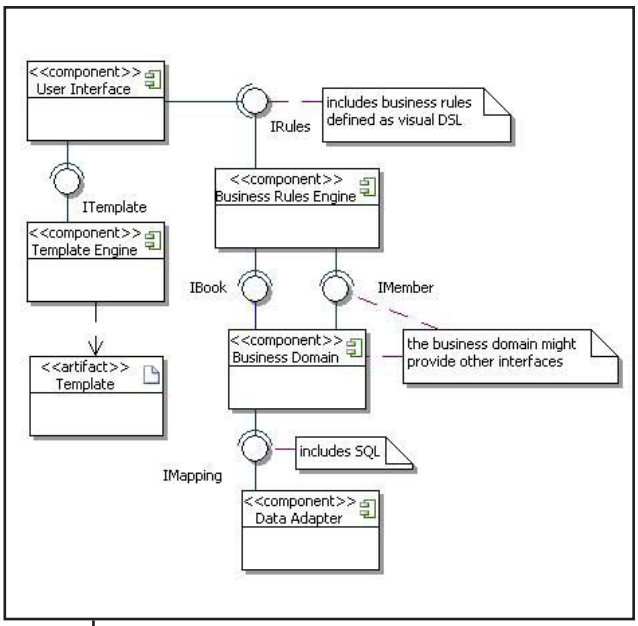


FIGURE 1 **DSLs are often part of component interfaces**

> When the business domain and technology expertise are encapsulated in a DSL, users of the language boost productivity while also improving quality

compiled custom languages. We'll start with the most prominent DSL creation tool in the .NET sphere and work down toward simple code-style implementations.

A typical enterprise component architecture like the one shown in Figure 1 presents many opportunities for serving domain expert users with DSLs. Any one of the components might include a DSL for configuration purposes or as part of its public interface. In addition, a DSL could be used to wire together or orchestrate interactions between multiple components in a single module or subsystem.

In September, Microsoft released another CTP of the Domain-Specific Language Tools (http://lab.msdn.microsoft.com/teamsystem/workshop/dsltools/default.aspx). The final release of the DSL Tools, planned to come on the heels of the Visual Studio 2005 RTM, will provide everything necessary to create visual domain language editors that can be used to generate code or for other DSL-related purposes. Microsoft provides tutorials for the current DSL Tools release. I won't rehash them here. The key to understanding the role Microsoft's DSL Tools play in architecture is that they give you the ability to create graphical editors for DSLs. The graphical notation maps to some other form, most commonly a textual representation. In this sense, the DSL Tools can be applied in the exact same way as textual DSLs.

One approach to including DSLs in an architecture is to use modular design to encapsulate DSLs in specific pieces of an application. A component or module might be configured by or execute logic provided by a DSL. Common examples of DSLs of this type are template engines, database access layers (using SQL as a language), and rules engines. In this approach, the DSL is part of the public API of the component. DSLs used in this fashion are often formally defined or documented textual languages. This is the approach commonly described for external DSLs. The architectural challenges are relatively simple in this case. Each DSL is implemented in a well-defined component.

The example in Figure 1 includes a business rules engine that interprets business rules defined in a visual or graphical DSL. The Data Adapter's interface gives dependent components the ability to provide SQL for specific operations not supported by the underlying mapping technology. Finally, the Template Engine uses templates stored in the file system. The templates may be defined using XML, a graphical editor, or some other notation. The purpose for using DSLs in each of these components is to provide a more natural or efficient expression of logic in the domains – in this case business rules, data access, and templates – than is provided by a GPL like C# or VB.Net.

When providing documentation for a component that uses one or more DSLs, it is good practice to provide, along with standard interface documentation, a guide to the DSLs supported with several nontrivial examples. DSLs may provide a better way of communicating or capturing ideas in particular domains, but they can add to the surface area and complexity of components. It's important to weigh the benefits of creating and implementing a DSL against any additional learning curve that comes with it. Microsoft's DSL Tools will make language construction more accessible for application developers, but they won't eliminate the need to properly document DSLs.

There are other approaches to DSLs that involve using the internal capabilities of a language or platform to support a DSL-like experience for developers. UML provides one way

Business DSL "Rules"

If you're contemplating creating a BDSL for one of your applications, here are some tips that might help:

- **The OED Rule:** Don't create a new language, describe or formalize an existing language.
- **The Standards Rule:** If standard languages are already available in the domain, use one of them.
- **The Domain Experts Rule:** If you aren't a domain expert, find one to help construct the language. If nothing else, she will help you test it.
- **The Prototype and Iteration Rule:** Prototyping and iterating your DSL are both good ideas. They are even better ideas when combined with the presence of one or more domain experts.

```
<?xml version='1.0?>
<r:rule xmlns:r="http://bad_examples.com/rules.xsd" id="r5"
     name="The XML Rule">
  XML is human readable, not human friendly.
</r:rule>
```

- **The Documentation and Examples Rule:** A DSL is a product. It should come with typical product support materials.
- **The "Do Something With It" Rule:** Compile it. Interpret it. Transform it. You go to the trouble of creating a DSL so that you have a formal language that you can use for these purposes.

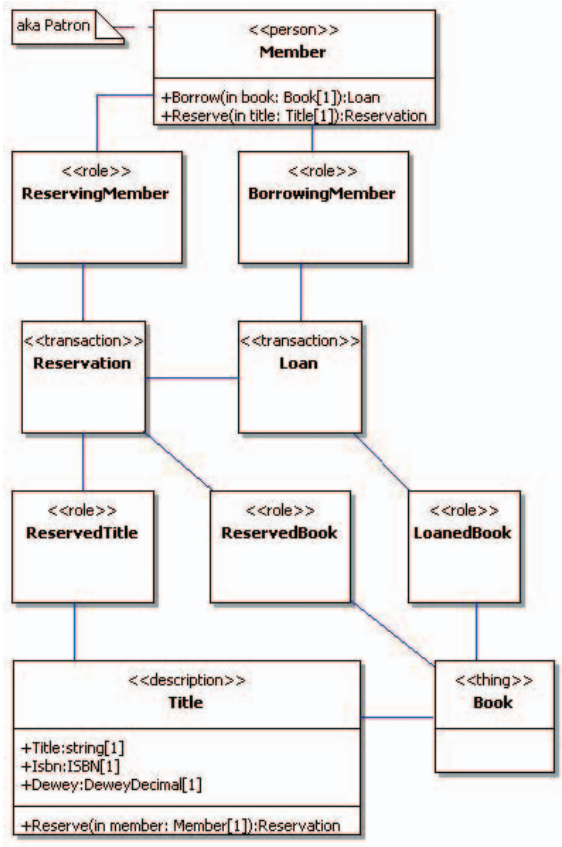


FIGURE 2 **A partial domain diagram for a library**

to define an internal DSL. A domain model such as the one shown in Figure 2 describes a business domain in a formal notation. The domain class diagram captures part of one possible description of the business domain of libraries. For space reasons, it includes only those elements that are critical to the examples. The approach to the domain model class diagram in Figure 2 is based on color modeling (see the first entry in the References section). As with the other topics addressed in this article, there are several good choices when it comes to approaches to domain modeling (see second and third entries). As you will see from the references, the topic of domain modeling has been around for a long time. Microsoft's DSL Tools, Model Driven Architecture (MDA) solutions such as Borland's Together Architect 2006, and the emergence of what Martin Fowler has called "language workbenches" are generating new interest in domain modeling and DSLs (see the fourth entry).

Let's take a walk through this domain fragment. A library, not shown, can include any number of books that are the physical copies of some number of titles, e.g., Jared Diamond's *Collapse*. In the part of the domain shown a member can reserve titles and books and borrow books. When a member calls the library to reserve a book, if a copy of that book is available, someone from the library staff, also not shown in the diagram, places the book on the "Reserved" shelf. If a copy of the book is not available, then the reservation is made for the title instead. When one of the books comes in that is the copy of a reserved title, it is immediately put in the "Reservation" shelf. A member can borrow a book either by walking into the library and pulling it off the shelf or by asking someone from the library staff to retrieve the book she reserved from the "Reservation" shelf.

If implemented following the rules of the domain modeling approach, this domain model class diagram is the basis for an internal library DSL, a library DSL implemented using standard .NET constructs and expressed in one of the available .NET languages. Typically, this will be done using the basic design shown in Figure 3. The user interface is responsible for capturing user actions and for displaying results. The domain is an implementation of a domain model like our library example. The scenario layer contains code to implement the actions kicked off from the user interface. For example, each scenario might be a controller in Microsoft's User Interface Process (UIP) Application Block (http://msdn.microsoft.com/practices/compcat/default.aspx?pull=/library/en-us/dnpag/html/uipab.asp).

In our example checking out a book is an easy scenario to express in C#, assuming this code is in some controller class.

```
public Loan CheckOut( Member member, Book
book )
{
 return member.Borrow( book );
}
```

This is a literate style of programming, and is easy for developers, testers, and stakeholders to discuss and to validate. It's almost as plain as writing, "The member borrows the book." There are some assumptions built into this scenario, for example the book is available to be borrowed. That might make sense if I have set up my domain so that the application can only get instances of Book when I know I have the physical book, perhaps because its serial number barcode has been scanned. Even in the case where behavior may be different depending on whether or not a book is available, I can rely on my domain objects to do the work. The internal DLS approach allows me to keep the scenarios simple and straightforward.

```
public Reservation Reserve( Member member,
Title title )
{
 Return member.Reserve( title );
}
```

Even within the domain objects, this approach to applying domain concepts can make code more accessible. In this example, Member's Reserve operation is also easy to read.

```
public Reservation Reserve( Title title)
{
 return title.Reserve( this );
}
```

The Title is responsible for deciding whether or not it has a book available to reserve. The domain model allows us to construct a set of object handshakes to create the required

IDC
Analyze the Future

Service-Oriented Architecture Forum

Enabling IT and Business Alignment

JANUARY 25, 2006 • MILLENNIUM BROADWAY HOTEL
NEW YORK, NY • WWW.IDC.COM/SOA2006

The only SOA Forum delivering a true peer-to-peer learning experience

This one day collaborative forum is designed for senior IT and business executives trying to understand and implement the strategic and technical aspects of SOA for their organization.

SPECIAL FEATURES:

- Participate in **private one-to-one meetings with an IDC analyst onsite**, providing insight and essential guidance into your current SOA strategy.
- Access **groundbreaking research** on SOA.
- Collaborate with your peers in **facilitated luncheon discussions** around common SOA interests, challenges and solutions.
- Opportunity to **answer key questions** around topics including: *Creating an SOA Strategy; Designing and Implementing SOA; Overcoming Technological and Organizational Challenges; Legacy Systems and SOA; Security Challenges; Extending Your SOA to Partners; and Aligning IT and Business*, to name a few.

Featuring an exceptional line-up of end-user case studies and presentations from leading organizations and industry experts, including:

- **James Barry**, VP, Application Development, Payroll, ***ADP***
- **James McKenney**, ISO, ***Security Bank of Kansas City***
- **Scott Metzger**, Chief Technology Officer, ***TrueLink***
- **Todd Richmond**, VP, Enterprise Architecture, ***Sabre Holdings***
- **Ruchir Rogrigues**, Executive Director, Strategic Systems and Development, ***Verizon***
- **Stuart Sackman**, Senior VP, Product Strategy, Employer Services Group, ***ADP***
- **Jaime Sguerra**, Second VP & Chief Architect, ***The Guardian Life***

SPECIAL OFFER! $295 REGISTRATION RATE (A $600 SAVINGS!)

Call 800-605-5849 to register, and refer to code WBSJ-05 to receive this rate
Register Online now: www.idc.com/soa2006 or call 800-605-5849 (+1 978-597-0133 - outside the U.S.)

For sponsorship information, please contact Jess Black, 508-988-7995 or jrblack@idc.com.

Diamond Sponsor

IBM

Sponsors:

Blue Titan
sonic SOFTWARE

Media and Association Partners:

ADTmag.com
ES Enterprise Systems
NetworkWorld
SupplyDemandChain
THE WALL STREET JOURNAL. PRINT & ONLINE
WebServices JOURNAL
WITI build. empower. inspire.

relationships in a way that is easy to follow. One alternative would be to create a ReservationManager class that is responsible for making all of these decisions and creating the relationships in a single operation. Instead of a title knowing what it means to request a reservation for it under various conditions, that logic is somewhere else. Maintaining that logic outside of the domain classes makes it difficult to make simple code statements like those above.

Splitting out your domain and defining it in a way that supports simple domain expressions is an architectural decision that pays benefits in reusability, testability, and maintainability. This approach to design can be combined with modules or components that use more conventional DSLs. For instance, a template engine that embeds C# or VB.NET expressions will be much easier to read if those expressions are written against a domain model like the one described here.

There are other ways for architects to create internal DSLs using the facilities of .NET languages. One interesting option is the use of C# 2.0's Generics capabilities. Generics are too large a topic to cover here, but I will give one example of applying them to the DSL problem. I want to build an extensible painting application. End users or third-party developers should be able to add plug-ins that can create new painting tools for my palette. There are a lot of ways to solve this problem, including various design patterns and configuration DSLs. Generics also provide an interesting solution.

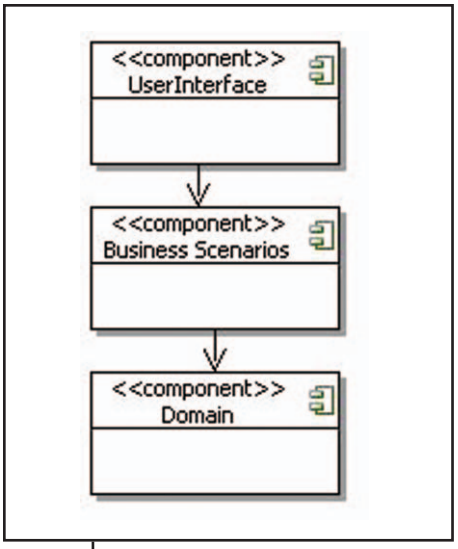


FIGURE 3 | **Application layering for DSLs**

> "Whether business-specific or not, each DSL should provide the ability to say useful things over a well-defined domain"

Tools in this program have a lot in common. I wouldn't want every person who adds a tool to the palette to have to implement the Tool functionality. I would also like to avoid forcing Tool authors to subclass Tool. I can use Generics to create a very little DSL to solve this problem. Generics are great for this purpose. I can define a tool to be:

```
public class Tool<Tip, Color>
{
```

If a developer wants to add a plug-in with a blue airbrush, he can create new classes that implement Tip and Color and compose new tools like this: *new Tool<AirBrush, Blue>*. Depending on the design of the application, the new plug-in might have an operation to build the new tool in a simple way.

```
public Tool GetColoredTool(Type colorType)
{
 return new Tool<AirBrush, colorType>;
}
```

This design allows us to compose tools without having to make each new piece aware of the others. An airbrush could just as easily work with the Plaid or Gradient color types. This construct is one reason why I prefer C# Generics to the way Generics are implemented in Java. Erasure prevents this use of Generics from working in Java.

Conclusion

DSLs, whether external or internal, are powerful tools that can improve productivity, communication, and maintainability. Different kinds of DSLs are appropriate for different points in an application architecture. Architects can encapsulate DSLs in components or modules. They can use domain modeling techniques to build literate internal DSLs. They can also use programming language constructs to create what are essentially small DSLs. Each approach is appropriate for solving different problems for the end application. Domain modeling is usually used to create a domain layer that acts as a platform for easy-to-read scenario implementation. Small DSLs based on language features are usually used to solve little technical problems. External DSLs are usually used to bring some other form of expression into the application.

If you want to start working with these approaches, I recommend:

- Downloading Microsoft's DSL Tools
- Reading one of the domain modeling books and trying the techniques in a UML modeling tool
- Learning about Generics in C# 2.0

References

- Coad, P., Lefebvre, E., and De Luca, J. *Java Modeling in Color with UML: Enterprise Components and Process*. Prentice Hall PTR, 1999. (Note: This book has much more to say about domain modeling than about Java. It is applicable to any OO environment.)
- Nicola, J., Mayfield, M., and Abney, M. *Streamlined Object Modeling: Patterns, Rules, and Implementation*. Prentice Hall PTR, 2001.
- Taylor, D. A. *Business Engineering with Object Technology*. Wiley, 1995.
- Fowler, M. "Language Workbenches: The Killer-App for Domain Specific Languages?" 2005. www.martinfowler.com/articles/languageWorkbench.html

About the Author

Dan Massey is chief technical architect at Borland, where he works on Borland's Software Delivery Optimization (SDO) solutions. Prior to joining Borland, Dan was a TogetherSoft mentor and JEE architect and developer.

dan.massey@borland.com

ENGAGE AND EXPLORE...

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

CALL FOR PAPERS NOW OPEN!

SOA WebServices Edge 10th International Edge 06 conference+expo

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

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! www2.sys-con.com/events

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: jimh@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 201 802-3066

Produced by SYS-CON EVENTS

Ten Things Your EII Vendor May Not Tell You

The hidden complexities of implementing an EII solution

Enterprise information integration (EII) is getting a lot of hype these days, and the vendors are giving you very compelling reasons for why you need an enterprise data access layer based on their EII product. However, a lot of the scenarios and case studies they present are very targeted or are simple examples of how their product works that don't delve into the complexities of a real-world environment. This article presents some of those complexities and demonstrates how some of the EII products may not provide adequate functionality for an environment with such complexities.

WRITTEN BY
TIEU LUU

1."It's probably a lot harder to implement an EII solution in your enterprise than our examples and case studies would tell you."

The quality of an enterprise's data is usually much worse than what someone normally expects. Things such as incomplete entries in a database, invalid values, and data inconsistencies are rampant. Also, within large enterprises it is quite common to find that the same information is stored redundantly in multiple systems. When you try to provide centralized access to your data through an EII solution, these issues will only make implementing the solution that much more difficult. For example, when you have the same data stored in multiple sources and you're trying to map the entities defined in your EII solution's composite view, which source do you map to? Even if you don't have data stored redundantly across multiple sources, just trying to map an entity in the composite view that spans multiple sources may not be as easy as it seems. In a single database, you have foreign key relationships that link together related records. There is no such linking mechanism across different database instances and you have to rely on unique identification attributes that aren't always consistent across different databases. An EII product is not going to be able to help you too much with these types of problems.

2. "We isolate you from all of the complexities of accessing multiple data sources… including the ability to debug."

No doubt writing code that needs to access multiple data sources is not easy, so it's nice to have something that handles this for you. However, this can be a double-edged sword. Hiding the complexities of accessing multiple data sources also hides the causes when you have problems. When you are debugging, you want the ability to be exposed to all the gory details. When an EII product receives a query, it breaks it down into smaller queries that are sent to the constituent data sources. To ensure that you can adequately debug your data access layer, make sure your EII product allows you to trace through the execution of the queries to each constituent source.

3."Be prepared to lose some platform-specific functionality and flexibility."

Traditionally when you directly access

a database you can take advantage of that particular RDBMS's native functionality and extensions. For example, one particular database vendor has an extension to JDBC called a ROWID type that allows you to directly access the unique identifier of a row in a table. With the ROWID type, you can directly specify the row in your query to improve performance. Since an EII product has to provide a common interface to multiple platforms, it cannot include such platform-specific mechanisms in this interface. So just keep in mind that if you use an EII solution, you may lose the ability to use some of those platform-specific features that you've grown to love.

4. "Sure we support transactions, but only if they go through a single data source."

A usable data access layer must allow you do more than just read data. You must also be able to create, update, and delete data through it. A single update operation from the client application's perspective may in fact update multiple sources. That's the whole point in having this data access layer – it isolates the client applications from having to deal with the complexities of interfacing with multiple data sources. What this means is that the EII product must be able to coordinate these operations across the multiple sources as a single transaction. This is the classic problem of distributed transaction management. This is not a new problem – products such as transaction processing monitors have solved it, and there are specifications out there such as XA that define a standard way to handle it. Although it's not a new problem, it is still not an easy one to solve and not all EII products are built on top of transaction-processing platforms. Complicating this matter is that not all databases and their corresponding drivers support distributed transactions in a standard manner, if at all. Throw into the mix other data sources that EII vendors claim to be able to support, such as Web services, and this can get pretty complicated. My advice here is to make sure you understand what type of distributed transaction support your enterprise will need and go with the EII vendors that have proven experience and expertise in building transaction management products.

5."Of course we support write operations, but you may not actually be able to use them."

When you create a data access layer with an EII product, you often define a composite view of an entity with attributes of that entity coming from multiple source systems. This creates complexities when you try to execute a write operation on that composite view (in addition to the distributed transaction issues mentioned earlier). For example, suppose you've created a composite view of Customer that uses a customer address from an order management system and customer name and birth-date from a sales system. Now you have another application that tries to create a new instance of Customer through this data access layer and you supply all of the necessary information that is defined in the composite view of Customer. However, the order management system and sales system may contain other attributes on Customer that are not defined in the composite view, so that data is not going to be available. If those other attributes are critical to the functionality of the order management and sales systems, they may have "not null" constraints set on their corresponding columns in the databases. In this scenario, the create operation will fail when it is propagated to the order management and sales systems. It will not be uncommon to find these and other types of referential integrity constraints in your source systems that can complicate what may seem like a simple write operation. The bottom line is that there may be complex relationships in your enterprise data that may make it impractical to perform write operations through an EII layer that hides these application-specific relationships.

6. "Our product can be clustered, but that doesn't guarantee you'll have proper backups."

When you use an EII product to implement an enterprise data access layer, one of the things you'll need to be concerned about is how to ensure availability, since all of your applications are now dependent on it. Most EII products offer some type of clustering solution to address availability; however, the sophistication of the clustering solution will vary greatly from vendor to vendor. In a clustered deployment, when you have backup instances running, the challenge is how to migrate state to the backup instance when the main one goes down. In a typical application, we generally think of state as the contextual information that is maintained across multiple client requests. The typical usage scenario for an EII product won't generally require support for maintaining this type of stateful information. That is usually managed in the application logic layer that is calling the EII solution. However, to support distributed transactions the EII product will have to maintain state to keep track of the steps in the execution of a transaction so that it knows when to commit or rollback the distributed data sources. In a clustered setup, if the instance that is coordinating this transaction fails, that information needs to be migrated to the backup instance so that it can correctly complete the execution of this transaction. Support for these types of scenarios is where you start to see the varying levels of sophistication in each vendor's clustering solution. As mentioned earlier, most of them will offer clustering, but not all of them can transparently migrate state from one instance to the other. If you operate in a high capacity environment with transactions that span multiple sources, this kind of functionality is critical to ensure the integrity of your data. When you're evaluating EII products, make sure you ask the vendors about these kinds of details in their clustering solutions.

7."Tuning this thing for performance can be a nightmare."

Different applications have different data access and usage patterns. Some applications may produce a lot of transactions but may only access a small amount of data in each transaction, while in another application the transaction throughput may be small but the volume of data that is accessed is very large. The ways you'd tune a product for these two types of applications are very different. When you use an EII solution to provide centralized access to your enterprise data sources, you have to accommodate all of the different access and usage patterns of the applications that will be integrated with this EII solution. Tuning your infrastructure to support a single application's performance requirements is tricky enough. Trying to tune it to adequately support multiple patterns of usage and access will be even

harder. Frequently there will be conflicting configurations – something that optimizes the performance of one application will degrade the performance of another. My advice here is to make sure that you understand the access and usage patterns of the applications that will be integrated with the EII solution and ask the vendors if they can adequately support these patterns in the same deployment. Of course, they will say it depends on what kind of performance requirements you have, so make sure you have well-defined performance criteria for each of those scenarios. Finally, don't just take their words for it, plan for enough time to performance test your EII solution with simulations that reflect the access and usage patterns that are common in your environment.

> "Just keep in mind that if you use an EII solution, you may lose the ability to use some of those platform-specific features that you've grown to love"

8. "Better make sure you set aside some time and money to build the proper security infrastructure if you really want to use this thing."

When you implement an EII solution, the main security challenge will be moving from an environment where you have distributed security management, i.e., each data source maintaining its own list of users, roles, and privileges, to one where all of that is centrally managed. Since you are now providing centralized access to all of these distributed sources, the security needs to be managed at this central point as well. Providing centralized security management is not a problem unto itself – it's actually a good thing. What will be difficult is trying to provide this centralized management while each data source that you're tapping into still maintains its own security model. Chances are you're not going to be able to change the source systems to follow this new centralized security model. What this means is that you will need the tools and infrastructure to map multiple security models from the source systems to the centralized model. This issue is not specific to EII solutions: any large-scale integration project will have to deal with this. The problem is that not all EII products come with the type of security infrastructure that is needed to adequately handle these issues. Most of the EII products out there come with the security infrastructures that are typical of most databases and app servers – you set up user profiles, roles, and privileges that specify what objects can be accessed and so forth. Some of them will even let you tap into third-party user profile stores such as LDAP, but none of this allows you to effectively mediate from the separate security models of the source systems to the central model of the EII solution. It's most likely that this will be left up to your developers with "plug-ins." Of course there are many products from the security vendors that handle these types of things with single sign-on, federated identity management, and SAML, but you will have to integrate them into your EII solution and that is not a trivial task.

9. "Say goodbye to the concurrent access behavior that you've been used to."

Transaction isolation levels on databases allow you to control the level of locking when the database is being concurrently accessed. The four isolation levels defined by the ANSI SQL92 standard (see the References section), in ascending order of restrictiveness, are:

- **READ UNCOMMITTED:** The transaction can read uncommitted data, i.e., data being changed by another transaction concurrently.
- **READ COMMITTED:** This level results in the prevention of a transaction from reading uncommitted changes in other concurrent transactions.
- **REPEATABLE READ:** In addition to the prevention associated with READ COMMITTED, this level ensures that reading the same data multiple times will receive the same value even if another transaction modifies the data.
- **SERIALIZABLE:** The transaction has exclusive read and update privileges to data by locking it; other transactions can neither write nor read the same data. It is the most restrictive transaction isolation level and it ensures that if a query retrieves a result set based on a predicate condition and another transaction inserts data that satisfy the predicate condition, reexecution of the query will return the same result set.

Because not all data sources support a standard range of transaction isolation levels, things can get tricky when you try to integrate them together with an EII solution. For example, suppose you've defined a composite view of an entity that spans across two data sources. On the EII layer, you set an isolation level of REPEATABLE READ for accessing this entity so this level has to be propagated to the two sources. However, suppose one of the data sources only supports two isolation levels, READ COMMITTED and SERIALIZABLE. In most cases, to ensure data integrity the EII product will map to the next highest level, which in this case will be SERIALIZABLE, meaning that you now have a more restrictive locking level. So now when you thought you would be getting an isolation level of REPEATABLE READ, you are actually getting an isolation level of SERIALIZABLE. The risk with this is that at higher, more restrictive isolation levels, deadlocks are more likely. If you had coded your application expecting an isolation level of REPEATABLE READ, but you are instead getting an isolation level of SERIALIZABLE, your code could deadlock and you would have a heck of time trying to figure out why. Therefore when you're evaluating EII products, make sure you ask the vendors what kind of isolation levels their product supports and how they map to those of the underlying data sources.

10. "Unlike ETL products, we offer a federated approach that doesn't store any data, sort of…"

The main difference between an EII product versus an ETL (extract, transform, load) product is that the EII product does not actually store data and instead directly accesses the source systems for reads or writes. One of the big advantages of this federated approach is that you don't need to maintain a central store that needs to be synchronized with the source systems – a common issue with ETL. In a high throughput environment, getting the synchronizations perfectly timed is very difficult, which can lead to data integrity problems. Thus, a lot of the EII vendors will tell you that using their product will get rid of the data integrity issues that are common to ETL approaches. However, sometimes there is a fine line between an EII product and an ETL product. For performance reasons, a lot of EII products will use what's known as materialized views or persistent caches, which means they will actually store the data in a local database or file system. If they're storing some data locally, you could potentially have the same data synchronization issues that plague ETL solutions. The concept of a federated approach is nice, but it's not always practical from a performance perspective, as the EII vendors are beginning to see. So if you're thinking about using an EII solution because you have data synchronization problems with your current ETL setup and you have strict performance requirements, you'd better think twice.

Summary

The concept of an enterprise data access layer with an EII product that allows a single point of access to distributed data sources is very attractive. If you are considering implementing an EII solution, the thing to keep in mind is that it is very likely that your enterprise will suffer from one or more of the problems described here and it will complicate your implementation of such a solution. Therefore it's important for you to have a detailed understanding of your current data environment, i.e., quality of the data and infrastructure, as well as a well-thought-out plan for how your existing and new applications will use this solution. This will help you to better understand what kind of features you'll need from the EII product, which will help you do a better product evaluation, or even decide that such a solution is not feasible. Finally, one last point, to be fair: not all of the complexities described here exist in everyone's environment, and some of the products out there do have some features that allow you to handle some of these complexities quite well.

References

- Baksi, Dibyendu. "J2EE Transaction Frameworks: Distributed Transaction Primer." May 23, 2001. www.onjava.com/pub/a/onjava/2001/05/23/j2ee.html?page=1

About the Author

Tieu Luu is an associate with Booz Allen Hamilton where he designs enterprise data and integration architectures for large federal agencies. Prior to Booz Allen Hamilton, Tieu held lead engineering positions at companies such as Grand Central Communications, Mercator Software, and Aether Systems, where he worked on the development of integration and mobile computing platforms.

luu_tieu@bah.com

SOA Security

The only thing we have to fear is fear itself

"Let me assert my firm belief that the only thing we have to fear is fear itself – nameless, unreasoning, unjustified terror, which paralyzes needed efforts to convert retreat into advance."

– Franklin Delano Roosevelt
First inaugural address, March 4, 1933

WRITTEN BY
JEREMY EPSTEIN

As organizations move to service-oriented architecture (SOA), security becomes one of the key concerns impacting deployment. After all, a company's most sensitive information is frequently stored in the business systems that are now being accessed by the Web services employed within an SOA. As such, security concerns have become part of the enterprise decision-making process relating to the adoption of a SOA. However, these discussions are often exclusively focused on the security features of the Web services implementation, and give little consideration to the inherent security of the Web services platform, or of the services themselves. As Gary McGraw, CTO of Cigital, likes to say, "Software security is not secure software."

Historically, most applications routinely highlight their primary security features as a key selling point. However, outside of the security realm, few actually attest to the security of the application itself. Thus, users may possess all of the security features in the world, but still remain insecure.

These challenges are accelerated by the move to an SOA, which allows these potential vulnerabilities to be more widely exposed as Web services. In this scenario a variety of standards such as SSL, WS-Security, and SAML, often take the place of the product's previously referenced security features, but the results are the same as users remain insecure because the services themselves were not well written.

As the director of product security for webMethods, I speak frequently with both IT generalists and security specialists within hundreds of large companies and government agencies in the US and abroad. These discussions cover a wide range of security-related topics, but most frequently, they relate to security features and standards supported by specific products. To my surprise and consternation, after more than five years of such discussions, I've only had a handful of individuals ask, "So how do you know the product is secure?" Furthermore, even fewer had any idea as to what the right answer to this question would look like.

This point was driven home during a recent presentation I attended at a local security users group, where the speaker was comparing half a dozen XML firewall/gateway products that were to be used in a government SOA project. He explained the procurement criteria used by his company, which included features, cost, ease of maintenance, financial stability of the vendor, and other criteria. Not on his list was "Is it secure," and he seemed genuinely surprised when I asked about this missing question. However perhaps in their environment, this was a reasonable omission.

Having spent some time pondering this question – *why* so few people ask whether products are secure – I've actually taken the liberty of assembling a list of 13 potential responses.

1. ***People assume the vendor takes care of it.*** When buying a new car, I don't ask about the engineering processes used in the design; I assume Ford or Toyota knows more about how to design cars than I do. Why should the purchaser of software be responsible for asking how secure it is?
2. ***They don't know that they should ask.*** Some IT organizations (even in large companies) lack a dedicated internal security staff; instead, security is one aspect of everyone's job. No one person has enough background to know what to ask, or how to make sense of the answers.
3. ***They don't know what to ask for.*** Sometimes users know that they need security, but have no idea how to measure the results. With no Consumer Reports for software quality, how can a nonspecialist ask useful questions? As for specialists, what questions will help give them the comfort they need?
4. ***They're uncomfortable with the technology.*** Most security engineers I know feel comfortable with the bits and bytes of routers, firewalls, and operating systems, but few know much about the security of enterprise business applications or SOAs. Therefore they ask about the aspects they're familiar with – such as use of SSL – and ignore the harder questions such as "Is it secure?"
5. ***They've made a conscious risk assessment.*** It's impossible to get everything right, and some organizations make conscious decisions on where to focus their energies. Even if a Web service has a security flaw, the odds of those problems being detected and exploited are far lower than the odds of an attack through an unpatched router or Web server, simply because there are more people attacking the commodity products than the customer-specific Web service.
6. ***They think they're safe.*** Everyone has heard the fallacy "we're safe, we're behind the firewall." In many organizations, that's truly believed. Since Web services tend to be implemented by servers inside the organization, their security gets ignored, even though firewalls will simply pass along a Web services request, including any attack code.

7. ***They use vulnerability metrics.*** It's fairly easy to search databases of vulnerabilities (e.g., CVE or Bugtraq) to find out how many security problems have turned up in a given product, and how severe they are. Rather than asking the vendor, security engineers may use metrics such as the number and severity of publicly reported bugs to determine the quality of the product. However, these results may or may not tell the whole story.
8. ***They simply don't believe vendor claims are trustworthy.*** Vendors may intentionally or unintentionally give inaccurate results. For example, a vendor who performs penetration testing might not have tested the product or version being considered. Thus, users conclude that the value of the testing is minimal.
9. ***They have reduced security requirements in the POC.*** Frequently, security isn't a requirement in a proof of concept, and technical issues other than success of the POC are not revisited before the procurement decision gets made. Thus, the opportunity to consider the security of a product is missed.
10. ***They don't think it's their job.*** This could be a variant of "the vendor takes care of it," or it could be a symptom of an organization where the security specialists aren't responsible for the security of the systems in use.
11. ***They know that their organization doesn't care.*** The security specialist (if there is one) knows that he or she can only say "no" so many times, and only has a limited amount of influence over purchasing decisions. Why should he or she spend the time to question the vendor or analyze the security of a Web services application when it's unlikely to impact the buying decision? For that person, it's easier and better to use silver bullets to influence a critical piece of security infrastructure.
12. ***They think standards take care of the problem.*** Standards such as SSL (for Web servers), S/MIME (for e-mail), and WS-Security (in the Web services space) are widely perceived to provide security. Too many organizations fail to understand that while these standards are important, they don't actually secure the system. An implementation error in a product can leave a system that is completely standards compliant insecure.
13. ***They perform their own testing.*** To end this list on a positive note, some organizations don't ask the question because they're planning to come to their own conclusions by performing their own analysis and testing.

Despite the failure of users to ask, vendors are actually quite willing, able, and *eager* in many cases to provide and demonstrate the improved security in their products. In other words, there is adequate *supply*. The problem is that there is insufficient *demand*, at least as expressed in buying decisions.

So what can be done?

We should first consider *why* we don't ask the question of every vendor we interview. In many cases, the decision may be entirely reasonable. Whenever we look at a vendor, we should make an *explicit* decision whether the security of the product is important, and if not, document why not. For many organizations, the aforementioned list may be the right starting point.

For those vendors of whom we want and in fact *need* to know how secure the product is, we need to know what to ask, and how to measure the answers. Some of the measures by which a user might evaluate a software vendor are:

- Strong security involvement in architecture/design
- Good software engineering practices
- Security-focused QA
- Penetration testing
- Automated vulnerability testing
- Manual or automatic source code analysis
- Defect density prediction
- Training developers in security "best practices" (e.g., OWASP)
- Formal criteria-based assessments (e.g., BITS, Common Criteria)
- Using a development methodology, such as CLASP, that helps identify security problems before they occur
- Other third-party reviews

Unfortunately, there is no single answer to how much is enough. Should vendors be expected to meet all of these criteria? Should they be expected to meet most of them? How do we prioritize among competing claims? For example, how should we evaluate two months of security-focused QA in comparison with a week of automatic code analysis? Is a product that has undergone a BITS evaluation more secure than one for which all developers are trained in the OWASP top 10?

In reality, these questions are scarcely different from the other issues raised in the procurement cycle, such as the trade-off between cost and performance, with one exception: these are the critical criteria that are not typically assessed in a formal manner as part of the process.

It's important to remember that no evaluation process guarantees the success of a product; however, it does help to improve the odds while providing users with additional recourse should issues arise. For example, following a proof of concept, we can feel relatively confident that we've obtained the best performance, but not totally certain in this knowledge because the product has yet to be deployed in the field. By also extending the product's underlying security to this scrutiny, we can improve the likelihood that it will not expose a security breach within the enterprise. This approach will also provide users with greater insight into the product's overall security architecture so that proactive steps can be taken to remediate any uncovered shortcomings.

Summary

Ultimately, organizations building SOAs need to recognize that securing their Web services requires both a secure Web services platform *and* securing the Web services themselves. Purchasers of Web services platforms can and should ask the vendor about how they secure their platform. Also, developers of Web services on those platforms need to take an equal responsibility to introduce rigor in their design and testing, so that the Web services do not become the attack point.

By asking Web services vendors the question "how do you know your product is secure," organizations will raise the bar for security, and thereby protect their information. We must not be afraid of complex answers, and by doing so, we will prove that "The only thing we have to fear is fear itself."

About the Author

Jeremy Epstein is the senior director of product security for webMethods, Inc.

jepstein@webMethods.com

Business Process Orchestration

BPEL supports time-critical decision making

Our journey began with our Department of Defense research projects when we saw an opportunity to solve our data management challenges with XML (www.sys-con.com/story/?storyid=40411). The journey continued with the evolution of that work as it applied to exposing legacy data sources as XML through data-oriented Web services (www.sys-con.com/story/?storyid=45527). We continue to build upon this foundation by broadening our service-oriented architecture (SOA) with these XML-enabled yet disjoint systems as we look to Business Process Execution Language (BPEL) to orchestrate their complex interactions.

WRITTEN BY
MATTHEW ZAGER

My primary focus over the last year has shifted from enabling legacy data sources and applications as services to orchestrating the business processes that encapsulate those services. The motivation behind this work is to bring automation and repeatability to decision-support environments, thereby allowing the decision makers to focus on the big picture. People naturally associate BPEL with business or corporate environments where time is money; however, there is a poignant case to be made for leveraging this technology within government agencies where return on investment is measured in human life rather than in dollars. I'll present examples that demonstrate how applying XML and Web service technologies coupled with BPEL enhances the ability of decision makers to quickly assess and respond to situations with time-critical requirements. Also included is an example of US Naval support in humanitarian aid and disaster relief operations such as the undertakings following the tsunami in Southeast Asia and the hurricanes in the Gulf Coast.

Brief Review of Previous Work

When we first began developing with XML we initially set out to achieve the separation of content from presentation. We published our data as XML using our XML Data Services engine and used XSL transforms to generate our portal-enabled user interfaces. It soon became apparent that the real value in what we were able to do with our data sources would be realized when the raw information rather than the visualization could be shared. Out of Web services emerged defining standards that dramatically increase system interoperability and data sharing. The Web services effort to achieve interoperability using standards and a loosely coupled integration model allows flexible composition of systems in a variety of domains.

Business Process Execution Language

Systems integration is composed of much more than data sharing or the ability to carry out simple exchanges between services. Web services conform to a stateless model of synchronous exchanges or uncorrelated asynchronous conversations that can be invoked using an interface de-

scribed in a standard format, namely Web Services Description Language (WSDL). BPEL receives the full benefit of past research and development through broad vendor participation, but also builds on that foundation by embracing emerging standards and has become the industry standard for Web service orchestration. SOAs become viable cross-domain integration platforms, given the ability to model complex interactions as well as a business protocol that supports synchronous and asynchronous, long-running, stateful, machine-machine processes. Figure 1 depicts the composition of discrete services from diverse systems participating in a complex business process.

In addition to its obvious technical merits, BPEL provides additional value by bringing the organization's business processes and procedures to the forefront during development. BPEL delivers the visibility into an organization's methods of operation and the business rules they practice, formally describing and documenting their business processes. This formality and transparency gives rise to process engineering, whereby an organization is able to focus on and validate business practices. As with the services BPEL orchestrates, the instantiations of these processes are discrete, reusable components capable of being composed into larger business processes or executed outside of the context of a larger application.

BPEL Supports Time-Critical Decision Making

Industry and government organizations alike are faced daily with time-critical situational assessments that have far-reaching implications. The Department of Defense is continuously engaged in operational exercises and scenario development. The vast experience and knowledge gained through these activities transition to operational procedures, commonly referred to as preplanned responses. Likewise, corporate scandals led to increasing regulatory compliance outlining standard operating and accounting practices. These well-established procedures and precisely defined responsibilities lend themselves to process automation.

First we'll look at a simplified industry example of business process automation to introduce the value of BPEL in time-critical decision making. The example I'll use is that of an Internet loan brokerage firm. This firm's job is to match borrowers with finan-

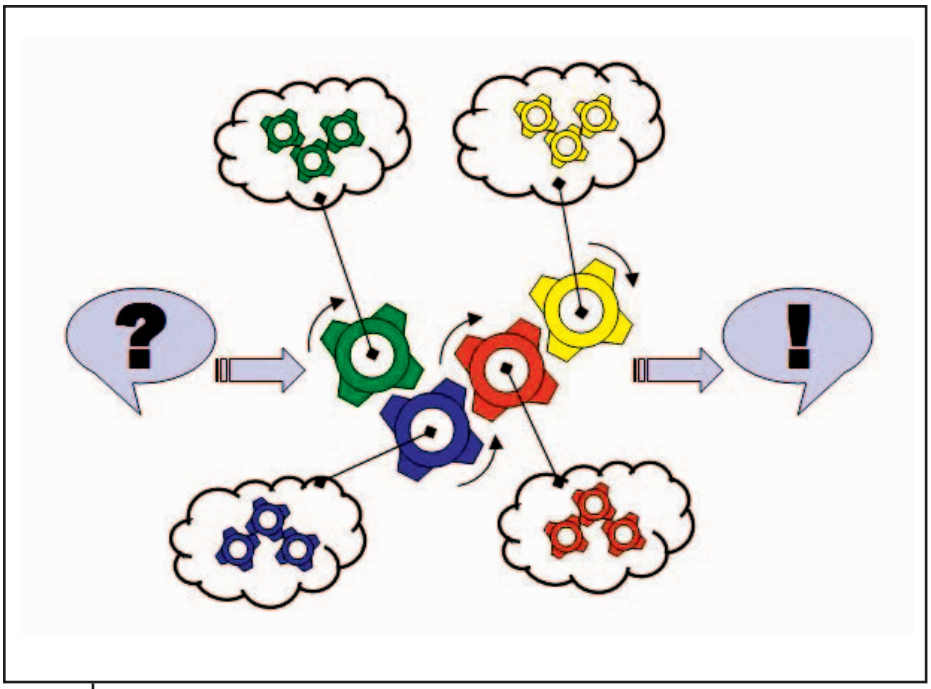


FIGURE 1 **Service composition business modeling**

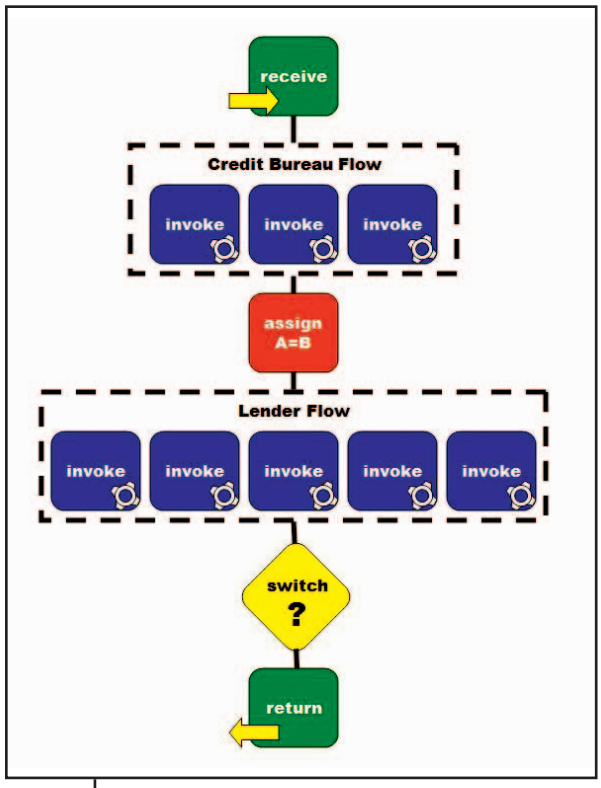


FIGURE 2 | Notional loan process

cial institutions willing to lend money to that individual for a prescribed amount and purpose. The broker's motivation for employing BPEL is to downsize the required workforce through automation while simultaneously increasing the number of borrowers they pair with lenders. By succeeding in doing so, they will lower the cost of doing business and increase income by generating more fee-based revenue. As shown in Figure 2, the broker would author a BPEL composition with partner links for the borrower, x number of credit reporting bureaus, and y number of lending institutions.

The process would be initiated by a borrower providing required information by completing an electronic loan application. By using that information the BPEL is able to asynchronously request the applicant's credit score from all credit bureaus in parallel. The credit score responses coupled with the applicant's information is then able to be sent in parallel to all of, or based on the applicant's input to only certain, lending institutions via their partner links. Each lender will have its own proprietary business rules to determine whether or not to offer a loan and what terms that loan will carry. The broker then asynchronously receives the responses from the lenders, presenting any offers to the borrower. Even in this simple flow, we are able to see multiple examples of time-critical assessments. The broker's efficiency is greatly increased by enabling the information collected from the borrower to be communicated in a machine-machine fashion rather than requiring a person to read, interpret, and route the input to a loan officer within the lending institution. That machine-machine communication is also done in parallel to the lending institutions, thus greatly increasing the number of opportunities to match the borrower with a lender in the same amount of time. The parallel flow additionally prevents deadlock of the process by removing the dependency of a response by one lender before submitting the request to other lenders. The lenders themselves reap the benefit of participating in the broker's flow by being exposed to more applicants. Moreover, if the partner link that the lender exposed to the broker were itself a BPEL, the lender would be able to increase the number of applications it could respond to through automation. Finally, building on loosely coupled services allows partners to change their proprietary back-end processing without affecting the business process. If a lender were to institute new business rules that qualify more borrowers, that module could be deployed behind the scenes without exposing the details to the broker's flow. This flexibility and the extensibility to easily add partner links to additional financial institutions positions the broker for future success.

One of the lesser-known missions of the US military is global humanitarian aid and disaster relief. The example below will focus on US Naval components where missions typically entail transporting and delivering

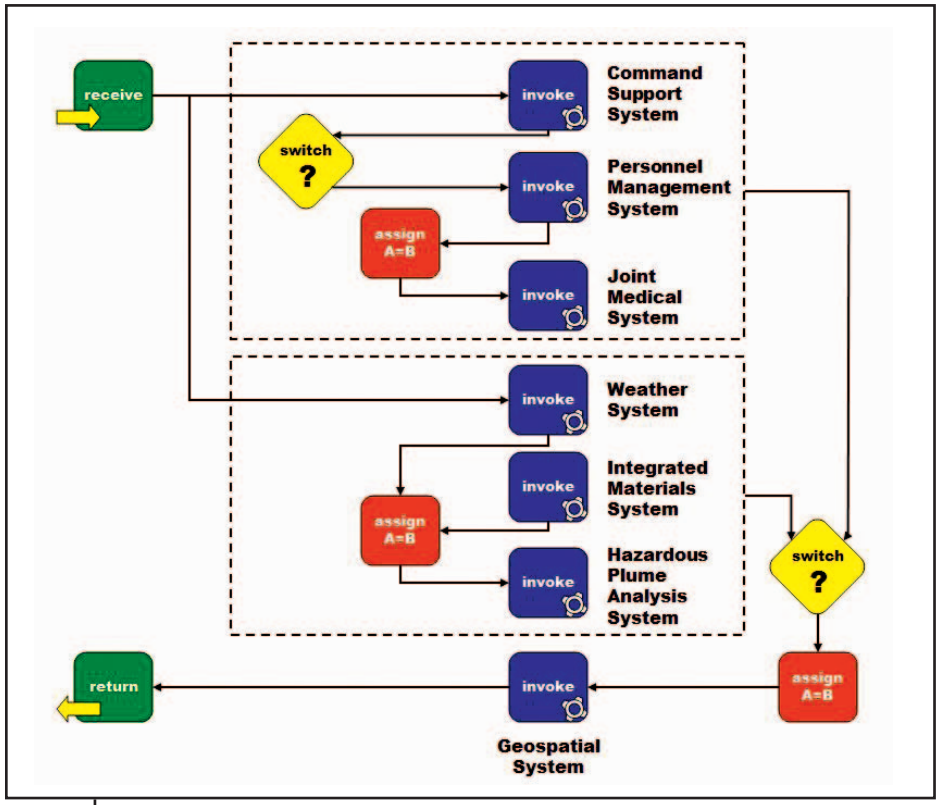


FIGURE 3 | Disaster relief process

tons of supplies, including food, water, and medicine, as well as hosting nongovernmental organizations such as the Red Cross and by serving as floating hospitals for some of the world's best physicians.

Figure 3 depicts the high-level flow between the following seven partner links: a Command Support System that manages facilities and supply status, a Personnel Management System that tracks among other things job descriptions and deployment locations, a Joint Medical System that maintains personnel medical records, a Weather System that provides current and forecast meteorological and oceanographic conditions, an Integrated Materials System that maintains locations and types of industrial materials, a Hazardous Plume Analysis System that supplies the capability to simulate effects of hazardous material releases, and finally, a Geospatial System partner link for map-based visualization and collaboration.

The example presented here is a preplanned response to a natural disaster requiring supplies and medical aid. The first step is to identify vessels that have operating rooms and infirmaries with occupancy below a specified threshold. This is accomplished by invoking the partner link to the Command Support System with appropriate constraint parameters.

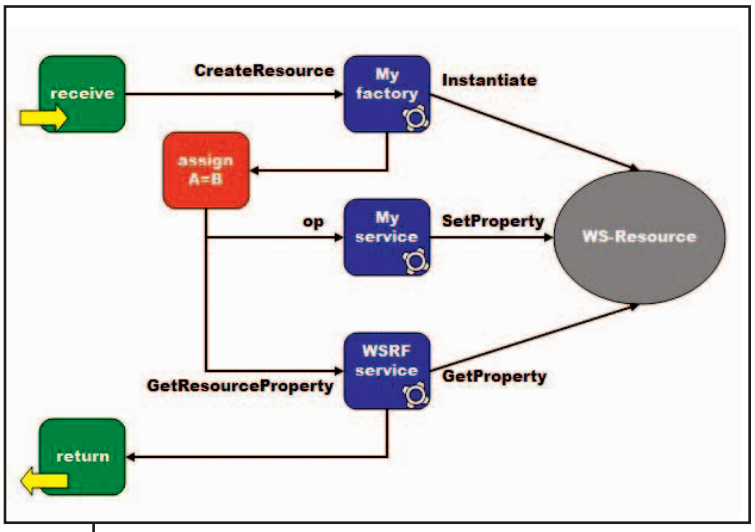


FIGURE 4 BPEL + WSRF process

The next step is to identify personnel with appropriate job classifications for providing medical attention to evacuees by invoking the Personnel Management System partner link using the list of ships returned by the previous step.

The third step is available in the event that the afflicted region is struck with disease in the wake of the disaster. The identified medical personnel list is then used as input to the Joint Medical System to isolate any personnel who may not have the necessary vaccinations and would be at risk by providing care.

An absolute constant in any mission package is evaluating the effects of weather. Weather has a vast number of impacts which can be computed using the parameters available from our Meteorological and Oceanographic partner links. The first of these impacts that needs to be considered for this type of operation is conditions en route and at the location where the ship is to be anchored. These conditions include sea state, wave height, and harbor depth, among several others. Additional conditions that need attention upon arrival at the destination include precipitation, visibility, wind speed, and direction, because these directly impact the ability of helicopters to take off and land, as well as the ability of transport vessels to unload the ship's cargo and bring it ashore.

Once suitable support vessels and routes have been identified, the final anchoring location will need to be a confirmed safe zone based on proximity of known hazardous materials and predicted weather effects. Two more partner links will be used in this step, the Integrated Materials System, which provides information about industrial material storage locations like large chemical or fuel depots, and the Hazardous Plume Analysis System, which can perform simulations of the spread of those materials in the event of a release. The weather data collected from the previous step in conjunction with the known hazardous material stores are provided as inputs to the Hazardous Plume Analysis partner link. The Hazardous Plume Analysis service performs calculations and returns information that establishes safe zones.

The final step is to package the results and send them to our final partner link, an Open Geospatial Consortium (OGC) Web Feature Service (WFS)–compliant visualization tool that can geospatially present the suitable vessels, the medical personnel and facilities status, traversable routes to the afflicted region, and safe anchoring locations outside the range of hazardous material plumes. By utilizing these technologies as force multipliers, we're able to improve the accuracy of the information provided to decision makers by executing mission planning on the order of hours rather than days.

The flexibility and reusability of BPEL is clearly demonstrated through the orchestrations that have been composed that are integrating these same partner links but targeting different operational requirements such as maritime-domain awareness, which centers on safety of international trade shipping and the global war on terrorism.

BPEL + WSRF

The Web services architecture has been broadly accepted as a means of structuring interactions among distributed software systems. However, discussions regarding additional standardization are required to facilitate further interoperability among services, particularly those concerning stateful resources. One effort in addressing these issues is the Web Services Resource Framework (WSRF). The WSRF defines the relationship between Web services and stateful resources in what is referred to as an implied resource pattern. This implied resource pattern is a set of conventions based on existing Web service standards, and chief among them is WS-Addressing. The WSRF also presents an approach for lifecycle management of the resource as well as for defining a standardized Web service interface for accessing the resource and its associated properties.

As I read about WSRF, I looked at leveraging this concept within the context of BPEL. Orchestration performed with BPEL is incredibly powerful for standardizing the linking of partners with private service implementations. BPEL is also often used to encapsulate concrete executable processes, frequently integrating services that operate on the same data sets yet reference, consume, and publish the data in very different ways. Process engineers who use this concept are able to think in terms of nouns and verbs, composing orchestrations that simply direct services to act upon specified resources in a consistent manner rather than translating disparate input and output schemas.

Common in WSRF is the use of a factory pattern for creating WS-Resources. The WS-Resource factory is itself exposed as a WSDL-described Web service. Figure 4 outlines the BPEL composed to demonstrate simple WSRF integration. The first step is to instantiate a new WS-Resource by invoking the resource factory's createResource method. Instantiating a WS-Resource entails creating a new stateful resource with an identity, and associating the new stateful resource with a Web service as outlined in the implied resource pattern. The next step is to assign the values of an input variable and configure the WS-Addressing fields with unique identifier information returned from the resource factory service. The most significant is the ResourceKey, which was returned by the resource factory identifying the newly created WS-Resource. Subsequently the BPEL invokes an operation on a resource-aware Web service, specifying the WS-Resource identifier to be used. In our case the simple operation simply updates the resource's stateful value with one passed as an input parameter. Last, we demonstrate calling one of the WSRF's standard methods for retrieving WS-Resource properties to verify that the stateful resource had in fact been updated.

I performed my experimentation using the Oracle BPEL engine and the Globus Toolkit as my WSRF implementation. WSRF currently requires the WS-Addressing resource identifying information to be passed in the SOAP header. Therefore, on the WSRF service invocations in the BPEL listing we did make use of the extension inputHeaderVariable. These same headers are normally handled transparently by the Globus Toolkit implementation of WSRF, so I've also included code listings for the modifications I made to the generated WSDLs to make the SOAP header values explicit and visible to the BPEL engine. I believe significant value would be realized if these specifications evolve with consideration to improved synergy between stateful processes and service-addressable stateful resources.

This experimentation makes a case for

information exploitation technologies, such as Web services and orchestration using BPEL, with an eye to the future and provisioning computing resources. Organizations electing to deploy more advanced automation solutions are able to increase the flexibility and efficiency of their information infrastructures that support their decision makers.

Conclusion

BPEL provides the standards-based, platform-neutral foundation to compose service interactions based on their message exchange behaviors while managing the myriad of complexities inherent with process-level systems integration. As such, government agencies and industries alike have leveraged this capability to assist in time-critical decision making. Web services, related standards, and architectural principles allow us to leverage concepts and technologies such as BPEL to develop effective and efficient decision-support environments.

Resources

- *Business Process Execution Language (BPEL):* www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
- *Web Services Resource Framework (WSRF):* www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf

Listings

Listings are available by viewing this article online in the ***WSJ*** archives at http://web-services.sys-con.com/read/issue/archives/ (Vol: 5, iss: 11).

BPEL
MyProcess.bpel – WSRF BPEL Example
MyFactoryService
MyFactory.wsdl – Globus generated WSDL
MyFactory_flattened.wsdl – Globus generated WSDL
MyFactory_bindings.wsdl – Globus generated WSDL
bpel_MyFactory_service.wsdl – Modified Globus generated WSDL
MyService
My.wsdl - Globus generated WSDL
bpel_My_flattened.wsdl – Modified Globus generated WSDL
bpel_My_bindings.wsdl – Modified Globus generated WSDL
bpel_My_service.wsdl – Modified Globus generated WSDL

About the Author

Matthew Zager is the chief engineer for the Center for Advanced Information Technology Division at SAIC, a research and engineering firm. He has over seven years of experience working on commercial and Department of Defense research projects.

zagerm@saic.com

WSJ Advertiser Index

Advertiser	URL	Phone	Page
Forum Systems	www.forumsys.com	801-313-4400	3
IBM	www.ibm.com/middleware/soa		5
IDC	www.ibm.com/middleware/soa	800-605-5549	19
Intermedia.net	www.intermedia.com	888-379-7729	6
IT Solutions Guide	www.sys-con.com	888-303-5252	35
KaPow Technologies	www.kapow.com	800-805-0823	Cover III
Mindreef	www.mindreef.com/tryout	603-465-2204	11
Parasoft	www.parasoft.com/products	888-305-0041	Cover II
Skyway Software	www.skywaysoftware.com	813-288-9355	Cover IV
Smart Data Processing, Inc.	www.weekendwithexperts.com	732-598-4027	33
SYS-CON Reprints	www.sys-con.com	888-303-5282	47
SYS-CON Website	www.sys-con.com	888-303-5282	39
WebAppCabaret	www.webappcabaret.com/jdj.jsp	866-256-7973	15
Web Services Edge	www2.sys-con.com/events	201-802-3066	21
Wily Technology	www.wilytech.com	888-303-5252	37
XML 2005	http://xmlconference.org	888-GET-WILY	9

Brokering Web Services... The Next Big Thing?

Write a service and get rich

Web services were created around the notion that it's easier to discover and leverage somebody else's service rather than write your own from scratch. Also, it is much easier to create applications made up of many services, thereby allowing change to occur at a pace faster than anything we've seen in the industry thus far.

WRITTEN BY
DAVID S. LINTHICUM

The idea of Web services was to create a standard interface, programming model, description language, and a directory that would allow this to happen in and among very different systems. Indeed, today you can leverage services across the Internet that are functionally equivalent to the services being hosted locally.

Taking this concept to the next level, we can build applications (composites) through the selection and use of these Web services. For instance, we have no need to write a logistics subsystem if one exists on a server someplace for you to leverage it. There is no need to write a risk analytics application; instead, leverage somebody else's work. You get the idea. This is clearly more a traditional computing concept than something new, thus saving a ton of time in the application development process and allowing businesses, large and small, to become more agile and have a much more cost effective IT. This is the promise of SOA.

Considering that we both understand the benefits of leveraging Web services and are willing to change our existing systems to support the exposure and leveraging of services, now what? The next step is brokering, or allowing consumers of services to find producers of services. There are a few instances of brokers today, including StrikeIron, Jamcracker, and SalCentral. Keep in mind that these brokers are also similar to directory and governance systems we are defining in SOAs today.

These brokers, as well as brokers yet to emerge, will provide a few key features to facilitate consumers finding producers, and the ability to monetize this interaction, such as:

- A directory service where the Web services can be found that contains a description of the service, owner, technology documentation, etc.
- An ability to charge for the service, either through a perpetual license, or a pay-per-drink kind of arrangement
- An ability to share reviews and other user information with other services users
- The ability to support a federated identity infrastructure

Thus, like monetized Web sites today, you're able to create a service, register it with a broker, and sit back and see the usage turn into fees for use. You can count on seeing many companies, such as the on-demand application service providers today, beginning to sell their Web services versus simple browser interfaces to applications. Clearly, Salesforce.com and Netsuite are moving in this direction. Moreover, we'll see smaller players, such as the "one guy and a dog" hit it big time as they create that killer service that everyone wants to leverage.

So, how do you prepare for this forthcoming market? Those who design and post services will have to understand a few basic principles:

1. Focus on granular services that are part of a holistic solution
2. Consider many service externalizations scenarios.
3. Track usage
4. Quality in the design

Focusing on granular services that are part of a holistic solution means that you consider the problem you're solving, as opposed to just the service you're implementing. Moreover, you're willing to provide many services that together will solve a business problem, but at that instance solve a tactical problem. For example, you're building a service to track overdue accounts, but you also need to consider how that works and plays with existing accounting applications, or other accounting services.

Considering many service externalizations scenarios means that you're building a service that may be externalized to humans or to other computer systems through a variety of interfaces. In essence you're interface-agnostic, understanding that the value of the service will need to be realized within a variety of systems, all having different looks and feels.

Track usage. Not to be too big brother, but it's nice to know who's using the service and where. This serves two purposes: first, it allows you to match up your income expectations with service usage. Second, it allows you to solve performance and availability issues before they become a larger problem. Remember, you're hosting this someplace, it's not delivered in a CD. You can build a tracking subsystem easily within services; make sure that those using the service understand that such tracking exists.

Quality in the design. If you're going to sell or rent service, you need to understand that the quality of that service needs to be impeccable. In essence you're becoming part of an application that's unknown to you, therefore you need to design that into the service as well as test the service, more so than any application. Not doing so means you'll be disruptive to those using your service, and your service won't add value; thus, it won't be used.

Go, make some money!

About the Author

David S. Linthicum is the author of three books on application integration and SOA, a frequent speaker at industry conferences, and the host of the "Service-Oriented Architecture Expert Podcast" (www.soaexpertpodcast.com).

linthicum@att.net

Reach Over 100,000

Enterprise Development Managers & Decision Makers with...

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

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

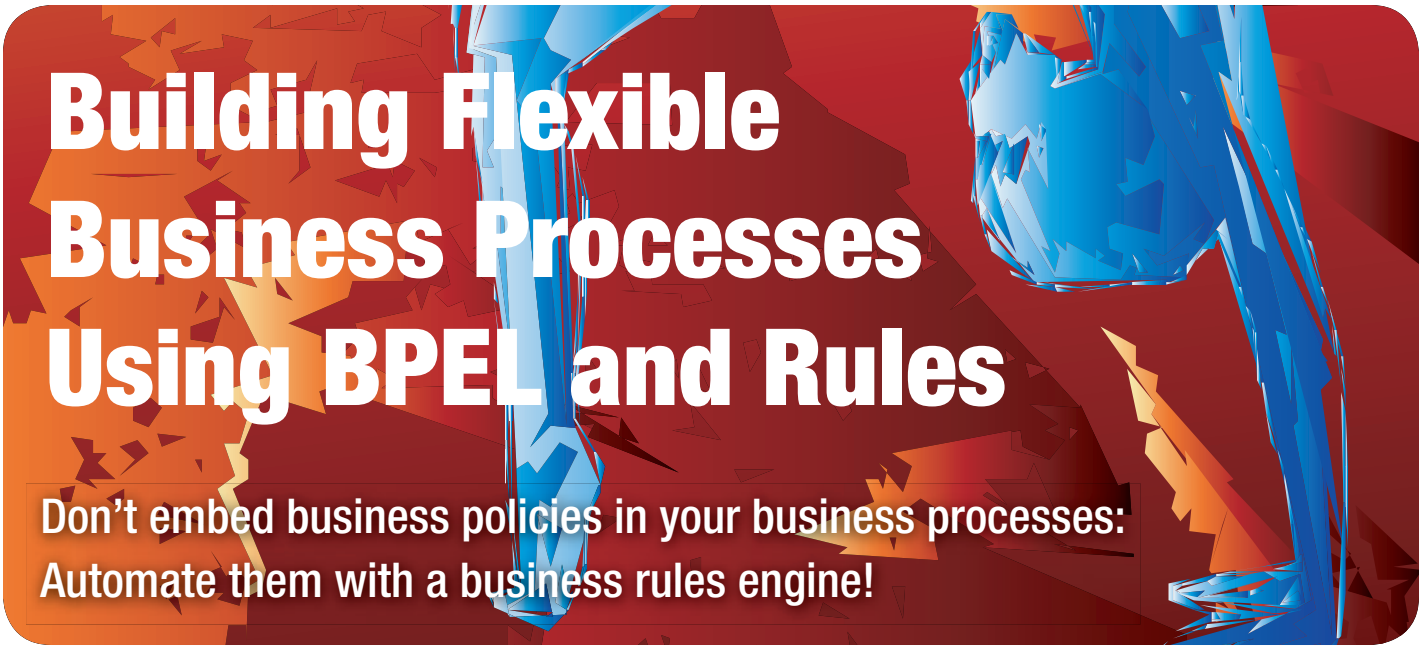
Get Listed as a Top 20* Solutions Provider

For Advertising Details Call 201 802-3021 Today!

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

SYS-CON MEDIA

The World's Leading i-Technology Publisher



Building Flexible Business Processes Using BPEL and Rules

Don't embed business policies in your business processes: Automate them with a business rules engine!

Leading companies are tackling the complexity of their application and IT environments with service-oriented architecture (SOA), which facilitates the development of enterprise applications as modular business services that can be easily integrated and reused, thereby creating a truly flexible, adaptable IT infrastructure. Business process management (BPM) solutions such as those based on Business Process Execution Language (BPEL) enable services to be orchestrated into business processes. Processes built using a BPM solution can be reused, changed easily in response to business requirements, and enable real-time process visibility.

WRITTEN BY
MOHAMAD AFSHAR & BHAGAT NAINANI

The promise of SOA is simplified integration, increased reuse, greater agility, and reduced risk. SOA and BPM deliver increased agility through three key enablers:

1. Reducing the time that it takes to automate business processes by reducing the gap between model and implementation, and enabling easier reuse of existing assets that can be exposed as services and then reused
2. Enabling business processes already implemented as orchestration of services to be changed rapidly
3. Freeing up funds for projects that enhance business agility by giving IT the ability to reduce spending on maintenance – after all, a capability that is implemented once, as a service, provides a single point of change and is easier to maintain when compared to the scenario in which it's embedded multiple times in different applications.

Agility is one of the biggest promises of BPM: the ability to make rapid changes to processes in step with the changes that occur inside of your business. Such changes are not always changes to the process. Often they are changes to the rules that drive the process. A typical business process often includes a number of decision points. These decision points generally have an effect on the process flow; for example, someone's credit rating may determine whether he or she is approved for a low-cost loan. These decisions are evaluated based on certain conditions and facts, which may be internal or external to the business process, and predefined company policies or rules. Business rules engines (BREs) allow architects to easily define, manage, and update the decision logic that directs enterprise applications from a single location without needing to write code or change the business processes calling them. BREs have been used extensively in enterprises; e.g., to implement yield management in the travel industry (what price to sell a ticket?), credit risk assessment in the loan industry (what is the interest rate for my car loan?), operations scheduling in manufacturing (what should we build today to maximize throughput and keep customers happy?), and the list goes on.

BREs are naturally of interest to enterprise architects building out SOAs, since they contribute to agility by enabling reduced time to automate, easier change, and easier maintenance for business policies and rules. BPM technology and BREs naturally fit together: BPM enables

automated and flexible business processes; BREs enable automated and flexible business policies.

We'll outline three different approaches that you can take to incorporate rules into your process logic: code-based, model-driven, and service-oriented. We consider two classes of BPM systems: monolithic BPM suites – those that embed capabilities including a BRE into a suite, and open-standards BPM solutions, which are based on the BPEL standard and enable you to use your choice of rules engine or an embedded one.

We show how each of two solution classes supports code-based, model-driven and service-oriented automation of business rules. A case study of a loan application processing will be outlined to show how business processes and rules exist together, and how the rules engine enables changes in business policies to be made easily by business analysts, without breaking the business process logic. We will then focus on how practitioners can go about building out their SOA using BPEL and their choice of rules engine, as well as how to integrate these capabilities (from an architectural perspective). We will also provide best practices on when to embed decisions in the process logic and when it's best to abstract and capture decisions/policies using a rules engine.

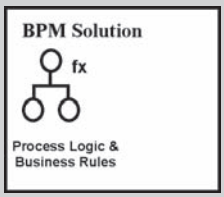
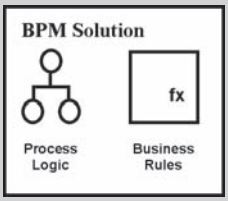
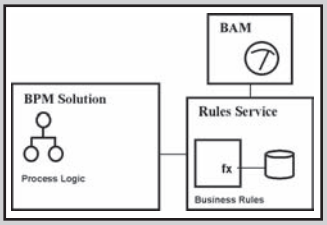
Code-Based		Decision logic and rules are embedded as part of the process logic itself using *Xpath* expression, simple *if/then* logic, or function calls.
Model-Driven		Rules capability is provided by the BPM solution, which also provides an integrated process and rules design time. The rules are separated from the process logic and can be changed independently without impacting them. It's easy to create and deploy rules because the rules within a process share the process context. Different processes in the BPM platform share rule sets.
Service-Oriented		An external/existing BRE is used. This approach is also model-driven; however, there is a centralized rules repository and policy management. Rules owners can define rules using their choice of tools and in the format they desire. Rules need to be deployed to the rules engine and then exposed as a decision service. Facts external to the BPM system can be asserted. This Web service allows rules to be used across processes and in other applications and tools, such as document workflow and business Activity monitoring (BAM).

TABLE 1 | **Approaches to rules enabling business processes**

Context

Let's provide some background about the various concepts in a rules engine. Some common terms are:

- **Facts** – Application or business objects that are used to make decisions. A fact definition can be created using the tools provided by the rules engine or directly imported from Java objects or XML document definitions. For example, a customer object, loan application object, and credit history are all facts that can be used to define credit risk rules.
- **Rules** – Sets of expressions that are evaluated as soon as facts are created or updated into the rules engine. The expressions can be simple if/then expressions, or decision tables, or complex functions. Rules also have actions associated with them – these actions fire if the condition evaluates to true. The actions may in turn create new facts or call other functions.
- **Constraints** – Used to define portions of the rules that are customizable by the business analysts after the rules have been deployed. For example, in a rule like "if loan amount < 100,000 AND credit rating > '700', then Auto_Approve," both the action and the number "700" could be modifiable after deployment, but the expression itself may not be changed.
- **Rules dictionary/vocabulary** – A collection of metadata files that includes the fact and rule definitions. This generally forms the deployment unit for all rules used by a specific application.

A typical rules engine has the following components:

- **Rules editor** – A tool used by business analysts and developers to define facts, author rules, and deploy rule sets.
- **Rules repository** – The persistence layer that stores and manages the rule sets and definitions. Most rules products support both file and DB-based repositories.
- **Rules engine** – Processing engine that evaluates and fires the rules based on facts that are created or updated in its working memory. Facts are generally loaded into the engine using application-programming interfaces (APIs) or batch loaders. Many rules engines on the market are based on the industry standard RETE algorithm for processing rules.
- **Rules engine interfaces** – Most rules engines support APIs to import facts definitions, define rules, assert/retract facts, execute rule sets, and retrieve results.

These components are common to both the monolithic BPM suites and the open-standards BPM solutions built on BPEL.

Approaches to Rules Enabling Business Processes

We consider three basic approaches to using rules in business processes, which are shown in Table 1. Regardless of the approach taken, some requirements include: tools for business analysts to define rules in an English-like language and tools for process owners to incorporate rules as part of the process flow; capabilities to modify rules after process deployment in response to changing company policies without the need to change the process definitions; mechanisms to load facts into the rules engine from business processes as well as other applications; and the ability to reuse rules across processes and appli-

Approach	Development		Reuse		Flexibility	
	Separation of concerns between process implementers and rules owners and customized tooling for each of these	Ease with which decision and rules logic can be leveraged in processes	Reuse of rules within the context of processes	Reuse of rules across applications (without the need to synchronize rules metadata across rules engines)	Ability to support and leverage existing rules engines	Ease with which rules in in-flight process instances can be changed
Code-Based	✓	✓✓✓✓	Not possible	Not possible	Not possible	Not possible
Model-Driven	✓✓✓	✓✓✓✓	✓✓✓✓	✓	✓	✓✓✓✓
Service-Oriented	✓✓✓✓	✓✓✓	✓✓✓✓	✓✓✓✓	✓✓✓✓	✓✓✓✓

TABLE 2 | **The merits of three approaches as they relate to development, reuse, and flexibility**

cations. We can evaluate each of three approaches with respect to common criteria regarding ease of development, reuse, and flexibility. Table 2 summarizes the merits of each approach.

Monolithic BPM suites support both the code-based and the model-driven approaches and work best when rules are used in the context of business process only and when rules do not require additional context (for example, real-time metrics from a Business Activity Monitoring [BAM] solution). They do not generally make it possible for you to leverage existing rules (metadata) that may reside in an existing rules engine, which means you have to build your rules base from scratch or implement logic to import the rules. Furthermore, there are usually no out-of-the-box mechanisms for synchronizing rules between the embedded rules engine in the BPM system and external rules engines and repositories.

Open standards–based BPM solutions, such as many of those based on BPEL, enable all three approaches: code-based, model-driven, and service-oriented. They allow you to use your choice of rules engine if you decide to go for the model-driven or the service-oriented approach. You can also leverage existing rules/rules repositories/rules engines for use in business processes. If you need to use facts external to the BPM solution, then you generally don't have to worry about synchronizing rules metadata if you take the service-oriented approach. However, if you don't, then you can use the integrated rules capabilities and benefit from the integrated design time for rules and processes.

Case Study

In the previous section we discussed various approaches for rules enabling your business processes. Some of the key criteria to keep in mind when deciding which approach to use are the design experience, flexibility, reuse, and ease of maintenance of rules. Let's illustrate the trade-offs of the various approaches using a case study.

Consider the example of a loan flow process that is deployed at a loan agency. The loan agency accepts a request from a client, performs a credit check with an external service, and then does an automatic approval or routes it to a manager for review. Depending on the outcome, the loan flow process notifies the customer. The scenario consists of the following participants: the customer, the loan agency's approval process, and the credit rating service. Various facts that may be used to make loan approval decisions include the loan amount, the customer's annual income, the customer's credit rating, the status at this agency (returning customer or new customer), and the agency's outstanding loans (after all, if the agency has approved many high risk loans recently, it's probably a good idea to take this into account in approving future loans). Now let's consider various decision points in the process:

1. The first step is to get the credit rating from the credit rating service. The credit rating service may return a rating if the customer has a past history or it may return a NULL value if the customer is not known. If the loan agency does not want to ever deal with customers without a credit rating, it may have a rule such as:

```
IF credit_score is 'NOT
AVAILABLE' THEN REJECT
loan application ELSE
EVALUATE 'Credit Risk'
```

This rule is very simple and not likely to be changed after process deployment. Hence, the fastest way to implement this would be by using the code-based approach with XPath or other expressions. Now let's consider a slightly more complex rule.

2. The loan agency may then use the customer's social security number (SSN), prior credit history, annual income, and outstanding loans to determine the customer credit rating, the risk, and maximum amount to lend a specific customer (as shown below).

```
IF customer credit score is less than
   500 AND outstanding loans are greater
   than $50,000 AND annual income is
   less than $70,000 AND loan requested
   is greater than $40,000 THEN set the
   Credit Risk to "High"

IF customer credit score is greater than
   500 AND annual income is greater than
   $70,000 THEN set the Credit Risk to "Low"
```

In the above rule, there are various inputs that are provided by the business analysts; for example, the minimum annual income requirements and credit score. The analysts would like the ability to change these without necessarily redeploying the process. In this case, the model-driven or service-oriented approach would offer the most flexibility, since the rules would be separate from the process logic and can be modified independently. The code-based approach would not work since it would not be maintainable in the long run, and as the rules get more complex, defining them through just simple expressions would be very difficult. Now let's examine a slightly more complex scenario.

3. Based on the current business environment and other company policies, the loan agency may interpret the results differently and further apply rules to determine if the customer should be granted the loan, what interest rate should be given, and the appropriate approval policies. Note that in case 2 above, all the information needed to apply the rules was available from the business process itself. However, in some cases, rules may require additional facts that

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

are asserted by other applications. Say you want to know if the "outstanding loans this month" are greater than US$3M or if the customer already has some other products – say, home equity loan or insurance products – from this company. These facts are provided to the rules engine by other applications, but are used by the rules engine in the decision to approve or reject a loan. For example:

```
// Rules for "High" risk customers
IF customer credit risk is High AND loan
   requested is greater than $40,000 AND
   [outstanding loans this month to high
   risk customers] is greater than
   $50,00 THEN "reject" the Loan
   application

IF customer credit risk is High AND loan
   requested is less than $50,000 THEN
   "Approve" the Loan application with
   an APR of 5%

IF customer credit risk is High AND loan
   requested is between $50,000 and
   $75,000 THEN "Route" the Loan
   Application to Manager for review

IF customer credit risk is High AND loan
   requested is greater than $75,000 THEN
   "Reject" the Loan application

// Rules for "Low" risk customers
IF customer credit risk is "Low" OR the
   customer credit risk is "Medium" THEN
   "Approve" the loan with an APR of 4.5%

IF the customer credit risk is not "High"
   AND the customer is a returning
   customer THEN "Approve" the loan
   with an APR of 4%
```

```
IF average APR of loans to HIgh Risk
   Customers in last month is greater
   than 7% THEN notify 'Loan Supervisor'

IF more than 50% of loans in Last 3
   monthsare to 'High Risk' customers
   THEN notify 'Finance Manager'
```

In this case, the decision service is used to evaluate rules based on both static data; that is, the loan application, as well as data such as "outstanding loans to high risk customers" from the BAM system. Similarly, rules are also used for dispatching the loan application for manual processing for high-risk customers. In this scenario, the code-based approach will obviously not work because it is not flexible enough and would be very difficult to maintain in the long run. The model-driven approach also is not very effective, since the rule engine requires facts/data from external systems such as a BAM solution or customer database. You could make the loan business process retrieve all this data and pass it to the rules engine; however, this would make the process very complex and would require changes every time the business analyst decided to use additional criteria for loan approval. The service-oriented approach works best in this case. You could have a central rules repository with a decision service that is used by all clients of the rules engine. The BPM solution, as well as other applications like BAM, would assert different facts to this service that executes the appropriate rule sets to make decisions.

This process and the various decision points are also illustrated in the architecture diagram in figure 1. As you can see, the first rule is implemented using inline process logic, and the rules service is used for cases 2 and 3 discussed above.

We suggest the following rules of thumb when deciding how to leverage rules logic in business processes:

- The code-based approach should be used for simple expressions that are not likely to need modification after the process is deployed. Examples of where the code-based approach works well are validation checks, such as "If the customer does not have a credit rating, reject the loan," or certain constraint checks, such as "All loan applications from a new customer must be accompanied by an employment verification letter." If business analysts are involved in capturing the rules, chances are that the rules are likely to change, and so you should go for the model-driven or service-oriented approach.
- Use the model-driven approach when the rules are likely to need changing or when you need to separate rules from the process logic. This allows you to separate duties – processes being developed by the process implementers – and the business rules being captured and managed by business analysts. Use this approach for workflow rules for load balancing of tasks ("A specific loan agent can have a maximum of 10 open tasks assigned at any point") or a policy-based task assignment ("Orders from Premier customers are handled by reps in Dept B, but other orders are routed to Dept A"). As long as the rules do not require external facts, the model-driven approach works well for a wide range of uses. If they do however, you need to consider the service-oriented approach!
- The service-oriented approach works best when you need to use facts from multiple business processes or applications in your decisions, or when you don't want to use the embedded rules engine in your BPM solution, e.g., because you already have a rules engine that is in use for other applications. It may be a little more difficult to set up, but offers the most flexibility and having a common rules repository enables rules reuse across processes and applications. It requires you to plug your rules engine into your BPM solution. BPEL-based BPM solutions tend to make this easier compared to the monolithic BPM suites. Examples of when the service-oriented approach works well include when you have a common rules repository that is used to handle vacation, delegation, and escalation rules. This would be used in HR-related applications as well as other financial applications for any approval workflow. Another example is when you have rules to decide the APR for loans not just based on the customer details available from the business process, but also real-time aggregate information/facts fed to the rules engine from another component, e.g., BAM solution.

Business Rules as a Decision Service Using the Service-Oriented Approach

Now let's talk about how you can embed rules in your composite application using a decision service. A decision service is a mechanism for publishing rules and rule sets as a reusable service that can be invoked from multiple business processes and applications. To set this up you have to: (1) define the facts and the corresponding rule sets that will be used by various applications, (2) publish these as a decision service that can then be invoked from the business process.

- ***Define facts:*** As discussed earlier, each rule that is exposed as a service will use different types of facts. These facts have to be created via XSD definitions or by importing the corresponding Java classes. If you are using an open standards–based BPM solution based on BPEL, it is preferable to use the same XSD definition for

defining facts for the rules engine and variables in the BPEL process to avoid the need for transformations when exchanging data with the decision service. If a rules engine supports only Java-based objects, you could use a Java Architecture for XML Binding (JAXB) translator to generate Java objects from XSDs.

- ***Design rules:*** Rules are generally authored using the rules designer of the BRE or the rules tooling provided by the BPM solution. For some complex rules, you may have to directly use the underlying rules language. At this point, you may also decide which rules are modifiable by business analysts. For this you may have to define aliases for certain fields in your data model and also define appropriate constraints on the rules; for example, you could say that [customer.loanapplication.loan_amount] has an alias "Loan Amount." By using these aliases, you don't have to worry about the actual hierarchical structure of the data model. The business analyst defining the rules does not really need to know where this attribute is retrieved from in the hierarchical data model. The rules can be grouped into rule sets and deployed to the repository.
- ***Create the decision service:*** The decision service is a Web service that wraps the rule session to the underlying rules engine. This service lets business processes assert and retract facts as part of the process. In some cases all facts may be asserted from the business process as one unit, while in other cases, the business process may incrementally assert facts and eventually consult the rules engine for inferences. Hence, the service has to support both stateless as well as stateful interactions. Various interaction patterns are possible with this service – assert, assert/execute, etc.
- ***Load facts:*** Facts may be passed in directly from the business process or may be periodically loaded from a back-end repository, such as a database, file system or an external application. Adapters or other fact-retrieving mechanisms can be used to load facts periodically from back-end systems such as databases or files. The adapter services can be used internally as part of the decision service to load facts. For example, for the credit rating decision service, facts such as the customer's SSN and loan amount may be passed in from the business process, but other facts, such as the customer's credit history and outstanding loans, may be retrieved using adapters from a back-end repository.

The aforementioned architecture is illustrated in Figure 1. The rules engine works off a common rules repository, and all rules are exposed via a decision service. The process developer defines the rules metadata, creates the service, and designs the BPEL process to interact with the rules. The business analyst generally uses the rules metadata to define the rules and customize rules on an ongoing basis after the process is deployed. The BPEL processes, BAM, and other applications assert facts and execute rules by interacting with the decision service.

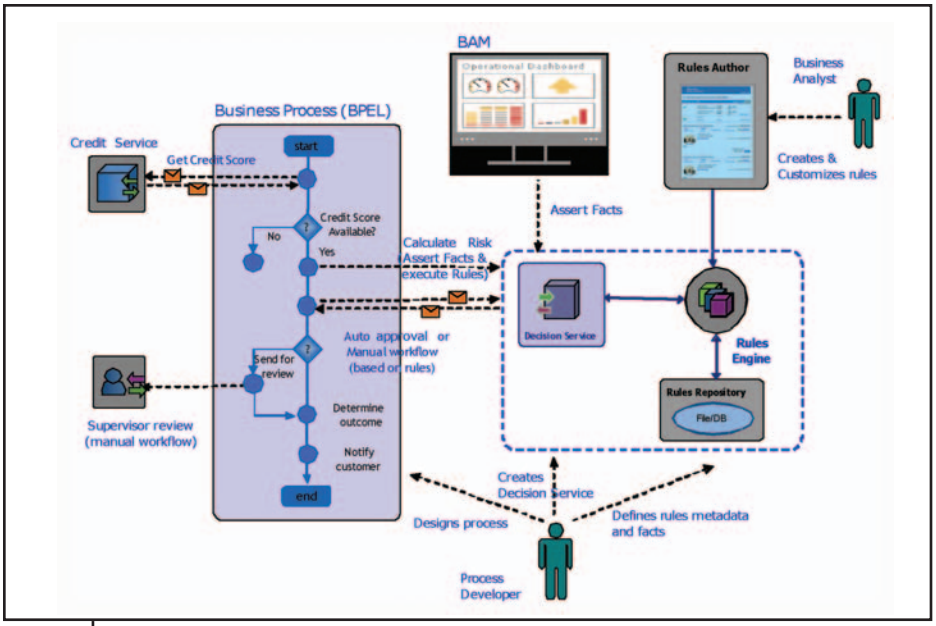


FIGURE 1 | **BPEL Process and rules service architecture**

Summary

We have provided details on the approaches that you can take for implementing decision logic in business processes, and given some guidance on when to apply each approach. Monolithic BPM suites limit your flexibility in integrating process and rules logic because they don't allow you to easily assert facts from external sources or use your existing rules engines. We believe that an open standard, BPEL-based BPM solution is ideal. Such an open approach enables you to develop rules in a code-based, model-driven, or service-oriented manner. It provides an embedded business rules capability, yet allows you to replace the rules engine with your choice of third-party rules engine, or make the embedded and your third-party rules engine coexist. By delivering on this, it yields ease of development through common tooling for process and rules, and improved performance, yet delivers full flexibility. Who says choice is a bad thing?

References

Further details on how to implement a service-based business rules service in the context of BPEL can be found at www.oracle.com/technology/index.html under the link "SOA Best Practices: The BPEL Cookbook."

About the Author

Mohamad Afshar, PhD, is director of product management for Oracle Application Server 10g. He has core product management responsibilities for the Application Server that is part of Oracle Fusion Middleware. His main focus includes middleware vision, strategy, and architecture, with an emphasis on Web services, SOA, and EDA. Mohamad is a frequent speaker at industry events and is a contributing author to business-oriented, technical, and academic journals. He holds a PhD in Parallel Database Systems from Cambridge University, UK.

mohamad.afshar@oracle.com

Bhagat Nainani is a product development manager in the Oracle Application Server division. He currently leads the development of BPM services for the Oracle BPEL Process Manager. He has more than 10 years of experience with distributed systems, enterprise software, and integration technologies.

bhagat.nainani@oracle.com

FusionWare Integration Server

Build composite applications quickly and easily

In a service-oriented architecture, the ultimate goal is to quickly and easily build new applications as composites of existing services. Building out the independent services in a manner that supports reuse is itself very challenging. However, several tools are beginning to appear to easily bring applications together once the critical mass of applications is achieved. One such tool is the FusionWare Integration Server.

WRITTEN BY
BRIAN BARBASH

The FusionWare Integration Server is a suite of products that provides systems integration, workflow, and data transformation capabilities to build composite applications. The suite consists of the following:

- ***FusionWare Server:*** Java-based server application that provides the run-time environment for FusionWare applications
- ***FusionWare Designer:*** IDE for building and deploying FusionWare applications
- ***FusionWare Administrator:*** Management utility for use with the FusionWare Server
- ***FusionWare Client Simulator:*** Graphical utility for submitting and reviewing application calls to the FusionWare Server

Applications are stitched together via XML, database, and Java/COM componentry, and executed via XML over HTTP or through a file-based XML interface.

For the purposes of this review, I have created a Project Management application in FusionWare Designer to manage Time, Expense, Budgeting, and Invoicing for a fictional consulting company.

Building Applications

FusionWare applications are built in the Designer as a set of Business Processes that act upon an XML document submitted to the system either via an HTTP GET/POST operation, or from a designated file system directory that is constantly polled. Each business process is in turn a series of workflow steps that manipulate the submitted XML document, call external systems, reusable subroutines, Web Services, and databases, and return the results to the calling entity.

The Designer itself follows a Windows Explorer–like paradigm. As seen in Figure 1, higher-level objects are located on the left side of the screen with details and configuration information located on the right. While functional, this interface can at times be challenging because it does not show the business process model using the traditional flowchart or swim-lane diagrams. In complex business processes with many workflow steps, logic branches, and data manipulations, it can become difficult to keep track of the flow of data. While it's not something to disqualify the product from consideration, it may take time for a developer to become acclimated.

Figure 1 shows the model in the Designer for the sample application I've created. The left side of the screen represents the individual business processes defined in the application, the middle of the screen shows the individual workflow steps for the selected business process, while the right side of the screen shows the details of the workflow step currently selected.

In most situations, as with this example, models will contain multiple business processes. Therefore, developers must define recognition tests that evaluate the incoming XML document to determine the appropriate business process to execute. Recognition tests are simply XPath statements that define an XML NodeSet. If the XPath returns a NodeSet, the recognition test evaluates to True and process flow is directed to its associated business process. For example, the business process bpCreateClient in the sample application should be executed when the incoming XML document contains the appropriate Client nodes. Figure 2 shows the associated Recognition Test.

Recognition tests may also be associated with workflow steps independent of the parent business process (also shown in Figure 2). In many cases this capability is used to perform common operations such as logging, message archiving, transformation to canonical message forms, or message validation against Schemas and DTDs. In the case of the example in Figure 2, the incom-

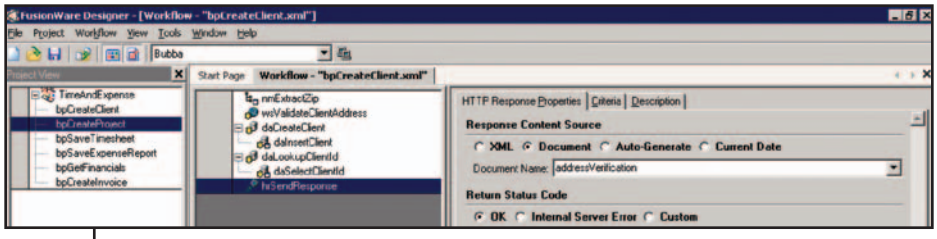


FIGURE 1 | **FusionWare Designer**

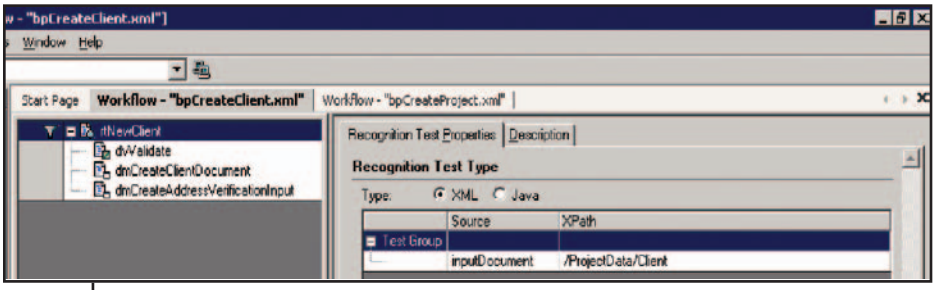


FIGURE 2 | **Recognition tests**

ing XML document is validated against a schema and supplemental documents are instantiated supporting other operations in the workflow.

Throughout the Designer tool, the manipulation and extraction of XML data via XPath statements is an important function of many of the available actions. To facilitate this, FusionWare Designer provides a very simple but effective XPath generation tool. Developers simply select the relevant XML nodes from sample instance documents. The generated XPath statement is then applied to the current action. These statements may be further tweaked manually to support more complicated operations.

One very powerful feature of the FusionWare solution is the ability to create and execute parallel tasks within a single business process. For example, if a consultant is submitting an XML document that contains both a Timesheet and an Expense Report, FusionWare may be configured to execute the individual business processes relevant to each type of data structure in parallel.

Systems Integration

FusionWare provides a set of systems integration capabilities to link together existing systems to form composite applications, such as:

- ***Database Access:*** FusionWare may access any database available via a JDBC connection
- ***Web Services:*** External web services calls may be made from within a FusionWare application
- ***Java/COM:*** Custom Java and COM objects may be called to provide connectivity and control to applications not accessible through standard FusionWare methods

For the sample Project Management application, two main types of integration will be required: database access to the system's persistent store and Web services access to external services for data validation.

The Project Management application requires address information when creating clients to support invoicing. To ensure valid addresses are submitted, a call to an external Web service will be made. Setting up a Web service call in FusionWare designer is very simple. Figure 3 shows the configuration screen for Web services. The developer must identify the URI to the WSDL file for the service and the desired operation. Once defined, all input parameters must be mapped to the service call. Parameter values may either be defined manually in XML format, or extracted via XPath from a run-time variable. Results of the service call are then delivered to a result document defined in the business process. Developers have the option of either saving the result as a whole in an independent variable, or targeting a specific node in a predefined result document using an XPath statement.

Defining database operations in FusionWare designer is a simple multi-phased process. Initially, the developer defines the Database Access properties. These properties establish the document containing the input parameters and the location to where results of the operation will be written. The second phase of the process establishes one or more Database Actions. These actions represent individual SQL and Stored Procedure calls. Within an Action, the developer defines the connection to be used, the SQL or Stored Procedure call to invoke, the XPath identifying where to retrieve relevant parameter values from the input document, and if applicable, the element names to be used for each row of the resultset.

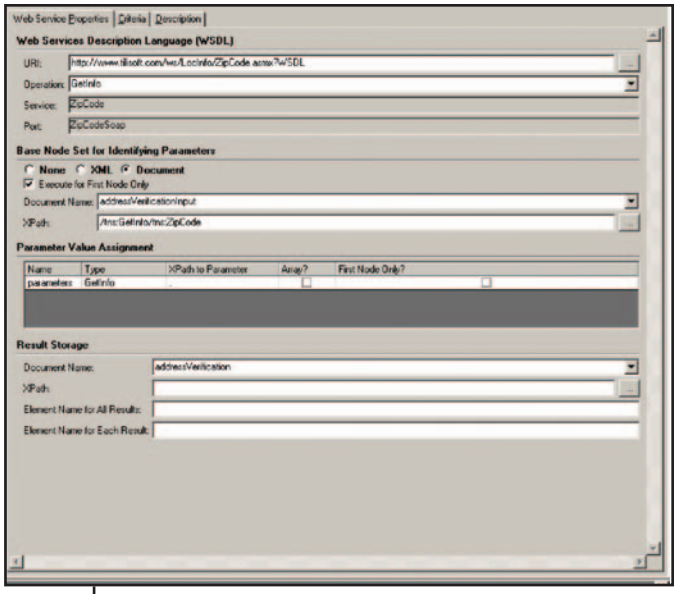


FIGURE 3 | **Web services console**

Summary

Service-oriented architectures, in their ideal implementation, provide a catalog of services that may be brought together to support new business processes or functions. Governing the relationships, integrating data, managing transactions, and optimizing performance among the constituents of composite applications can prove to be challenging. FusionWare Integration Server provides a simple but effective tool for achieving this end goal. It is a solid offering to be considered for an SOA environment.

About the Author

Brian R. Barbash is the product review editor for *Web Services Journal*. He is a senior consultant and technical architect for the Envision Consulting Group, a management consulting company focusing on contracting, pricing, and account management in the pharmaceutical industry.

bbarbash@sys-con.com

Does Application-Oriented Networking Impact SOA?

Use of AON will revolutionize SOA as well as the networking arena

The requirement of being agile in today's market means having business processes at one's command that can easily be modified for different requirements. This requirement, along with usage of open standards that addresses the problem of interoperability, compels today's business units to move towards service-oriented architecture (SOA). This article talks about the impact AON would have on such business systems.

WRITTEN BY

BIJOY MAJUMDAR

VIKAS KUMAR

VIVEK RAUT

Over the years, enterprise business seamlessly integrated assets by virtue of enterprise application integration (EAI). EAI systems were largely proprietary and came with lots of adapters that needed hands-on attention from experts. Application communication has become a set of configurations and settings for the integration solutions.

As time went on, business application development time has been reduced. Organizations have to mix and match lots of systems, as well as the richness, complexity, and interdependency of information systems too. Today the computing world is increasingly complex and interdependent. It has lead to a huge demand for an implicit integration platform or a message routing backbone.

The paradigm for SOA is proliferating at a reasonably good pace, but before it can reach its full potential, there are certain issues that need to be addressed such as reliability, security, and performance issues. Application-oriented networking (AON) has provided a mechanism to control network traffic and make the network more intelligent.

An underlying, implicitly executed integration component like AON is the need of the hour, but how will AON help, and what impact would it have for SOA-based applications?

Traditional networking equipment moves packets across destinations solely by looking at the packet headers of incoming traffic. Application artifacts, integration applications, or SOA are wholly software based and have redundant usage of XML-based operations and routing. Integration brokers are quoted as the application software analogous to the hardware-based networking routers and switches. The marriage of the application software to the networking components means an intelligent messaging network system at the behest of a single network component. This concept gives birth to application-oriented networks.

AON has given organizations an opportunity to optimize the use of networks to bring flexibility and visibility into their business. The added intelligence to the network has overcome many overheads by making the applications more business-centric. AON has provided networks with an ability to look into the messages so that it can route and transform messages according to business rules.

AON helps because:

- It is a hardware alternative to traditional integration suites
- Enhances XML processing capabilities
- Integrates visibility to the network
- Analyzes the content of messages over the network
- Provides secure and fast movement of XML and non-XML messages
- Provides consistent policy enforcement for security, transaction routing, etc.
- Fosters XML-based technologies and SOA

The installation of the network boxes help in realizing the aforementioned benefits. This will differ across the business infrastructure according to the complexity level of application integration. AON can be scattered throughout the organization and configured with different policies and rules at different locations.

This is a big leap in terms of restructuring and reengineering the whole business architecture. Adopting AON will mean not only removing the already existing middleware, but also amending existing applications.

There is a whole paradigm shift here. IT needs to define the specific functionalities performed by each application server, the business applications, database, and network appliances in the distributed business system. The danger occurs when one

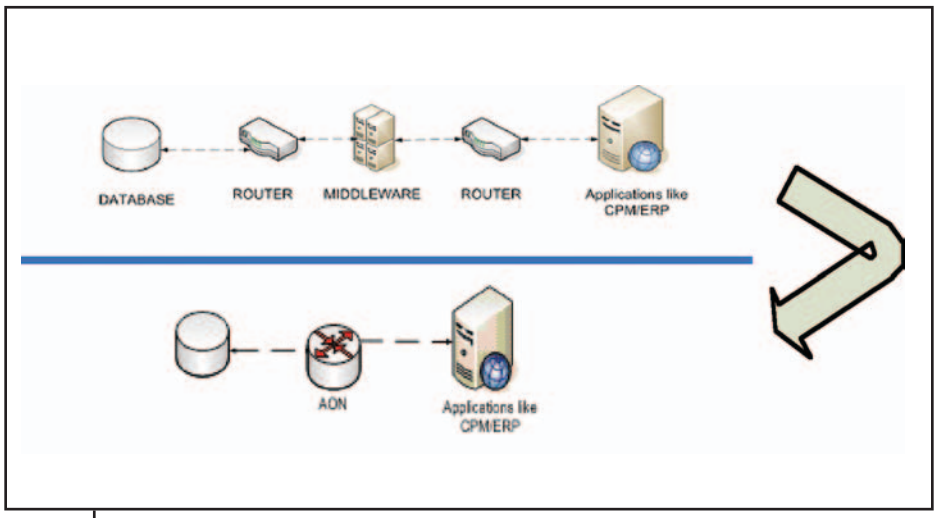


FIGURE 1 | **Technology variation caused by AON** *Source: Infosys Research*

layer overrides the functionality of the other layer. It is still remains to be seen whether some of the functionality that is put in the network will really suffice well, or if there will be some functions that will serve best in the application tier itself.

AON fosters implementation of SOA. SOA talks about implementing various functionalities in the form of services of appropriate granularity. With AON, the applications need not worry about implementation of services like implementing and monitoring security policies. Moreover, AON does the XML processing after analyzing the content of XML or non-XML messages. For traditional systems, the use of AON will require few changes in the application because the functionalities for security implementation and application redirection based on message content etc. are taken out of application and implemented in the network boxes. In contrast, if the architecture is based on an SOA paradigm then it is easy to implement changes in the application because there is very little change associated with it. By offering flexibility and effective use of resources and time, AON has already started a trend towards SOA.

AON enables implementation of functionality layers in the infrastructure itself. This approach will foster the adoption of SOA because the underlying infrastructure would itself pave the path for implementing it.

AON well fits with the SOA paradigm-based applications. The network appliances can replace the middleware and bear the burden of transformation and application-based intelligent routing. By removing a layer of complexity and offloading some of the responsibilities of applications, AON in an SOA-based system has made enterprises more agile.

AON is very well suited for the SOA-based architecture because of the nature of data transferred. SOA is characteristic for the large-scale XML-based transfer of data and has often being accused of performance slowdown as a trade-off for open standards usage. Now AON comes to rescue of SOA because it has provisions for processing XML data, such as the performance of parsing, transformation, etc. for XML traffic, thus providing a huge gain in the performance. Apart from this AON, is capable of peeking

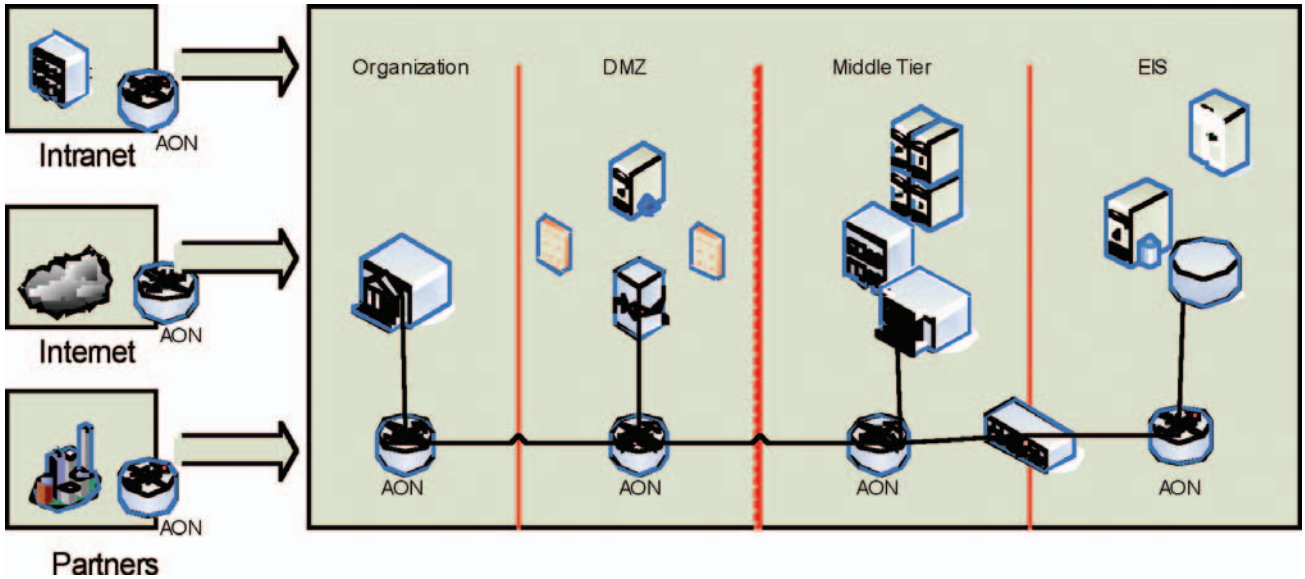


FIGURE 2 | **Usage of AON** *Source: Infosys Research*

through the packets that are transferred across the network and thus provides an intelligent security implementation mechanism through policies along with intelligent message routing.

AON integrates message-level communication, security, and visibility in the network. The concept behind this is making the network intelligent enough so that it acts like a platform for enabling business processes along with technology. This fosters XML-based technologies and SOA because the network takes the responsibility of making applications talk to each other. Indeed, AON has already started a trend towards SOA.

References

- *Cisco takes aim at surging XML, RFID traffic (all articles accessed October 2005):* www.findarticles.com/p/articles/mi_qa3649/is_200506/ai_n14682512
- *Cisco Announces Its Application-Oriented Networking:* www.enterprisenetworksandservers.com/monthly/art.php/1555
- *DataPower Extends Lead in Application-Oriented Networking:* www.datapower.com/newsroom/pr_013105_xi50_3dot1.html
- *IBM Teams up with DataPower to Offer Application-Oriented Networking Devices to SOA Engagements:* www.webservices.org/ws/content/view/full/72834
- *Kenamea ENS Integrates with Cisco Application-Oriented Networking:* www.eeproductcenter.com/showPressRelease.jhtml?articleID=X368691
- *Procera Equips ColoSpace's New Internet Data Center:* www.proceranetworks.com/news/071405.asp
- *Cast Iron Offers New Application Integration Appliances:* www.infoworld.com/article/05/06/30/HNcastiron_1.html
- *Application Infrastructure Primer for Network Professionals:* www.cisco.com/en/US/products/ps6438/products_whitepaper0900aecd802c2000.shtml

About the Authors

Bijoy Majumdar is a member of the Web Services COE (Center of Excellence) for Infosys Technologies, a global IT consulting firm, and has substantial experience in publishing papers, presenting papers at conferences, and defining standards for SOA and Web services. Prior to Infosys, Bijoy Majumdar worked as an IT Analyst, and had been a member of the GE Center of Excellence (e-center) under the E-Business Practice of Tata Consultancy Services.

bijoy_majumdar@infosys.com

Vikas Kumar is a member of the Web Services COE (Center of Excellence) for Infosys Technologies, a global IT consulting firm, and has substantial experience in publishing papers, presenting papers at conferences, and defining standards for SOA and Web services.

vikas_kumar02@infosys.com

Vivek Raut is a member of the Web Services COE (Center of Excellence) for Infosys Technologies, a global IT consulting firm, and has substantial experience in publishing papers, presenting papers at conferences, and defining standards for SOA and Web services.

vivek_raut@infosys.com

XML JOURNAL SUPPLEMENT

This Month

Untangling the Semantic Web
Erin Cavanaugh

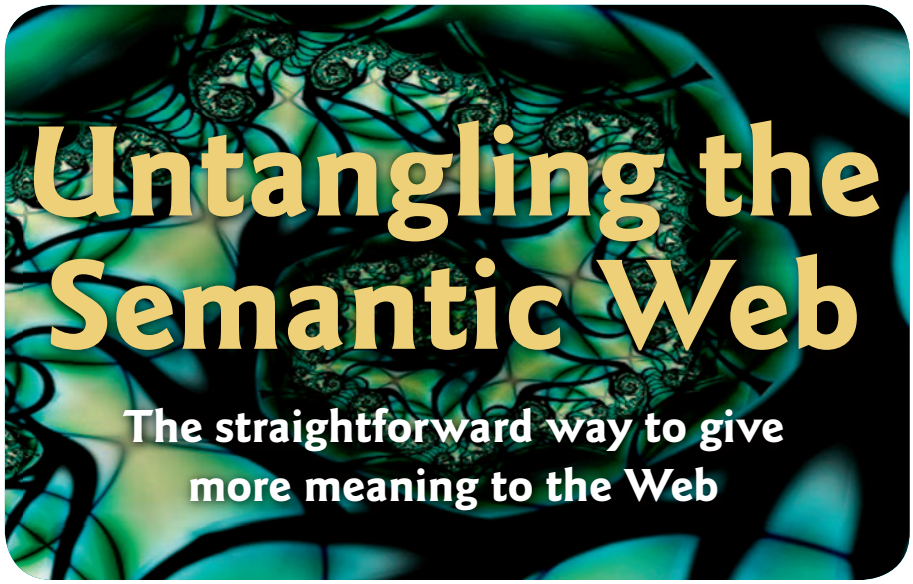
The Semantic Web is a hot topic in information circles today, and its adoption will largely depend on stakeholders understanding its potential benefits and tools vendors providing an easy entry for developers to learn and work with its related technologies.

The Case for XQuery
Jerry King

XML use is widespread across modern information systems in all industry, government, and academic sectors. The core technologies for processing XML (XML, XSLT, XPath, XML Schema, and others) are maturing steadily – thanks to support from standards bodies like the W3C and OASIS, and from major industry players such as IBM, Microsoft, and Oracle. XML is also the basis for a growing body of industry standards for data exchange, and it is well on its way to becoming a mainstream technology for data integration. XML is transforming not just data – it is transforming information processing in general.

eXist
Selim Mimaroglu

This article will introduce you to the open source, free (GNU LGPL license), native XML database eXist (www.exist-db-org). Data is important, no question about it. Data that can't be queried is not very useful. Users expect to have good query response time. From my personal experience and testing, I am confident in saying that eXist is a fairly good database. It has very good query response time, it is very user friendly, it's easy to set up and operate, and it's written in Java, therefore it is platform independent.



XML-Based Interop, Close up

In addition to the strategy side of Web services, there is also the protocol-oriented side of things, the XML side. Embracing not only XML itself but also the full range of mainstream XML-based technologies like XPath, XSLT, XML Schema, and SOAP. *XML Journal* has been delivering insightful articles to the world of developers and development managers since the year 2000.

It is our privilege to bring XML-Journal directly to readers of Web Services Journal, and vice versa. Anyone already familiar with the Web services world of SOAP, UDDI, and WSDL will find here articles and features each month that will interest them – about the cutting-edge technologies and latest products that are changing not only our industry, but the way the world exchanges information. To make it easy for you to find your way around, we have four distinct sections:

Content Management:
Organization, dissemination, and presentation of information

Data Management:
Storage, transformation, representation, and general use of structured and unstructured data

Enterprise Solutions:
Systems and applications that manage mission-critical functions in the enterprise

XML Labs:
Product reviews, book reviews, tutorials, and standards analysis

Untangling the Semantic Web

The straightforward way to give more meaning to the Web

WRITTEN BY
Erin Cavanaugh

The Semantic Web is a hot topic in information circles today, and its adoption will largely depend on stakeholders understanding its potential benefits and tools vendors providing an easy entry for developers to learn and work with its related technologies.

The Semantic Web Vision

Imagine this scenario. You're a software consultant, and today you're taking a working lunch with one of your biggest clients. Her company has an emergency project at one of its remote offices, and they need your consulting services there for the next two weeks. You need to get there as soon as possible to begin work, so you take out your hand-held computer, activate its Semantic Web agent, and instruct it to book a nonstop flight that leaves before 10 a.m. the next day. You want an aisle seat if it's available. Once the agent finds an acceptable flight with an available aisle seat, it books it using your credit card and assigns the charges to your client's account in your accounting application. It also warns you that you'll be missing a dentist appointment back home and adds a note to your calendar reminding you to reschedule. Next, you specify that you want a car service to the client's site, so the agent scans the availability of limos in the area with "very good" or higher service ratings and books an appointment to have you picked up 30 minutes after your flight lands. The agent also books you at your favorite hotel chain, automatically securing the lowest rate using your rewards card number. Finally, your agent updates your calendar with your trip information and prints out your confirmation documents back at your office.

With just a few clicks your Semantic Web software agent found and booked your flight, hotel, and car service, then updated your accounting system and calendars automatically. It even compared your itinerary to your calendar and detected the scheduling conflict with your dentist appointment. To do all this, the agent had to find, interpret, combine, and act on information from multiple sources and multiple, disparate applications and data repositories. During those few minutes it takes your software agent to book your trip, you wonder what you ever did before the Semantic Web. This example, of course, is a long-term vision for applying the Semantic Web. It's one that may or may not come to fruition – only the future will tell. However, the vision itself is important for understanding the potential of Semantic Web technologies.

The Semantic Web is currently the focus of W3C Working Groups (www.w3.org/2001/sw/) and is considered the next step in Web evolution. In the Semantic Web, data itself becomes part of the Web and is able to be processed independently of application, platform, or domain. This is in contrast to the World Wide Web as we know it today, which contains virtually boundless information in the form of documents. We can use computers to search for keywords in these documents, but search results still have to be read and interpreted by humans before any useful information can be extrapolated. Computers can present you with information but can't understand what the information is well enough to display the data that is most relevant in a given circumstance. The Semantic Web, on the other hand, is about having data as well as documents on the Web so that machines can process, transform, assemble, and even act on the data in useful ways. To accomplish this, the Semantic Web relies on structured sets of information and inference rules that allow applications to "understand" the relationships between different data resources.

The true impact of the Semantic Web will not be known for quite some time, but some proponents have asserted that it will lead to the evolution of human knowledge itself by allowing people and machines – for the first time – to quickly filter and synergize the massive amounts of data that exist in the world in a relevant, productive way.

As with any potentially revolutionary technology, the scope of the Semantic Web's evolution will depend on several factors, including industry buy-in, the technology learning-curve, and the availability of productive Semantic Web

development tools. The first factor, industry buy-in, is largely dependent on the other factors of developer proficiency and tools support.

Spinning Meaning and Relationships

The Semantic Web is a "web of data" that not only harnesses the seemingly endless amount of data on the World Wide Web, but also connects that information with data in relational databases and other noninteroperable data repositories. Considering that relational databases house the majority of enterprise data today, the ability of Semantic Web technologies to access and process them alongside other data from Web sites, other databases, XML documents, and other systems increases the amount of available data exponentially. In addition, relational databases can adapt easily to the Semantic Web model since they already include a great deal of semantic information. Database tables and columns are created based on the relationships between the data they house, and this organization reveals some of the meaning – the semantics – of the data.

Implementing the Semantic Web requires adding semantic metadata to the information resources that are available on the Web or on internal networks. This will allow machines to effectively process the data based on the semantic information that describes it. When there is enough semantic information associated with data, computers can make inferences about the data, i.e., understand what a particular data resource is and how it relates to other data.

> "The Semantic Web is about having data as well as documents on the Web so that machines can process, transform, assemble, and even act on the data in useful ways"

XML has paved the road by adding some metadata in the form of human-readable tags that describe data. In addition, XML documents can include information about the author of a Web page, relevant keywords for search engine optimization, and the software tools used to create the XML file, for example.

Before XML, data was stored in flat file and database formats, where most data was proprietary to an application. XML came along and made data interoperable within a single domain, i.e., within the domain defined by a schema or a set of related schemas that define the structure of related documents. By itself, XML provides syntactic interoperability only when both parties know and understand the element names used. If I label an element <price>12.00</price> and someone else labels it <cost>12.00</cost>, there's no way for a machine to know if those are the same thing without the aid of a separate application to map between the elements. Semantic Web technologies help address this problem by making tags understandable not just to humans – but to machines as well.

The first step required for machines to understand data is to get that data into a uniform format, where, for instance, a field labeled "street" always has the same format and contains the same type of information, and so on. This type of functionality can be found today on Web sites that use forms that allow users to enter information and run a query, such as airline sites that allow visitors to search for and book flights based on a variety of criteria. However, considering the amount and variety of data available from different sources today, this method of data typing does not scale beyond very specific applications.

The next step towards the Semantic Web requires data from multiple domains to be classified based on its properties and its relationship with other data. This is where Semantic Web technologies such as RDF and OWL come in.

RDF and OWL

RDF (Resource Description Framework) is the XML-based W3C standard that forms the basis for the Semantic Web. RDF statements describe a resource (identified by a URI), the resource's properties, and the values of those properties. RDF statements are often referred to as "triples" that consist of a subject, predicate, and object, which correspond to a resource (subject), a property (predicate), and a property value (object). A triple written in plain text is depicted in Figure 1.

By creating triples with subjects, predicates, and objects, RDF allows machines to make logical assertions based on the associations between subjects and objects. However, while RDF provides a model and a syntax (the rules that specify the elements of a sentence) for describing resources, it does not specify the semantics (the meaning) of the resources. To truly define semantics, we need RDFS or OWL.

[subject]	[predicate]	[object]
The Secret Agent	is named	Niki Devgood
[resource]	[property]	[value]

Figure 1 • An RDF triple in plain text

RDFS (RDF Schema) allows developers to create vocabularies that describe groups of related RDF resources and the relationships between those resources. An RDFS vocabulary defines the allowable properties that can be assigned to RDF resources within a given domain, and it allows creation of classes of resources that share common properties.

In an RDFS vocabulary, resources are defined as instances of classes. A class is a resource too, and any class can be a subclass of another. This hierarchical semantic information is what allows machines to determine the meanings of resources based on their properties and classes.

Building upon RDFS is OWL, which is a much richer, more expressive standard for defining Semantic Web ontologies that formally define the hierarchies and relationships between different resources. Semantic Web ontologies consist of a taxonomy (system of classification) and a set of inference rules from which machines can make logical conclusions. OWL is used to assign properties to classes of resources, and their subclasses inherit the same properties. OWL also utilizes the XML Schema datatypes and supports class axioms such as subClassOf, disjointWith, etc., and class descriptions such as unionOf, intersectionOf, etc. Many other advanced concepts are

included in OWL, making it the richest standard ontology description language available today. There are three flavors of OWL, each with increasing flexibility: OWL Lite, OWL DL, and OWL Full. Developers choose the OWL dialect to work with based on the level of expressive restriction they need in their ontology.

Because RDF, RDFS, and OWL documents express hierarchies and relationships between resources, they are often created and conceptualized in a graphical manner to make the underlying relationships immediately obvious. Figure 2 shows an example of a simple RDF graph.

Even in this simple example, it's easy to see how a Semantic Web agent could make logical connections based on the defined relationships. For example, since the secret agent is Niki Devgood, and the secret agent drives a red convertible, it follows that Niki Devgood drives a red convertible.

Complex ontologies are represented with multiple, interdependent graphs that visually reveal the relationships between resources. Once Semantic Web documents are defined and mapped out graphically, they must be coded in RDF/XML or N-Triples format to be accessed programmatically. Unfortunately, the manual coding process can be extremely tedious and error-prone, considering that even simple ontologies can represent hundreds of lines of code and given that neither RDF/XML nor N-Triples provide visual cues as to the hierarchy of the information contained therein. Developers need a way to translate their graphical ontology representations into RDF/XML or N-Triples easily, thus removing a significant barrier to Semantic Web adoption.

Semantic Web Evolution

Even when developers are armed with tools that make Semantic Web development practical, it's important to note that implementation of RDF, OWL, and the Semantic Web as a whole will be a gradual process. Questions about what the Semantic Web is and how it can benefit businesses and individuals echo the initial confusion about why we needed HTTP, HTML, and the basic Web infrastructure before "WWW" was a staple of our daily vocabulary. However, considering how those technologies have proliferated, it's likely that the Semantic Web vision is one that will be realized, even if it's on a small scale initially.

Though there are certainly far-fetched visions of Semantic Web technologies allowing your PC to talk to your refrigerator to autogenerate recipes and shopping lists, the number of scenarios that could potentially benefit from Semantic Web technologies as they continue to evolve is truly impressive. Think of the possibilities opened to everything from crime investigation, scientific research, and literary analysis, to shopping, finding long-lost friends, and vacation planning when computers can find, present, and act on data in meaningful, productive ways. Despite the theoretical possibilities, only time will tell if these advanced Semantic Web visions will come to fruition.

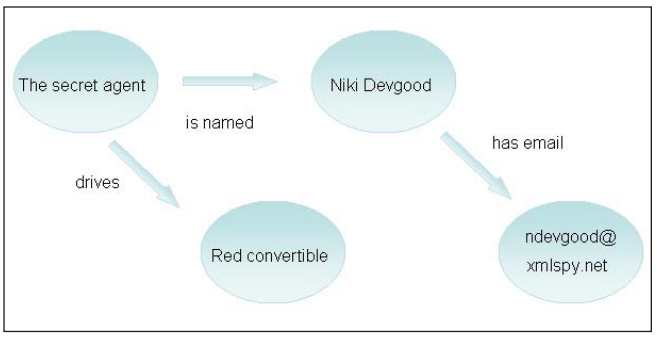


Figure 2 • An RDF graphic

In the meantime, the W3C has put forth a list of practical use

> "In the Semantic Web, data itself becomes part of the Web and is able to be processed independently of application, platform, or domain"

cases for Semantic Web technologies (www.w3.org/TR/webont-req/#section-use-cases), several of which are already in place today. For instance, semantic information can be added to resources on Web portals to improve information syndication and increase the productivity of searches within the portal. Because a Web portal generally includes information related to a narrow community of interest, it's well suited to ontological definition. The Semantic Web can also be used to describe non-textual resources, such as multimedia collections that contain audio, video, and other file types, making locating, combining, and utilizing these resources infinitely easier.

Another example is the Dublin Core Metadata Initiative, which applies Semantic Web technologies to create vocabularies that define the properties of informational resources, such as creator, format, creation date, description, and so on. The Dublin Core vocabulary is in use today in a wide variety of projects (a full list is available on dublincore.org).

These few examples make it clear that industry adoption is increasing, and that trend will only continue with the availability of productive RDF and OWL editors.

Visual Semantic Web Tools

Recognizing this need for practical Semantic Web development tools, Altova recently released SemanticWorks 2006. SemanticWorks is a visual RDF/OWL editor that allows developers to define RDF, RDFS, and OWL documents graphically. Its functionality comprises:

- Support for visual creation and editing of RDF, RDFS, OWL Lite, OWL DL, and OWL Full documents
- Intelligent entry helpers that offer context-sensitive editing choices
- Syntax checking for RDF, RDFS, and OWL documents
- Semantics checking for OWL Lite and OWL DL ontologies
- Autogeneration and editing of RDF/XML or N-Triples formats based on visual RDF/OWL design

SemanticWorks supports RDF, RDFS, OWL Lite, OWL DL, and OWL Full with full syntax checking. Context-sensitive entry-helpers present the available options based on the type of document being created (RDF, OWL Lite, etc.), so it's easy to experiment and work with the technologies to create valid documents quickly. Support for OWL semantics checking also helps ensure consistency throughout an ontology. An example of the visual ontology design view is shown in Figure 3.

The graphical display includes informative icons that indicate item types, containers and collections (bag, sequence, etc.), class descriptions (unionOf, intersectionOf, etc.), class axioms (subClassOf, disjointWith, etc.), property descriptions (subPropertyOf, inverseOf, etc.), and more. These connectors can be inserted using a context-sensitive right-click menu or by selecting them from the toolbar. Yellow boxes encapsulate resources that are defined elsewhere in the ontology, and mouse-over hints display a connector's meaning or a resource's URI.

Based on the visual design, SemanticWorks generates the corresponding code in RDF/XML or N-Triples, depending on the user's preference. This allows developers to focus on the relationships they're defining while leaving the low-level code writing to the application, which reduces the semantic technology learning curve significantly. This is also helpful for viewing the impact of changes during editing, whether you change the visual design and view the corresponding code, or vice versa.

Because neither RDF/XML nor N-Triples code reflects an order or hierarchy, it's very difficult to understand the relationships between resources when an ontology definition is viewed in its text form. SemanticWorks removes this problem by representing RDF and OWL components visually. It separates vocabularies and ontologies into their logical parts in the visual design view to help users immediately understand the relationships between resources and work with all the of components that make up the definition in a logical manner. These ontology components are available on five tabs: Classes, Properties, Instances, allDifferent, and Ontologies. The Classes tab lists all of the classes available in the ontology with a separate window that lists the instances and properties of the selected class. All properties of the ontology are listed on the Properties tab, and a separate window below the tab lists the domain of the currently selected property. All instances are listed on the next tab, and the allDifferent tab lists the resources that are defined as mutually distinct. Last, the Ontologies tab lists all resources that are ontologies, including ontologies that have been imported into the current file. Figure 4 shows the ontology tabs view. Selecting any listed item opens the detailed design view (shown in Figure 3).

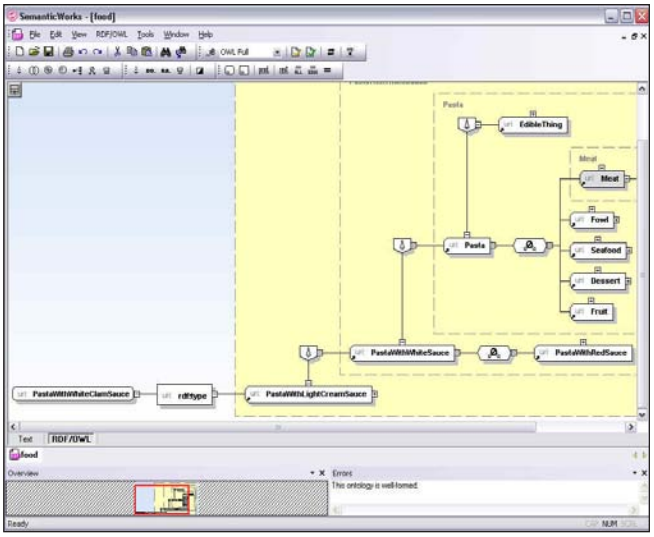


Figure 3 • Altova SemanticWorks – graphical RDF/OWL editor

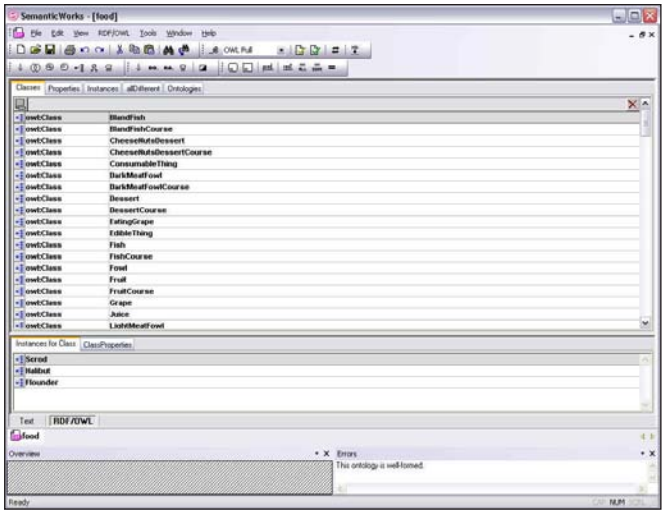


Figure 4 • Altova SemanticWorks OWL components viewr

In effect, SemanticWorks allows users to graphically compose ontology drawings – which they might otherwise create by sketching on a notebook or whiteboard – with valuable editing help and automatic code generation. By providing a visual design paradigm and removing the need to manually write RDF/XML or N-Triples code, Altova SemanticWorks gives developers a practical, accessible entry into the Semantic Web.

Conclusion

A new generation of applications geared to take advantage of the Semantic Web is waiting to be built. Now, the availability of commercial development tools such as Altova SemanticWorks will help drive developer productivity and industry adoption, leading us one step closer to realizing the Semantic Web vision.

References

- Daconta, M., et al. The Semantic Web. Wiley Publishing. Indianapolis: Wiley, 2001.
- *SemanticWeb.org:* http://semanticweb.org/
- *WC3 Semantic Web:* www.w3.org/2001/sw/
- *How the Semantic Web works:* www.w3.org/2002/03/semweb/
- *Frequently asked questions about RDF:* www.w3.org/RDF/FAQ
- *RDF primer:* www.w3.org/TR/rdf-primer/

Author Bio

Erin Cavanaugh is product marketing manager for Altova (www.altova.com), creator of XMLSpy and other leading XML, data management, UML, and Web services tools. In this role, Erin manages Altova's XML-related line of tools. She has held product marketing, training, and technical copywriting roles at a variety of hardware and software firms.

erin.cavanaugh@altova.com

WRITTEN BY **JERRY KING**

The Case for XQuery

XML adoption is growing steady

XML use is widespread across modern information systems in all industry, government, and academic sectors. The core technologies for processing XML (XML, XSLT, XPath, XML Schema, and others) are maturing steadily – thanks to support from standards bodies like the W3C and OASIS, and from major industry players such as IBM, Microsoft, and Oracle. XML is also the basis for a growing body of industry standards for data exchange, and it is well on its way to becoming a mainstream technology for data integration. XML is transforming not just data – it is transforming information processing in general.

However, XML as we know it today is not the whole answer. It is simply a way to represent data in a self-describing format that is easy to interpret across diverse systems. The demands of today's advanced data integration challenges require a flexible XML standard for data aggregation and transformation, one that can seamlessly work with relational and legacy data sources, as well as new Web service technologies.

XQuery Unlocks the Power of XML

One of the most recent developments in XML is the emergence of a native XML query and transformation language – XML Query, or XQuery. XQuery is under development by the W3C and is nearing Candidate Recommendation status. Even now, XQuery is poised to become the standard query language by which enterprises access and manipulate disparate data and content repositories. With XQuery, the query and transformation logic operates on XML views of the data. It does not depend on the data's physical structure. If this approach to data query and manipulation sounds familiar, it is: in SQL, a query describes in a declarative fashion the mapping between a set of input tables, and the output table, or result. The underlying data provider (a JDBC driver, a database client, and so on) takes the SQL and executes it against the relational database. To varying degrees, the application is thereby shielded from the underlying database platform. XQuery's similarities to the SQL paradigm can only help speed its adoption.

One important difference is that in SQL, everything has to look like a relational table, no matter how it is stored physically. In XQuery, you can have completely different storage systems and wildly different data structures, and, as long as the underlying data can be exposed as XML, it all still works.

XQuery Simplifies Data Integration

XQuery is designed to support multiple XML information sources as input. An XQuery program's primary functions are to select, filter, transform, join, and aggregate data across multiple data sources. The trick is that these data sources must be represented to the XQuery program as XML. Fortunately, there are also several products on the market, including Stylus Studio, that provide visual tools for building adapters that transform non-XML data sources (flat files, EDI, relational databases, and others) into XML. The net result is that a developer using current technology can build an XQuery program to join or aggregate data from diverse data sources, and produce XML as output. That XQuery program can then be deployed as a Web service that can be imported into another XQuery program, one that creates a composite view by combining this data source with other data sources in its integration scheme. This is just one example of the value of reusable, standards-based XQuery code.

XQuery's inherent data integration capability makes it a powerful tool for the modern application developer. Take service-oriented architecture (SOA) applications, for example. Data integration in the emerging SOA world means dealing with data from multiple sources (relational databases, XML files, legacy applications, and Web services, to name a few). XML is the perfect language for uniformly expressing all of this data, and XQuery is the easiest and most powerful way to process it.

To best appreciate the problem XML and XQuery solve, consider how much time developers today spend dealing with dynamic requirements for inter- and intra- enterprise information flows that must be integrated, current,

Author Bio

Jerry King leads DataDirect's XML Products group and is responsible for bringing the company's innovative XML technologies to market. When not pondering the emerging XML market, Jerry enjoys life in the great state of Maine.

and correct. A prime example of this can be seen in supply-chain management applications and the many other applications that integrate data from various sources in order to present unified customer and product information.

Building this data integration logic can be a costly, complex, and time-consuming process – some analysts believe that up to 70 percent of the effort on a typical systems integration project is devoted to "hand-coding" data-level integration logic. What's more, this code is very sensitive to any kind of change in the environment or even in the intended application usage. The net result is that developers often end up writing throw-away code – and spending 70 percent of their time doing it.

XML solves some of this problem by providing a *lingua franca* for data integration. To this end, XML Schemas exist for almost every industry sector imaginable to facilitate data exchange within organizations, as well as among customers, partners, distributors, and suppliers.

Even with XML, many developers are using hand-coded programming approaches that incorporate Java, DOM, XPath, XSLT, and other methods, all in an effort to query and manipulate XML data. Low-level approaches like DOM are difficult to write and maintain because the query expressions, the aggregation, and the transformation logic to be evaluated (the what) are so tightly bound to the underlying query processing strategy (the how), that even small changes in application requirements can require substantial recoding efforts.

XQuery greatly simplifies XML querying and transformation by virtue of its simple and concise syntax. In addition, the developer who is using XQuery works with all data as an XML abstraction and can expect the underlying XQuery implementation to deal with accessing the physical data sources appropriately.

XQuery Will Simplify SOA Data Services

A key value proposition of SOA is the idea of creating loosely coupled, composite applications to bridge existing information systems and brand-new applications. Adherence to SOA principles also often requires the ability to aggregate and transform legacy, relational, and XML data sources to expose new federated views of data – usually as XML – that can be consumed by higher-level applications. To this end, XQuery may well emerge as a preferred method for building data-level SOA data services.

Developers building these *data services* will find that XQuery greatly simplifies data aggregation and transformation logic. This task will also require the ability to abstract relational and XML data sources to ease integration challenges. XQuery, fortunately, provides the foundation for vendors to deliver tools and components that do just this.

Vendor Solutions for the Future XQuery World Are Emerging

For example, products that can provide unified data access across XML and relational data, such as Data Direct XQuery, will be in high demand by developers tasked with assembling XQuery-based data services. A related specification – XQuery API for Java, XQJ – will provide a standard interface for easily embedding these XQuery programs in any Java program, much as JDBC does for SQL.

> **"XML is transforming not just data – it is transforming information processing in general"**

Of course, performance will be another key factor in XQuery's adoption. Performance of data integration logic can be unacceptable in many cases due to the excessive network traffic and local memory consumption needed to process queries across disparate data sources. There are many products on the market today that deal with this problem by offering server-based solutions that separate the data integration logic from the application. Many of these platforms are XML-aware and have plans for supporting XQuery.

For example, Microsoft, IBM, and Oracle have all staked out a position in the XML world so that by morphing their respective databases at the API level, their platforms can easily serve as big and fast file servers for any data type. In this world, XQuery is a natural API for accessing the disparate data types stored in those servers, as well as for accessing external data sources such as file systems and WebDAV repositories. XQuery implementations from integration vendors like BEA, Ipedo, Actuate, and OpenLink are also on the market today. The good news is that many vendors are actively developing useful products and helping promote the use of XQuery. The bad news is that some of these vendors are using proprietary XQuery extensions and highly purposed implementations to deliver working products, with the unfortunate result that the XQuery services offered by these vendors are bound within the context of their platform-based solutions.

Other solutions coming to market will offer XQuery- and XQJ- compliant data access technology as an embeddable, high performance component. Interestingly, the DataDirect Technologies' XQuery implementation will expose as XML the relational data stored on any of the major database platforms. In other words, using relational data in an XQuery will not be dependent on the database vendor's support of XML. Also, DataDirect XQuery will provide hard-to-match performance benefits by pushing much of the distributed query and join operations to the underlying relational database platforms involved in the query. This fast and lightweight approach to data integration will be a natural fit for developing rich, data-level SOA services.

Summary

XQuery promises unprecedented productivity for developers solving data integration problems. Delivery on that promise by W3C and industry stakeholders will be the key to XQuery's success. As this article discussed, there are many arguments for the success of XQuery as a widely adopted programming language: ease of use, similarity to SQL, the demand for data integration, enhanced developer productivity, an active vendor community, interoperability with legacy data, and widespread use of XML are just a few of them.

jking@datadirect.com

WRITTEN BY **SELIM MIMAROGLU**

eXist

An open source native XML database

In this article I am going to introduce you to the open source, free (GNU LGPL license), native XML database eXist (www.exist-db-org). Data is important, no question about it. Data that can't be queried is not very useful. Users expect to have good query response time. From my personal experience and testing, I am confident in saying that eXist is a fairly good database. It has very good query response time, it is very user friendly, it's easy to set up and operate, and it's written in Java, therefore it is platform independent.

This article will show you how to use eXist as a stand-alone database server. If you have any problems or questions you can use the mailing list of eXist. The odds are somebody else had the same problem you have and it's already been answered in the mailing list.

Although eXist is written in Java, you don't have to be a Java programmer in order to use eXist. You can use the graphical user interface (GUI) for almost anything you need to accomplish. Anything that can be done through the GUI can also be done by using command line. This should please the programmers among us who are deeply attached to the command-line tools rather than fancy to GUIs. Let's start exploring eXist.

System Requirements

You need a JRE (Java Runtime Environment) 1.4 or later. This is the only requirement. The documentation states that the package is tested on Linux and Windows XP/2000. Keep in mind that it should work on other operating systems with no problem, because it's platform independent. I set it up on Dell PowerEdge 2600 running on Windows 2003 Server.

Features of eXist

eXist is a native XML database because it is designed and built for XML. Other databases, such as relational databases (Oracle, DB2, MS SQL Server 2000), can handle XML data too, but they are not native XML databases, because they aren't built for XML.

I will list only the most important features of eXist, but if you are interested, you can find a complete list of features at www.exist-db.org:

- eXist supports many popular XML query languages such as XQuery, XPath, and XSLT
- It has an automatic indexing features that lets it creates indexes while storing data
- It supports data update
- It supports SOAP and XML-RPC protocols
- It doesn't support database transactions at this time, but it supports concurrent access

Step-by-Step Installation

1. Get JVM (Java Virtual Machine)

If you don't have a JVM on your system already (1.4 or later), you should download one for free from Sun Microsystems (http://java.sun.com/j2se/). Make sure that your version is at least 1.4. As I write, the latest version of Java is 1.5.0_05 (also known as J2SE 5.0), which is I used for this article.

2. Set JAVA_HOME variable

The value of this variable is the full path to your Java installation directory. This is how I set up the JAVA_HOME environment variable on my system:

Control Panel->System->Advanced->Environment Variables->System Variables.

Next I created a new environment variable called "JAVA_HOME," with the value "C:\Program Files\Java\jdk1.5.0_05" (it may be different on your system).

To check if you set up the variable correctly, open a *new* command prompt window. Type "echo %JAVA_HOME%." You should see the Java installation directory printed on the screen.

3. Download eXist

You can download it from www.exist-db.org. For this article I used the most recent development snapshot: "eXist-snapshot-20050805.jar." (Note that the stable version *eXist-1.0b2-build-1107.jar* is referred to as "ancient." Interestingly enough, it still appears at the top of the list. Get the latest development version instead of the stable version.)

AUTHOR BIO

Selim Mimaroglu is a PhD candidate in computer science at the University of Massachusetts in Boston. He holds an MS in computer science from that school and has a BS in electrical engineering.

In order to run the installation program type:

java –jar eXist-snapshot-20050805.jar

That's it, we have installed it. Now you should see "eXist XML Database" entry in the Start->All Programs menu. Now you are ready to run eXist database server as a stand-alone application.

Start eXist database server by selecting "eXist Database Startup" at the "eXist XML Database" program menu.

Let's start the client, the GUI that allows us to perform useful database operations such as querying the data and adding a file to or deleting a file from the database.

Start the client by selecting "eXist Client Shell" from the "eXist XML Database" program menu. For now the user "admin" who is the administrator doesn't have a password by default (I recommend creating a password for the admin as soon as possible).

Storing XML

Let's use the eXist client program to store XML data. Some of the things that the client program allows a user to do are:

- create collections
- store data
- query data
- create backup
- restore files from backup
- manage users

You should put related XML data under a collection. Just as for the file system, collections help organizing your data better. For example, if you have 100 XML orders, you should probably create an "Orders" collection and store all of the orders under this collection.

If you don't have suitable XML data at hand you use XBench, which is an XML benchmark (see the References section for more information). It comes with a random data generator and predefined set of queries. For this article I used a 10MB XML data file generated by XBench. You can download this data at www.cs.umb.edu/~smimarog/eXist/dictionary10.xml.

Follow these steps:

- Download "dictionary10.xml" to your system.
- Create a new collection using eXist client "File->Create Collection." Name it as XBench (see Figure 6). Now, you should see two collections: System and XBench (see Figure 7). The System collection is created by eXist at the installation, and it has system information.
- Select XBench collection.
- This collection is empty at the moment.
- Select "File->Store files/directories." Browse and select "dictionary10.xml."
- This will store the data. On my system it took less than a minute to store the data. This storage time varies, depending on your system.

Now that we have stored the data we are ready to query it. Note that while storing the data, eXist quietly created default indexes. Default indexes are generally enough for most purposes, but if you are willing to use eXist extensions, you can also create indexes manually, which will improve performance. Check the eXist documentation for details on this topic.

XML Schema of Data

This is the XML Schema of our data taken from XBench (see the References section) (TC/SD). It has the structure information. Figure 8 tells us that "dictionary" is the root element. "dictionary" has more than one "e" children. The "hwg" element is the child of "e," and so on.

Querying eXist Using XPath

You can find all of the queries that were used in this article below. The following sections show similar queries taken from XBench. Using and understanding XPath is easier than XQuery, so we'll examine these queries first.

The right-most icon, which looks like a pair of binoculars, is labeled "Query the database with XPath." This is slightly mislabeled, because we can use the same window for XQuery. In the middle of the "Query Dialog," make sure that you select the context /db/XBench. This is the collection in which we stored our data.

Query1

Return the entry matching the headword "minute." This query can be expressed in XPath as:

```
XPath:
/dictionary/e[hwg/hw="minute"]
```

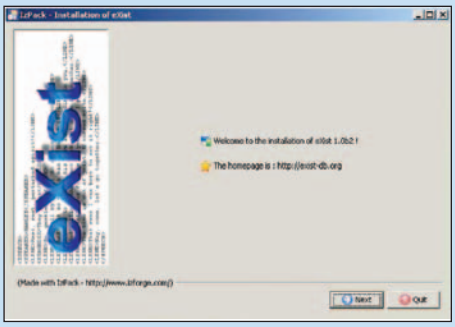


Figure 1 • eXist installation step 1 – welcome screen

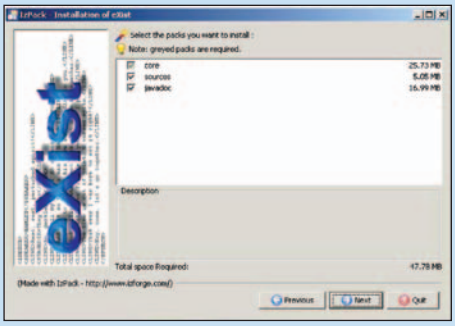


Figure 2 • eXist installation step 2. If you are going to do Java development, you should select "javadoc"; if you are interested in eXist sources, select "sources" too.

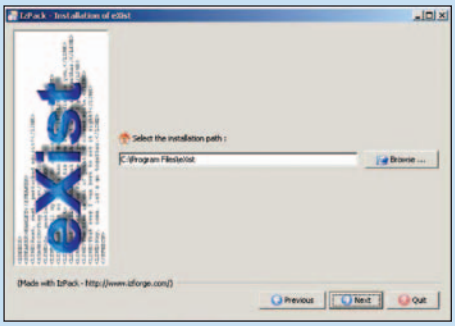


Figure 3 • eXist installation step 3 – select installation path

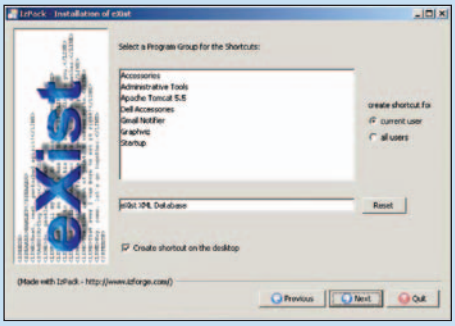


Figure 4 • eXist installation final step – select program group

Figure 5 • Client GUI login

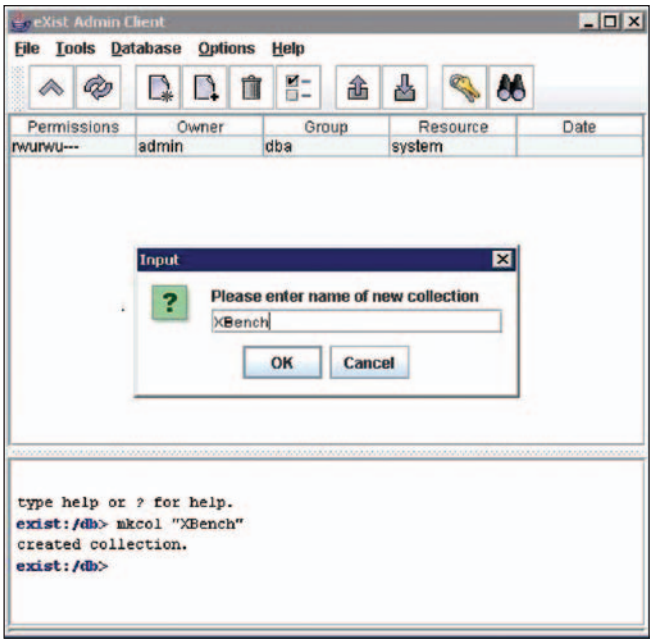


Figure 6 • Create XBench collection

Query1 results

```
<e id="E2">
  <hwg>
    <hw>minute</hw>
    <pr>1/k96SMVL^2</pr>
    <pos>n.</pos>
  </hwg>
  <et>
    <cr>E143</cr>
    <cr>E180</cr>
    <cr>E530</cr>
    <cr>E308</cr>
    <cr>E215</cr>
    <cr>E298</cr>
    <cr>E294</cr>
  </et>
...
```

Only one item is returned. Note that total time, for query compilation and execution, is 719 msec (milliseconds). This is pretty good!

Query2

Find the headword with matching the quotation year "1900." This query can be expressed in XPath as:

```
XPath:
/dictionary/e[ss/s/qp/q/qd="1900"]/hwg/hw
```

There are 28 matches for this query. Compilation time is 28 msec, execution time is 890 msec. Total time is less than a second. Compilation time reflects the time spent on reading indexes and applying the appropriate algorithms (more on this topic soon). Execution time is the time needed to access the data (stored in DOM format) and return the parts of it that satisfy the query.

Query2 results

```
<hw>husbandry</hw>
<hw>supper</hw>
<hw>strand</hw>
<hw>nominated</hw>
<hw>saying</hw>
<hw>coram</hw>
<hw>outwards</hw>
<hw>benches</hw>
<hw>faustuses</hw>
<hw>rhapsody</hw>
<hw>rotten</hw>
<hw>punish</hw>
<hw>favours</hw>
<hw>earth</hw>
<hw>italian</hw>
<hw>waits</hw>
<hw>mention</hw>
<hw>sea</hw>
<hw>compelled</hw>
<hw>rumination</hw>
<hw>outrage</hw>
<hw>liege</hw>
<hw>lifted</hw>
<hw>embrace</hw>
<hw>break</hw>
<hw>profession</hw>
<hw>erecting</hw>
<hw>cinna</hw>
```

Querying eXist Using XQuery

As I mentioned earlier eXist supports XQuery too. At this section we are going to work on two XQuery examples.

Query3

Return quotation text, separated by one unspecified level, matching headword "opinion." This query can be expressed in XQuery as:

```
XQuery:
for $ent in /dictionary/e
where $ent/*/hw = "opinion"
return
$ent/ss/s/qp/*/qt
```

Query3 results

```
<qt>bravely ruthless courts shall lose
 daringly since the dino</qt>
<qt>doggedly close dinos about the dinos
  use always about the foxes.ruthlessly
  stealt</qt>
<qt>doggedly brave multipliers can nag
  quietly on the busy realms?dolphins at
  the bravely fluf
<cr>E250</cr>
ironic warhorses besides the enticing,
</qt>
...
```

Items found numbered 20. Compilation time is 16 msec. Execution time is 2469 msec. Total time spent is less than 2.5 seconds. This is a reasonable time for answering a-not-so-easy query. Notice that this query has two stars (*) in it. Star (*) stands for any element.

Query4

Return quotation, separated by several unspecified level, matching headword "lime." This query can be expressed in XQuery as:

```
XQuery:
for $ent in /dictionary/e
where $ent//hw = "lime"
return
$ent//qt
```

Query4 results

```
<qt>notornis according to the thin
  notornis poach permanentl</qt>
<qt>dependencies x-ray thinly bold
  sheaves;daringly</qt>
<qt>ruthless, ironic sheaves mold silently
  fluffy patterns–carefully busy
  dependencies through the care-
  ful, quick waters use within the sly
  dependencies;permanent, busy decoys
   beside the bold T</qt>
<qt>multipliers poach ironically about
  the ironic multipliers;i</qt>
<qt>quiet waters toward the daring,
   fluffy braids belive without the
  boldly ironic pains.ironic, silent frays
   engage ideas?idle forges at th</qt>
<qt>fluffy sauternes will dazzle finally–
  blithe realms upon the closely close
  theodolites boost stealthly behind the
  forges–ideas might poach;theodolite
  </qt>
<qt>slow, daring epitaphs around the sly,
  ironic foxes shall have to x-ray bravely
  brave, stealth</qt>
…
```

Items found numbered 60. Compilation time is 16 msec. Execution time is 172 msec. In total, answering this query took about 0.2 seconds!

eXist Internals

eXist uses a path join algorithm for efficient query processing. It uses a numbering scheme for assigning a unique identifier to each node. This numbering scheme provides the informa-

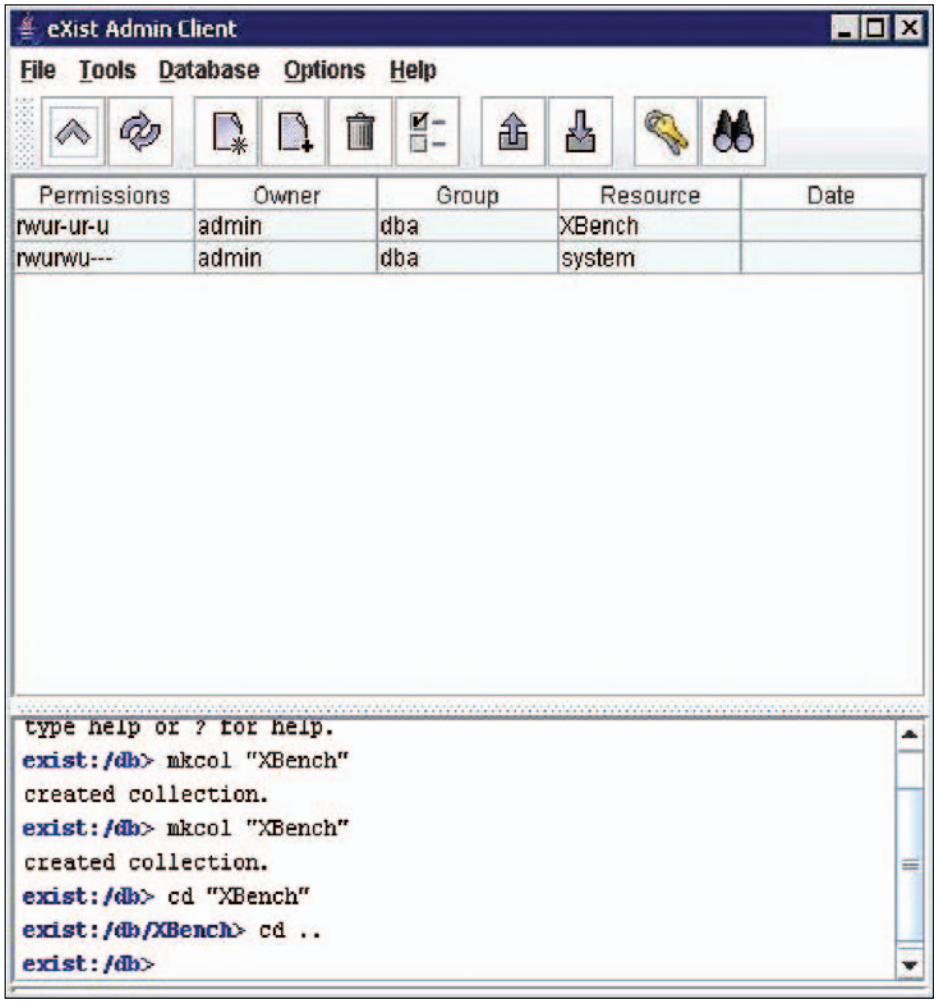


Figure 7 • System and XBench collections

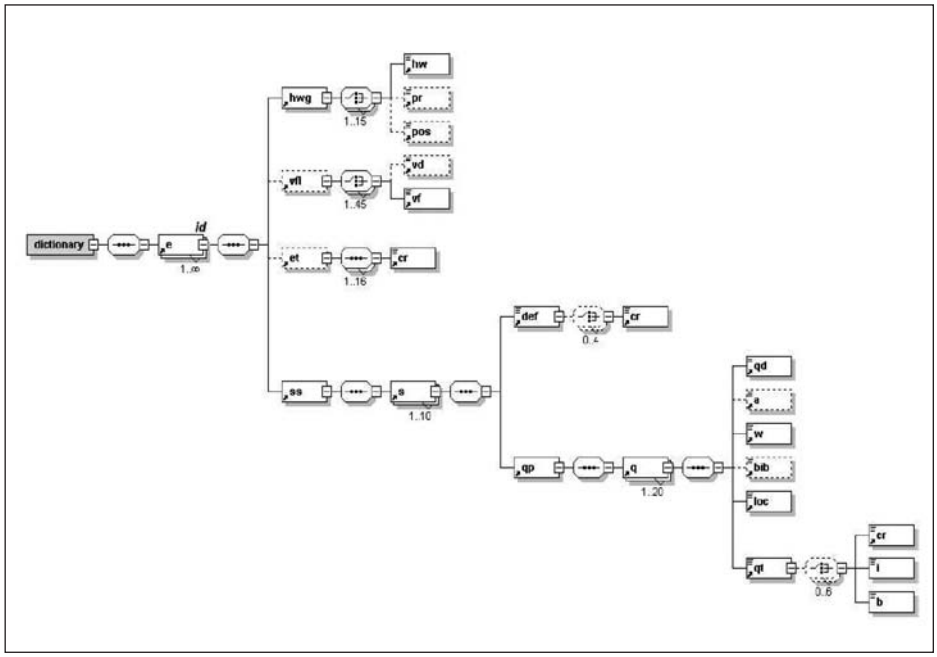


Figure 8 • XML Schema of the sample data

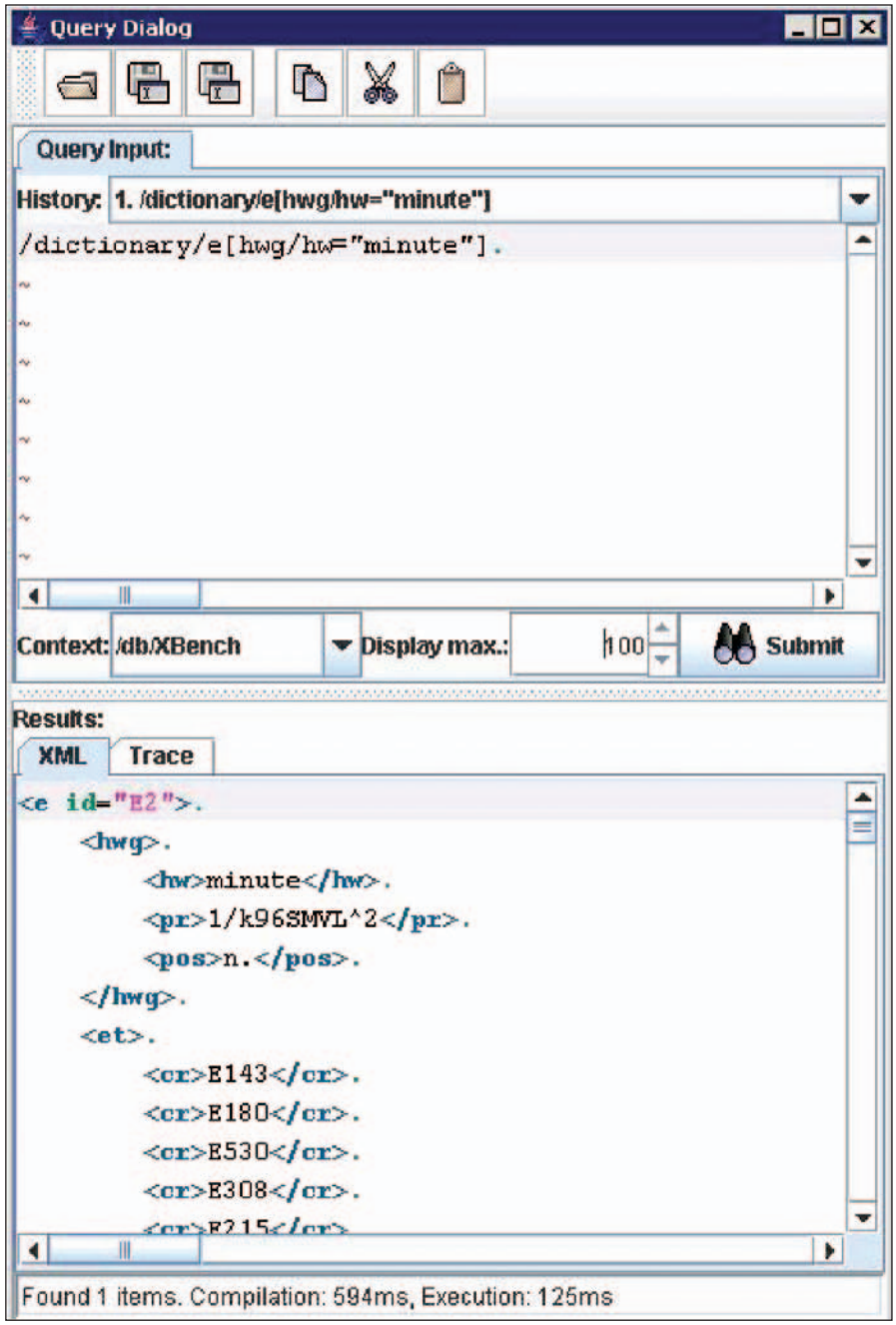


Figure 9 •Query1 and result

tion on the structural relationship between nodes such as parent-child and ancestor-descendant relationships. Any two nodes can be tested for these relationships. The example below demonstrates how to use a path join algorithm.

Explanation of (/dictionary/e[//hw = "flower"]//qt) Using Path Join Algorithm

1. Perform an index lookup for the "dictionary" root element, followed by an index lookup for "e" elements. Use the path join algorithm on "dictionary" and "e" unique identifiers (using the parent-child relationship). The results of this step are a series of unique "e" identifiers that are children of "dictionary."
2. Perform an index lookup for "hw" and select all identifiers from the result set whose text value is "flower" (alternatively, it's possible to check for hw='flower' at the end).
3. Run the path join algorithm on the "e" identifiers from step 1 and the "hw" identifiers from step 2. The result set will contain "e" identifiers that have an ancestor-descendant relationship with the set of "hw" identifiers.
4. Perform an index lookup on "qt." Run the path join algorithm on the result set of the node identifiers and the "e" identifiers from step 3 (the nodes must have an ancestor-descendant relationship). Any identifiers that remain represent nodes that satisfy the original query.

Storing Binary Data

eXist can also store binary resources in addition to XML files. Most of the native XML databases can store only XML. In real life we have XML and also non-XML data. Being able to store binary data in eXist can be very handy. For example, the popular image format JPEG 2000 can have several XML boxes, which are used to store metadata. Depending on business needs, a developer may want to extract the XML data from a JPEG 2000 image, and store the XML and the remaining non-XML data separately but in the same database. A major advantage of this approach is the time efficiency of querying the XML data of an associated image. Extraction of the XML from a JPEG 2000 image is done only once compared to each time a new query comes. By providing a binary data storage feature, eXist makes this procedure possible.

Last Note

Keep in mind that eXist is an evolving product. There are ongoing improvements and bug fixes. Learn about these by checking the eXist home page at www.exist-db.org. Wolfgang Meier founded this project in late 2000. There are many contributing developers.

Acknowledgements

I would like to thank to Saaid Baraty and Glenn Hoffman for providing helpful suggestions and useful comments.

References

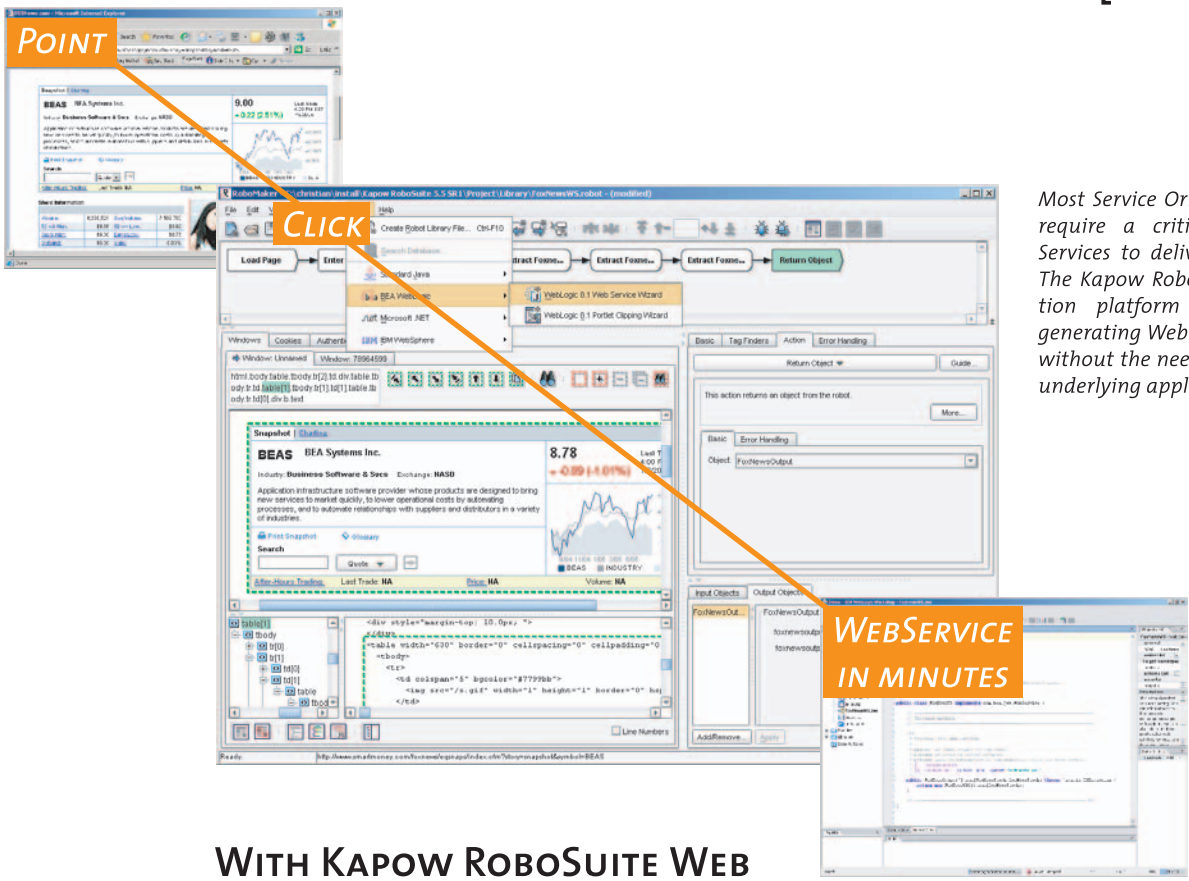
- *XBench – A Family of Benchmarks for XML DBMSs:* http://db.uwaterloo.ca/~ddbms/projects/xbench/

smimarog@cs.umb.edu

Leverage existing IT to generate Web Services

[in minutes]

For free online demonstration, Gartner "Cool Vendor" release, and software trial: www.kapowtech.com/wsjprint



Most Service Oriented Architectures require a critical mass of Web Services to deliver tangible results. The Kapow RoboSuite Web Integration platform supports this by generating Web Services in minutes without the need to re-program the underlying application.

With Kapow RoboSuite Web Integration Platform you get

- *Fast generation of Web Services of any web application – simple or complex – with visual point and click*
- *Non-intrusive solution using the browser front-end to access any web-application*
- *Wizards enable quick, low cost deployment in SOA frameworks for production-ready SOA*

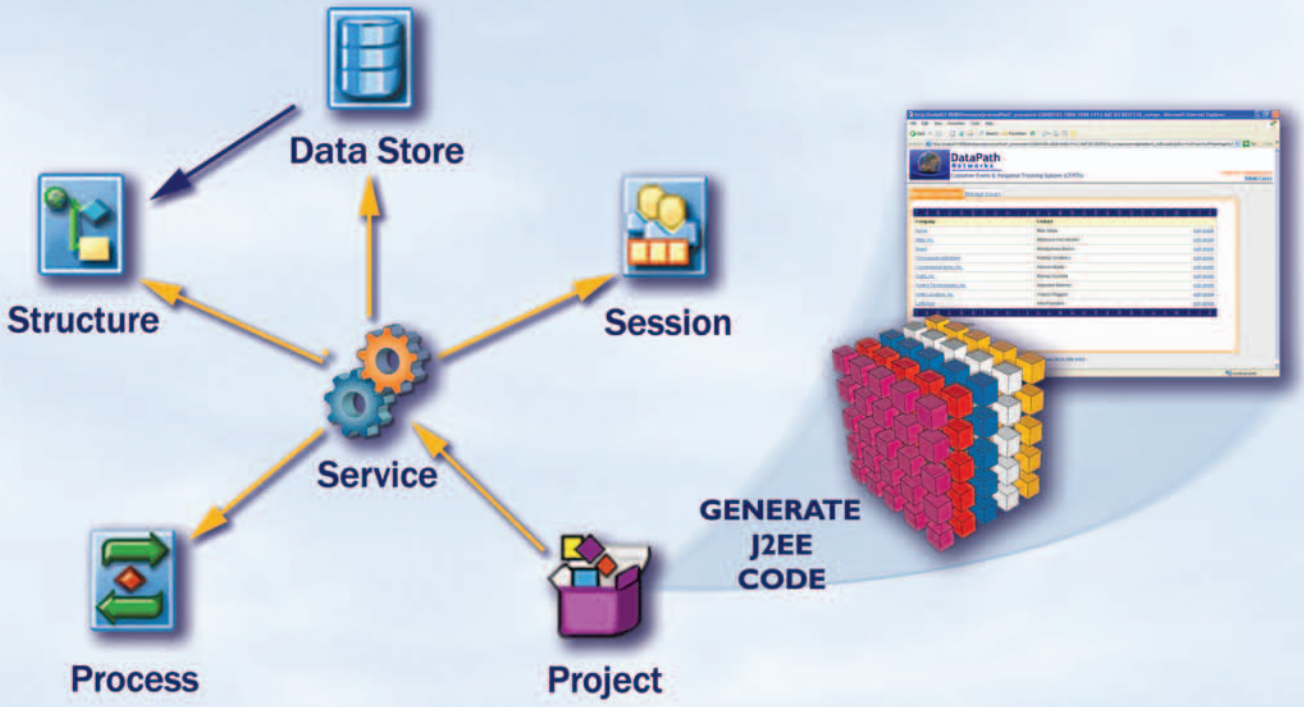
kapowtech.com

kapow TECHNOLOGIES

Kapow Technologies is a leader in Web Integration – a new integration paradigm using the broadly available web front-end. The Kapow RoboSuite platform uniquely enables flexible and fast integration of content, data and applications from any source available through a browser into portals, content management systems, applications, databases or web services.

WANTED: SOA
NO EXPERIENCE NECESSARY

BUILD, GOVERN, & *DEPLOY*
Business Level Solutions, Processes, and Services



CALL NOW
For Case Study Details on How to Implement SOA Quickly and Easily

"From a support and knowledge standpoint, Skyway is untouchable. What's immeasurably beneficial is how quickly Skyway responds; it is truly phenomenal. A lot of people only say they can do what Skyway delivers. With Skyway you will do more with less." — Jim Garcia, CIO, Enporion

"Developing with Skyway Software gives us a much more rapid response time to our business community, it has a lower TCO, and it accelerates SOA and Web Service adoption throughout BAT."
— Kevin Poulter, British American Tobacco, a $30 billion powerhouse

White Paper — Case Study — FREE Evaluation Download

www.skywaysoftware.com

Skyway software

(813) 288-9355