SYS-CON

NOW INCORPORATING XML JOURNAL

Web Services JOURNAL

.NET J2EE XML

May 2005 Volume 5 Issue 5

BizTalk Server 2004: Too Hot to Handle?

Where to successfully apply BizTalk Server technology

pg.24

Effective Business Process Management Software Puts Business People in Control

Placing process design, execution, and management into the hands of people who use it

pg.28

Large Scale B2B via Web Services

Scaling mission-critical Web service for hundreds of partners

pg.30

BPM THEORY FOR LAYMEN

Theory matters! The scenic tour of the Pi Calculus and Petri nets for BPM and choreography practitioners

pg.10

RETAILERS PLEASE DISPLAY UNTIL JULY 30, 2005

$6.99US $7.99CAN

05>

71486 03420 9

Parasoft SOAPtest®

Keep Your Skills Ahead of the Crowd

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

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

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

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

*– **Adam Kolawa**, Ph.D.*
Chairman/CEO of Parasoft

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

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

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

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

For Downloads go to www.parasoft.com/soaptest_WSD PARASOFT®

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

Features

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

Benefits

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

Protocol Support

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

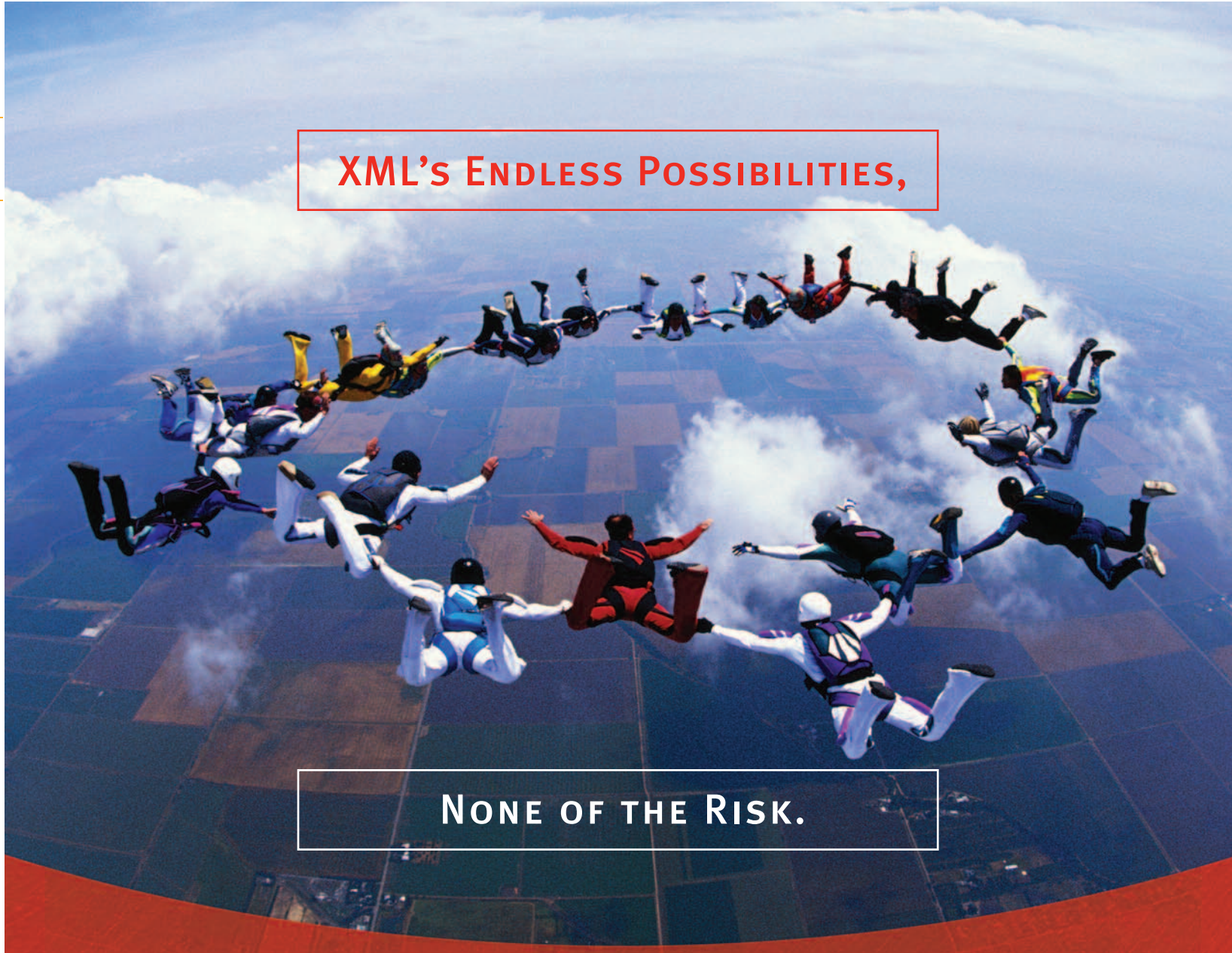
Platforms

- Windows 2000/XP
- Linux
- Solaris

Contact Info:

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

www.parasoft.com



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

WebServices .NET J2EE XML JOURNAL

May 2005 Volume 5 Issue 5

XML JOURNAL

inside WSJ

WebSphere.

Key

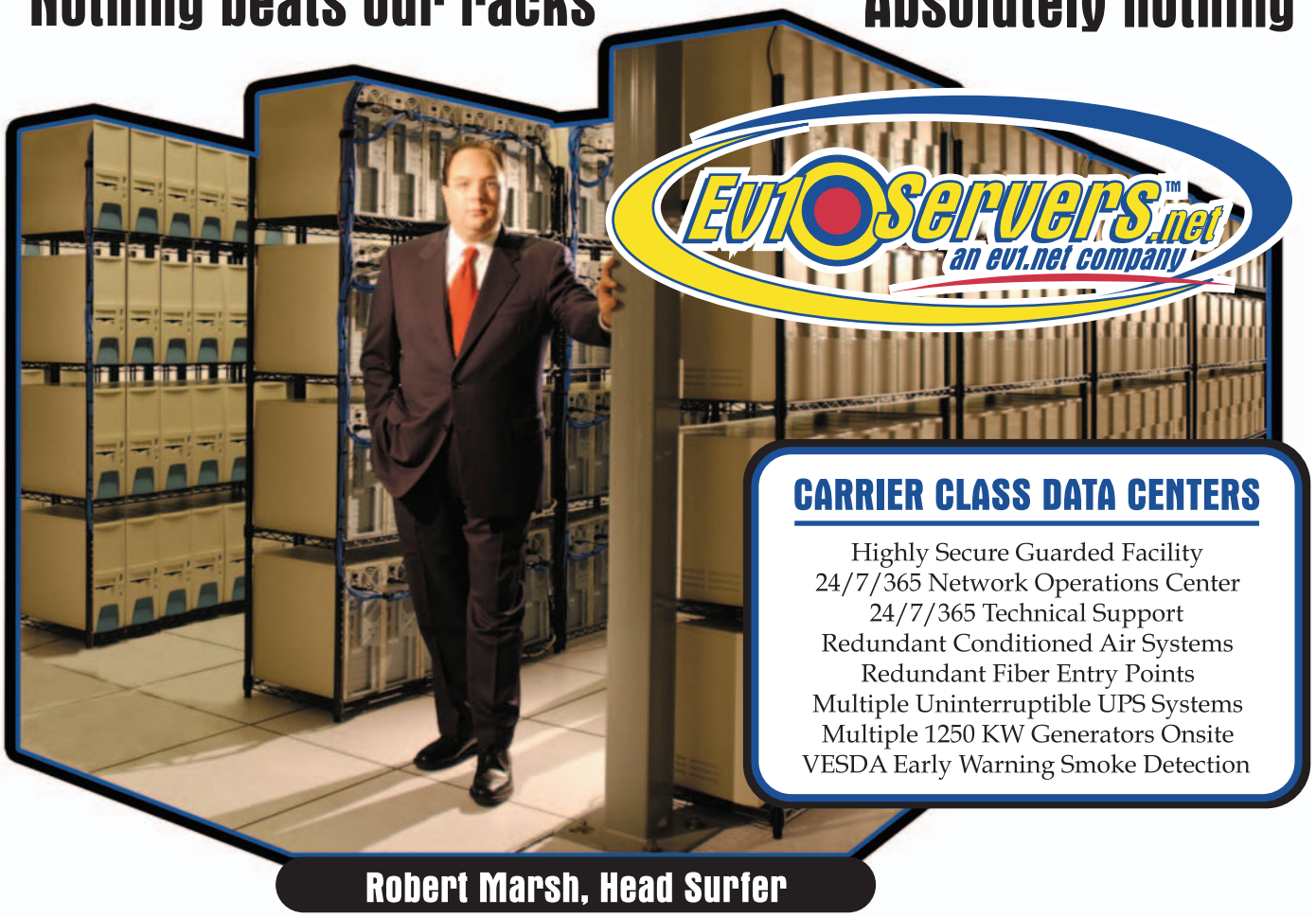
1. Deductible viewed quickly.
2. Claim info filed accurately.
3. Vendor receives complete data.
4. Quotes researched easily.
5. Great service boosts policy sales.

MIDDLEWARE IS IBM SOFTWARE. Powerful WebSphere software. For an integrated IT environment that makes your business more flexible. WebSphere delivers a secure, scalable platform. Enabling a service-oriented architecture. So your business can respond faster. More efficiently. To partners, vendors and customers. With no ripping. No replacing. No headaches. No kidding.

Middleware for the on demand world. Learn more at **ibm.com**/middleware/flexible ON DEMAND BUSINESS™

Nothing beats our racks

Absolutely nothing



Robert Marsh, Head Surfer

CARRIER CLASS DATA CENTERS

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

START YOUR OWN WEB HOSTING BUSINESS TODAY!

from **$299*** *Instant Activation!*

Dedicated Server

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

Over 20,000 Servers!

1-800-504-SURF | ev1servers.net

PLESK7 RELOADED
Preferred Control Panel

IP Compliant. Price subject to change. Quantities Limited.
*Per month. Set-Up fees apply. See web site for complete details.

WebServices 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

A Matter of Perspective

WRITTEN BY
SEAN RHODY

Sometimes the old clichés are the best ones. I've been working with a number of different groups and companies recently, defining and refining their takes on service-oriented architecture and Web services. In thinking about the different perspectives necessary for a full understanding of how to use Web services, SOA, and business process to achieve business agility, I was struck by phrases like "forest for the trees" and "50,000-foot view." As usual, what you think about Web services and how to use them depends on what you do and how it affects your day-to-day work.

Software architecture, by which I mean the design of software systems according to patterns, best practices, and business conditions, can look wildly different depending on your role in the process. An architect charged with building a stand-alone application may make decisions that an architect building an enterprise infrastructure may think are ludicrous – and they may be, at the "50,000-foot level."

Look at the management of identities, for example. For a stand-alone application, this may be as simple as a logon ID and password combination maintained in a database table. That may be perfectly reasonable for the application architect, based on requirements.

However then the enterprise architect steps in. Charged with achieving high levels of integration and reducing costs, the enterprise architect typically would look for a shared service model that will allow common routines and functions, such as security, to be defined and managed once.

Here we have the classic conflict – Application versus Service, mainly because I described the "traditional" role of the application architect, which is to design stand-alone applications.

In a world where every new system needs an ROI justification and staff never increases along with work, SOA is an answer organizations are turning to for productivity enhancements. But in order for SOA to assist in this fashion, perspectives have to be adjusted throughout the chain of command.

One of the biggest changes is the move from building to assembling applications. Developers no longer build applications, they build services. These services are assembled using business process management to form business processes. The application builder now provides an interface for interaction with business processes by assembling services and processes as needed to complete the particular task.

The job of the SOA enterprise architect is to design and designate all of the services necessary to support the business processes. Part of this job entails understanding the various specifications for Web services that are available and deciding how to use them (or even whether to use them) to support activities across the board. In the case of security, it might mean selection of a security management system and the use of SAML, XACML, and other standards. Then these standards have to be designed for appropriate usage, such as determining what level of security is required for each service.

The question of achieving business agility from SOA is similar to that need for perspective between the application/service designer and the overall enterprise architect. An SOA enables integration and solves a number of long-standing but primarily technical issues (such as getting J2EE and .NET to talk meaningfully), but it doesn't completely address the needs of the business when it comes to being adept and open to change.

Business process management allows an organization to take a set of services and create business processes. A business process is a task (which may be complex and involve multiple touch points and interactions) assigned to one or more individuals and accomplished via software.

However to accomplish the goal of achieving business agility, we need another perspective – the business perspective. The ultimate goal of an SOA is to provide the business community with software that enables the agile enterprise and allows the business to make the changes it needs to cope with changing market conditions and regulatory issues.

Are we there yet? It's all a matter of perspective.

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
Dorothy Gil dorothy@sys-con.com
Kim Hughes Kim@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
Linda Lipton linda@sys-con.com

sys-con.com
CONSULTANT, INFORMATION SYSTEMS
Robert Diamond robert@sys-con.com
WEB DESIGNERS
Stephen Kilmurray stephen@sys-con.com
Percy Yip percy@sys-con.com
Vincent Santaiti vincent@sys-con.com

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

BPM: Too Much or Too Little?

WRITTEN BY
AJIT SAGAR

I'd like to take a moment to introduce myself. I've been working with SYS-CON for about eight years now, across different publications, so when Sean talked to me about providing regular content for *WSJ*, I thought to myself, "Cool." I am also the enterprise editor for *JDJ* – so you should see a lot of cross-magazine content. Part of what I want to bring in as "International Technical Editor" for *WSJ* is some articles from around the world.

So, the uppermost thought in my mind is that I just started jogging two days ago after a gap of several years. While I was cajoling my body to go along with my mind – tricking it into going a further distance than what we had mutually agreed upon, my thoughts started wandering to the commentary for this month. I also thought about the several client engagements that Infosys has been called in on to help the client determine how much of BPM to bring into their organization. I guess the two scenarios (jogging and consulting) have some commonality – the negotiation of "selling" more than the original agreement, and maybe offering questionable benefits (at least that is what my body told my mind!).

Recently I've been involved in initial discussions with folks from some large firms in the financial industry who are considering adopting BPM because of several reasons:

- It is the new buzzword
- There is promise of integration to a level never seen before
- Workflow did not fulfill the promise of enabling the agile enterprise – BPM will
- It is time to adopt SOA
- BPM will bring in the ability to do Web services.

Fortunately, the folks at the CTO/CIO level in the financial sector are driven by reason (and cost), and understand the workings of their existing portfolios enough to at least ask the question – what is it we are gaining, and is it worth the cost? As one of the CTOs of a large global investment banking firm mentioned to me in a recent meeting, "The vendors we are looking at promise functionality that will solve only 60% of my problem, and only 30% of the functionality they offer is useful to me." In other words, 40% of this problem remains unsolved and 70% of what he bought is useless to him. Furthermore, the cost of the products is in the millions of dollars. However many factors, including rigorous changes in compliance rules, have led people to look at BPM as the next big thing.

First of all, there is a mishmash of terms that confuse the whole issue. There is obvious confusion between workflow and business process management, process orchestration and service orchestration, enterprise application integration, and process integration. And vendors from each of the spaces have wrapped their offerings under the BPM umbrella; same pig, new lipstick. It used to be all about work management and integration. Now it is all about process management.

BPM is composed of five building blocks – a graphic process designer/modeler, a runtime engine, process monitoring and management capabilities, analysis tools, and an interface to modify the process at runtime. If you look closely, you will see that each of these has its humble (or grand) beginnings in the offering from a workflow vendor, or an application server vendor, and so on. At the end of it, you have to determine which part of the suite is really beneficial for you. Unfortunately the BPM standards haven't been adopted widely enough to mix and match offerings between vendors or with your own in-house components.

Don't get me wrong. I think BPM has a great future, but you need to choose wisely when you bring it into your organization. You need to proceed with due diligence and a rigorous cost-benefits-analysis. And yes, Web services can be leveraged at many levels without requiring a formal BPM tool; however, if you find yourself in the situation where you are building too much from scratch (or had little in-house IP to start with), consider a BPM tool.

About the Author

Ajit Sagar is a senior technical 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 75 articles.

ajitsagar@sys-con.com

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

Mindreef®
www.Mindreef.com

BPM Theory for Laymen

Theory matters! The scenic tour of the Pi Calculus and Petri nets for BPM and choreography practitioners

In most software topics, the boundary between theory and practice in software is clearly demarcated: theory is for academics who seldom descend from the ivory tower, practice is for industry professionals who have long forgotten the concepts and application of theory. In concurrency, for example, most developers either know or have programmed semaphores, but few remember the conceptual underpinnings devised by Dijkstra. But Business Process Management (BPM) – a key Web services technology with close ties to Web services choreography – belongs to a rarer category, in which theory informs practical design and theoretical jargon is part of the hype with customers.

WRITTEN BY
MICHAEL HAVEY

Somehow the abstruse terms "Pi Calculus" and "Petri net" – as impressive to the ear as database management's "relational calculus" or capacity planning's "Erlang formulae" – have permeated the consciousness of the BPM community. Many BPM onlookers are familiar with, and are interested in, Pi and Petri, but have at best a vague understanding of them.

This article is a guided tour of the scenic route of BPM theory. The Pi Calculus and the Petri net are studied by example, with just enough detail to give a sense of the BPM connection without overdosing on algebra.

Why Theory Matters

Process theory is practically important for several reasons:

- Theory is mentioned frequently in connection with BPM, even in nonacademic material. Countless presentations, for example, state without explanation that the leading BPM language, Business Process Execution Language (BPEL), is influenced by the Pi Calculus and Petri nets. For the many practitioners who are intimidated by the pedantic name-dropping but are curious to uncover its meaning (e.g., What is the Pi Calculus? What are Petri nets? Which parts of which theory are used in BPEL? Why is BPEL based on two theories rather than one?), this article helps elucidate the nature of the connection.
- BPM is relatively immature and benefits from the ideas and rigor of theory. Control flow, for example, is often treated too casually by vendors, who are more likely to emphasize ease of programming than semantic precision. Regrettably, as several papers on process design patterns have demonstrated, most vendors, and even most standards, struggle to support certain common control flow scenarios (e.g., the "multiple instances without runtime knowledge" and "interleaved parallel routing" patterns described in www.workflowpatterns.com). To build successful solutions, practitioners should insist on knowing exactly how a given process will run. To accomplish this, they should choose a good language and understand how the language works. Judged on the basis of control flow, the strongest languages are those based on the Petri net, notably BPEL and Business Process Modeling Notation (BPMN). And, arguably, to understand the nuances of

these languages (e.g., dead-path elimination in BPEL) requires an appreciation for the Petri net.

- Contemporary BPM and its cousin Web Services Choreography are obsessed with the construction of complex participant conversations. Choreography especially, because it is mandated to build global collaborative contracts, requires a conceptual framework that can express dynamic communicating processes precisely and concisely. The leading choreography language, Web Services Choreography Description Language (WS-CDL), bases its constructions (e.g., channel passing) on the Pi Calculus. BPEL is also alleged to have Pi underpinnings. By learning the basics of Pi, the practitioner gains insight into the use of these languages.

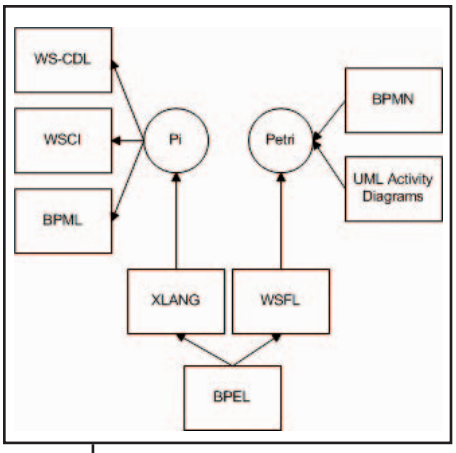


FIGURE 1 | **Theory family tree**

Family Tree

As Figure 1 shows, four major contemporary standards – WS-CDL, Web Services Choreography Interface (WSCI), Business Process Modeling Language (BPML), and XLANG – betray Pi Calculus influence, three – BPMN, UML activity diagrams, and Web Services Flow Language (WSFL) – derive from the Petri net, and one – BPEL – is a blend, inheriting traits of Pi and Petri from its parent languages XLANG and WSFL, respectively.

The Pi Calculus

Developed by the Scottish mathematician Robin Milner in the 1990s, the Pi Calculus is a formal language for defining concurrent, communicating processes, including, but not restricted to, business processes. In its detail the Pi Calculus is a rather advanced algebraic system requiring a senior level of mathematical training. Milner's presentation of the subject in his landmark paper "The Polyadic Pi-Calculus: A Tutorial" is written in the mathematical idiom of definitions, theorems, and lemmas, inaccessible to most BPM onlookers. And few business analysts or software developers could survive if required to compose their business processes as lines Pi Calculus code.

However somehow, despite its academic roots and its inherent complexity, the Pi Calculus has become one of BPM's most attention-getting cocktail party terms. Popular BPM literature states boldly that major languages such as BPEL and WS-CDL are based on the Pi Calculus. This stunning level of influence, charges a leading BPM commentator (Wil van der Aalst, in his paper "Pi Calculus versus Petri Nets: Let Us Eat Humble Pie Rather Than Further Inflate the Pi Hype"), is dubious, and surely nothing but hype.

Let the people who advocate BPEL4WS, BPMN, … and WSCI show the precise relation between the language and some formal foundation. People who cannot do this but still claim strong relationships between their language and e.g., Pi-calculus only cause confusion

("Pi Calculus versus Petri Nets," page 2).

Whether or not it is hype, the Pi Calculus-BPM connection merits a serious look. What, in a nutshell, is the Pi Calculus, how does it apply to BPM, and what is the extent and nature of its influence on contemporary popular languages like BPEL and WS-CDL?

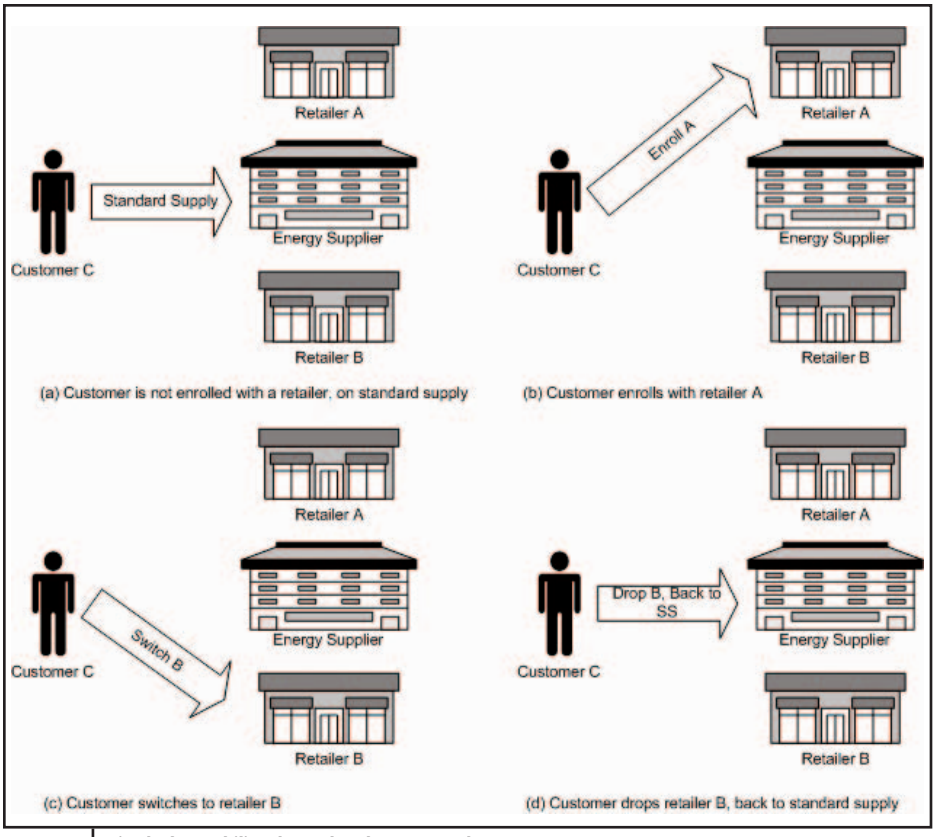


FIGURE 2 | **Pi Calculus mobility: deregulated energy market**

The Pi Calculus in a Nutshell

The Pi Calculus is a language used to define concurrent processes that interact with one another dynamically. Each process consists of one or more actions, which can be arranged sequentially, in parallel or conditional paths, or recursively. An action is either the sending or receiving of information on a channel. According to the Pi-Calculus convention, when one process sends to

another, it includes the name of the channel to be used for the other process to respond. This name is variable and, as we will see, can change in response to changing conditions.

One of the most distinctive features of the Pi Calculus is mobility, in which the topology of communicating processes changes dynamically in response to changing conditions. An example of mobility is the enrollment of customers with retailers in a deregulated energy market. In part (a) of Figure 2, customer C initially buys energy directly from the supplier (a "standard supply" arrangement), but in part (b) enrolls with retailer A. In part (c), the customer switches to competing retailer B, but then drops the retailer in part (d), thus returning to standard supply.

The source code to model this scenario is remarkably terse:

```
1 CustomerSS(enroll,switch,drop,rets)=
2    ∑(r:rets). (enroll r,"mike".CustomerR
(r,enroll,switch,drop,rets))
3 CustomerR(r,enroll,switch,drop,rets)=
4    ∑(r2:rets).(switch r,r2,"mike".Custom
erR(r2,enroll,switch,drop,rets)) +
5    drop r,"mike".CustomerSS(enroll,
switch,drop,rets)
6 Supplier(enroll,switch,drop)=
7    (enroll(r1,c).r1 "addcust",c +
8    switch (r1,r2,c).r1 "dropcust",c.
r2 "addcust",c +
9    drop(r1,c).r1 "dropcust",c.Supplier
(enroll,switch,drop)
10Retailer(r)=
11   r(action,c).Retailer(r)
12Market=
13   (new chEnroll,chSwitch,chDrop,retSet=
{retA,retB})
14   CustomerSS(chEnroll,chSwitch,chDrop,
retSet)|
15   Supplier(chEnroll,chSwitch,chDrop) |
16   Retailer(retA)|Retailer(retB)
```

The code defines five processes: *CustomerSS* (lines 1-2), *CustomerR* (lines 3-5), *Supplier* (lines 6-9), *Retailer* (lines 10-11), and *Market* (lines 12-16). Each process is written as a mathematical equation, with the left side and right side separated by an equals sign ("="). The left side gives the name of the process and the channels it uses to communicate with the other processes; the right side (indented in the code sample above) is the definition.

The first two processes model a customer. The first, *CustomerSS*, represents a customer on standard supply who enrolls with a retailer. The obscure expression *∑(r:rets).(enroll r,"mike".CustomerR(r,enroll,switch,drop,rets))* means:

- ***∑(r:rets):*** Choose a retailer channel *r* from the set of retailers *rets*
- ***enroll r,"mike":*** Send *r* and the customer's name "mike" on the channel *enroll*
- ***CustomerR(r,enroll,switch,drop,rets):*** Change state to that of an enrolled customer by calling the process *CustomerR*, passing the chosen value of *r*

The second option, on line 5, is *drop r,"mike".CustomerSS(enroll,switch,drop,rets):*

- ***drop r,"mike":*** Send *r* and the customer's name "mike" on the channel *drop*
- ***CustomerSS(enroll,switch,drop,rets):*** Change state to that of a customer on standard supply calling the process *CustomerSS*

The supplier process, in lines 6-9, listens on channels *enroll*, *switch*, and *drop* for messages from the customer. In the enrollment case, in line 7, the process first receives the message on the enrollment channel, using the notation *enroll(r1,c)*, and then sends a message to the channel of the specified retailer (*r1*) to add customer *c* to its customer base(*r1 "addcust",c)*. The drop case, in line 9, is similar, but this time the message from the customer arrives on the channel *drop* (i.e., *drop(r1,c)*) and the supplier's message to the retailer is to drop the customer (*r1 "dropcust",c)*. The switch case, in line 8, *switch(r1,r2,c).r1 "dropcust",c. r2 "addcust", c*, is a combination of drop and enrollment. It breaks down as follows:

- ***switch (r1,r2,c):*** Receive on the *switch* channel the channels of the new retailer (*r2*) and old retailer (*r1*), plus the customer name
- ***r1 "dropcust",c:*** Instruct the old retailer to drop the customer
- ***r2 "addcust",c:*** Instruct the new retailer to add the customer

> "For the BPM practitioner, theory, in the form of the Pi Calculus and the Petri net, actually matters!"

These steps are separated by the sequential operation ("."), and hence are performed one after another in the order given.

CustomerR is the process for a customer enrolled with a retailer, who can either switch to a different retailer or drop the current retailer and return to standard supply. The conditional operator ("+") separates these two options. The first option, on line 4, is written as *∑(r2:rets).(switch r,r2,"mike".CustomerR(r2,enroll,switch,drop,rets))*, meaning:

- ***∑(r2:rets):*** Choose a retailer channel *r2* from the set of retailers *rets*
- ***switch r,r2,"mike":*** Send *r2*, as well as the channel of the retailer with which the customer is currently enrolled, *r*, and the customer's name "mike" on the channel *switch*
- ***CustomerR(r2,enroll,switch,drop,rets):*** Keep the same state, but change the retailer of record, given by the first variable, from *r* to *r2* by recursively calling the process

When these activities are completed, the supplier process continues recursively (*Supplier(enroll,switch,drop)* on line 9), much like a daemon.

The retailer process, in lines 10-11, listens on its channel *r* for the required action and customer (*r(action,c)*) and then, like the

supplier process, continues recursively.

The market process, in lines 12-16, creates a simple energy market. Line 12 creates specific instances of the *enroll*, *switch*, and *drop* channels (*chEnroll*, *chSwitch*, and *chDrop*, respectively), as well as a set, named *retSet*, of actual retailer channels (*retA* and *retB*). Lines 14-16 create four concurrent process instances – separated by the concurrency operator "|" – and passes to them the newly defined channels. The processes include a single customer on standard supply, a supplier, and two retailers using channels *retA* and *retB*, respectively. Figure 3 depicts the overall exchange.

The Pi Calculus and BPM

BPM is fast becoming the practical study and design of solutions for elaborate, multi-company communicating business processes. For those seeking a formal basis for BPM processing, the Pi Calculus offers three key features:

- ***Control Flow.*** In the Pi Calculus, the sequential, parallel, conditional, and recursive behavior of a process can be declared succinctly.
- ***Message-Based Communication.*** The heart of the Pi Calculus is its clean syntax and semantics for inbound and outbound messaging.
- ***Mobility.*** Contemporary processes cannot hardcode service endpoints, but require the ability to pass around and change addresses dynamically. Dynamic addressing, or mobility, is the most distinctive feature of the Pi Calculus.

A good BPM language has the control structures of a flow chart, and has as its most significant steps "mobile" message-based interactions. The major contemporary process languages, as it turns out, are like this anyway. XLANG, for example, has similar control structures (sequence, all, switch), explicit service-oriented message support (the operation action), and dynamic channel bindings. WS-CDL is based on interactions, message exchanges on channels, and channel passing between participants. (WS-CDL, according to its author, is based on the Explicit SolosCalculus, a variant of Pi, which allows a system to be modeled from a global viewpoint. [See Kavantzas, "Aggregating Web Services: Choreography and WS-CDL."] Robin Milner, Pi Calculus creator, is an invited expert in the W3C Choreography Working Group.)

The crux of the "hype" criticism of van der Aalst is that the use of the Pi Calculus in the creation of contemporary languages is overstated; that, perhaps, these languages came together more casually and with less academic rigor than advertised. They might resemble the Pi Calculus, but they are hardly "based" on it. His challenge to prove the connection, though stated in polemical language, could inspire a landmark BPM paper.

Petri Nets

The Petri network (or Petri net), a notion devised in 1962 by the mathematician Carl Adam Petri, is a formal graphical process modeling language that can be used to design systems as diverse as train track switches and business processes. With respect to the latter, Petri nets help describe, and indeed can be used to implement, the semantics of process control flow, ranging in complexity from basic branch and join rules to complicated synchronization techniques such as "dead-path elimination," a core topic in WSFL and BPEL.

Petri Nets in a Nutshell

The graphical building pieces of a Petri net diagram are the following:

- ***Place.*** Drawn as a circle, a place is a stopping point in a process, representing, in many cases, the attainment of a milestone.
- ***Transition.*** A transition is a rectangle that represents an event or action.
- ***Token.*** A token is a black dot residing in a place. During the execution of the process, tokens move from place to place.
- ***Arc.*** An arc is a link from a transition to a place or from a place to a transition.

The Java applet design editor and simulator shown in Figure 4 is a typical Petri net

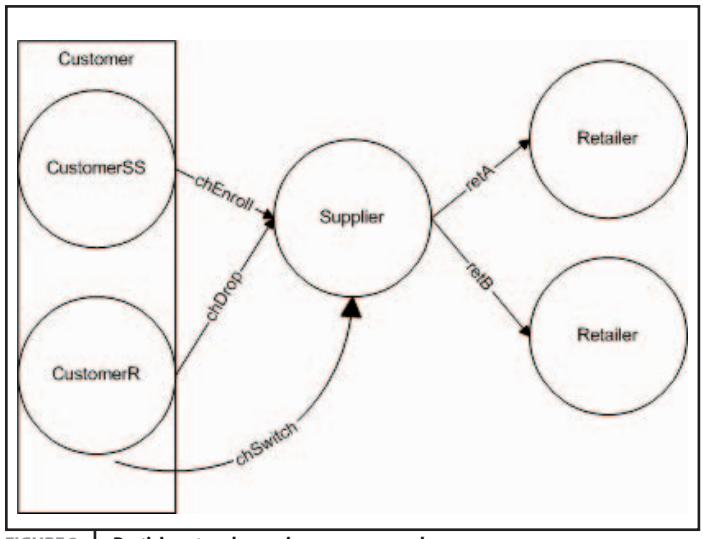


FIGURE 3 | **Participant exchange in energy example**

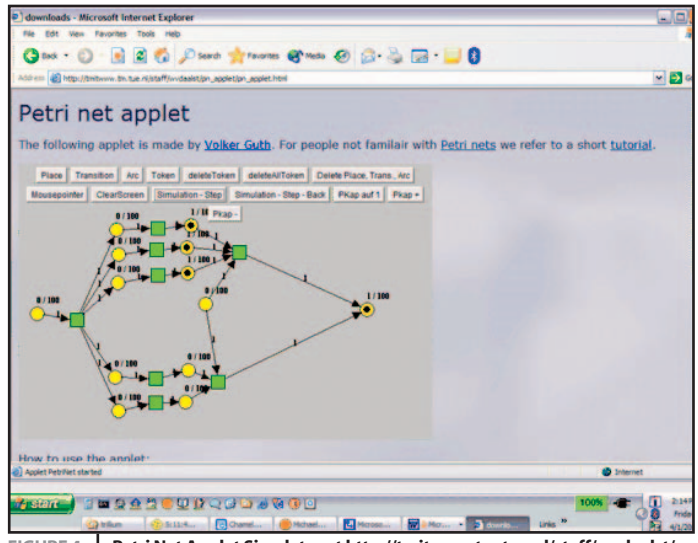


FIGURE 4 | **Petri Net Applet Simulator at http://tmitwww.tm.tue.nl/staff/wvdaalst/pn_applet/pn_applet.html**

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

modeling tool, which promotes a design-and-simulate approach to building a Petri net. Design involves dragging places and transitions onto a canvas, drawing arcs between them, and dropping an initial placement of tokens into a subset of the places. Simulation is a step-by-step run of the Petri net, where each step involves firing a transition and watching the tokens advance to new places.

Some elementary rules of Petri net design are:

- An arc can connect a place to a transition, or a transition to a place, but never a place to a place, or a transition to a transition. Thus, the order of places and transitions alternates on a given path: place→transition→place→transition→place, and so on.
- A path should begin and end with a place.
- Place and transition are many-to-many. A place can branch into multiple transitions. Multiple places can join into a single transition. A transition can branch into multiple places. Multiple transitions can converge into a single place.
- Tokens belong in places. A transition cannot contain a token.

Rules of execution include the following:

- A transition is enabled (i.e., capable of firing) if each input place (i.e., each place that links to it) has at least one token.
- When a transition fires, it consumes a token from each input place and generates a token for each output place (i.e., each place that the place links to).

Figure 5 shows the Petri net for a process in which magazine editors vote on whether to accept an article for publication. When the article arrives, it is distributed for review and for voting by a number of editors. The voting completes when either two editors accept it or one rejects it; in the former case, the author is sent an acceptance notice, in the latter a rejection.

The Petri net starts at the *New* place. When the *Send For Vote* transition fires, the net branches into three paths: two for acceptance, one for rejection. In the acceptance paths, the place *Accepting* links into the transition *Accept*, whose output place is *Accepted*; in effect, when *Accept* fires, a token is passed from *Accepting* to *Accepted*. The rejection path is similar: the transition *Reject* moves a token from the place *Rejecting* to the place *Rejected*. The two acceptance paths converge in the transition *Send Accept Notice*. Similarly, in the rejection path, the place *Rejected* links to the transition *Send Reject Notice*. Each of the "send notice" transitions links to the output place *Done*, the final step in the process. The curiously named place *Mutex*, an input to "send notice" transitions, is discussed below.

The behavior of the voting process is best understood by following the path of tokens through the net. Step (a) shows the initial placement: one token in *New* and one in *Mutex*. The shading of the transition *Send For Vote* indicates that it is enabled; it is enabled because its one input place, *New*, has a token. When *Send For Vote* fires, the token is removed from *New* and tokens are generated in the *Rejecting* and *Accepting* places, thus enabling the transitions *Accept* and *Reject*, as shown in step (b). Step (c) depicts the state of the net after one acceptance and one rejection. The token from the uppermost *Accepting* has now moved to *Accepted*; likewise *Rejecting* has now transitioned to *Rejected*. Of the two "send notice" transitions, only *Send Reject Notice* is enabled, because both of its input places – *Rejected* and *Mutex* – have a token. When *Send Reject Notice* fires in step (d), the tokens from *Rejected* and *Mutex* disappear and a token is moved into *Done*.

Exemplifying the powerful synchronization capabilities of Petri nets is the *Mutex* ("mutual exclusion") place, which ensures that only one notice – an acceptance or a rejection – is sent to the author. *Mutex* is an input place to both the *Send Accept Notice* and *Send Reject Notice* transitions, and hence neither of these transitions can fire unless *Mutex* has a token. However in the initial token placement, *Mutex* has exactly one token. If two acceptances and one rejection occur at roughly the same time, both *Send Accept Notice* and *Send Reject Notice* will contend for that token. But when one

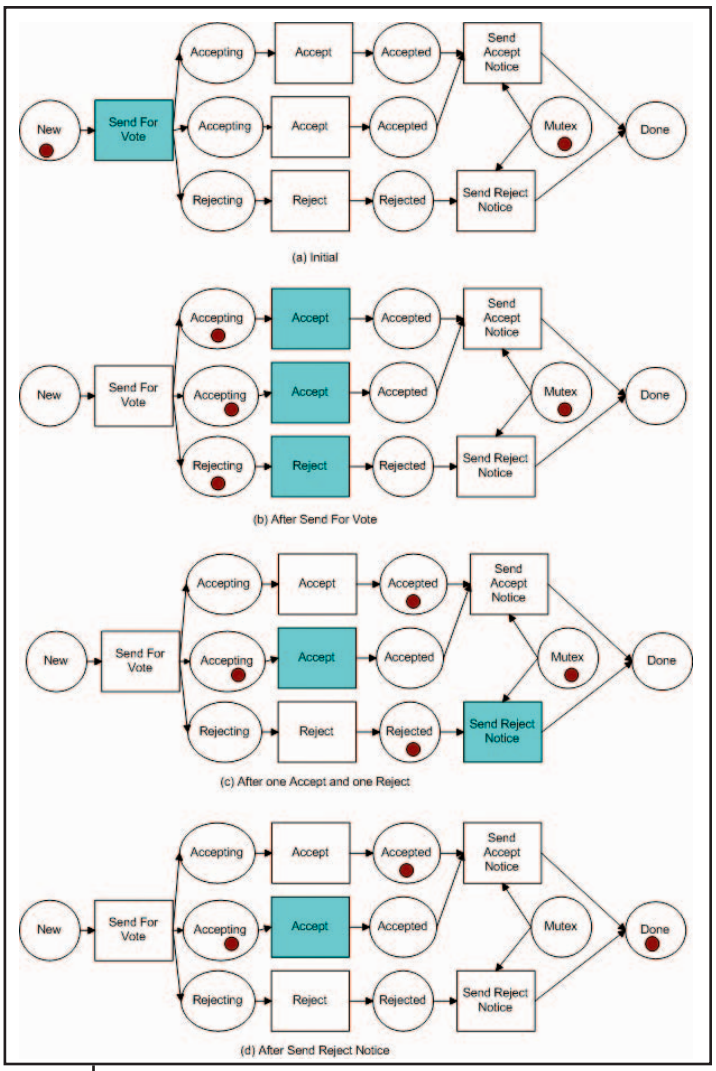
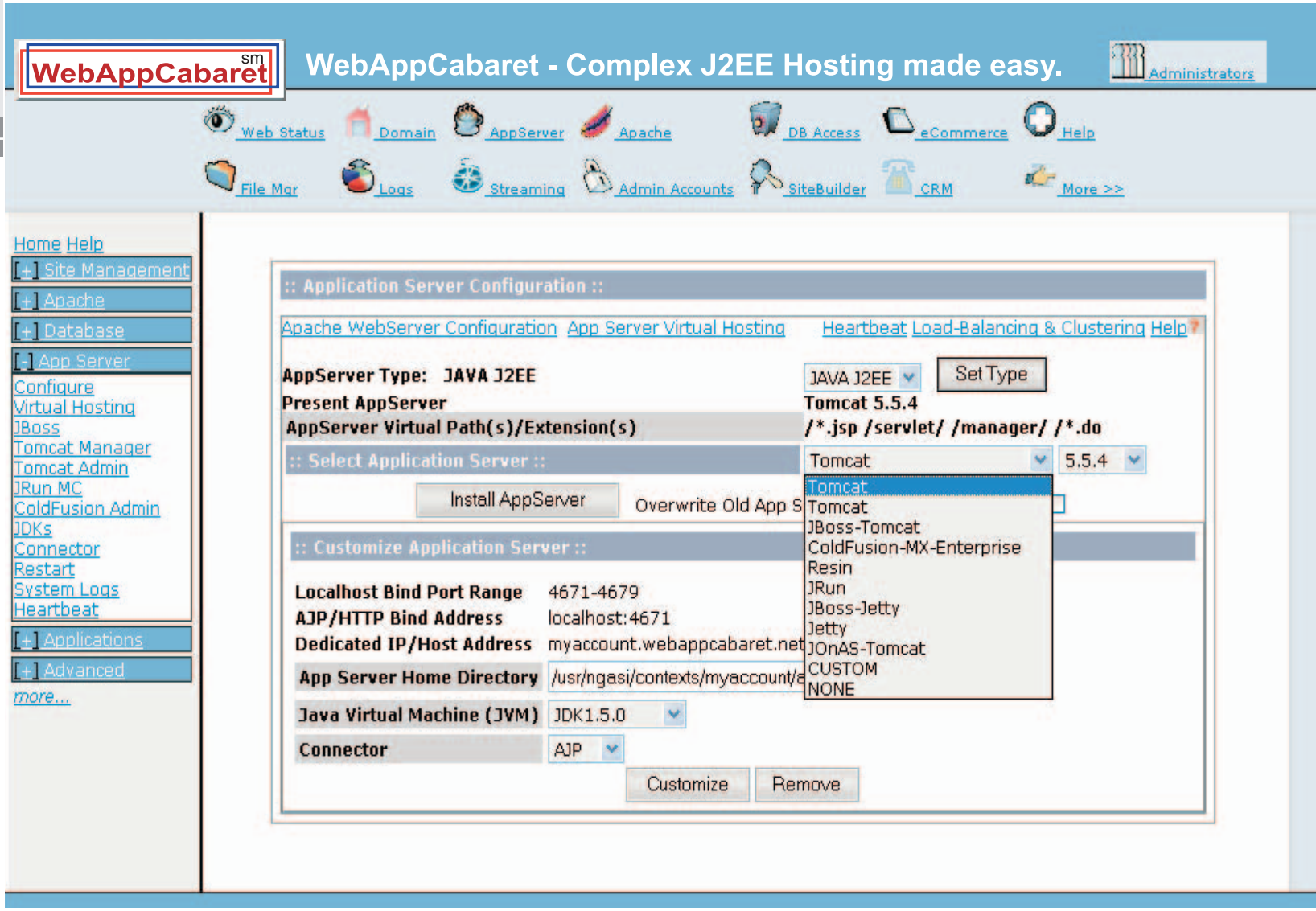


FIGURE 5 **Petri net example: editor vote**



Imagine a hosting company dedicated to meet the requirements for complex web sites and applications such as those developed with Java J2EE.
At WebAppCabaret our standards based process and tools make deploying Java J2EE applications as easy as a point-and-click.
We call it **Point-and-Deploy Hosting**.
Our advanced NGASI Web Hosting management Control was designed for the hosting and management of complex web sites and applications thus cutting down on maintenance time.

Backed by an experienced staff as well as a **Tier 1 Data center** and network. Our network is certified with frequent security audits by reputable security firms.

All hosting plans come with **advanced tools** to manage your application server and Apache web server. Not to mention the other features, such as virus-protected email, bug tracking, and many more portal components.

Complete power and control at the tip of your fingers. We take care of the system and hosting infrastructure so you can concentrate on development and deployment of your application. That is **real ROI**.

Log on now at http://www.webappcabaret.com/jdj.jsp or call today at 1.866.256.7973

NGASI POWERED

WebAppCabaret sm

http://www.webappcabaret.com/jdj.jsp **1.866.256.7973**

Point-and-Deploy J2EE Hosting

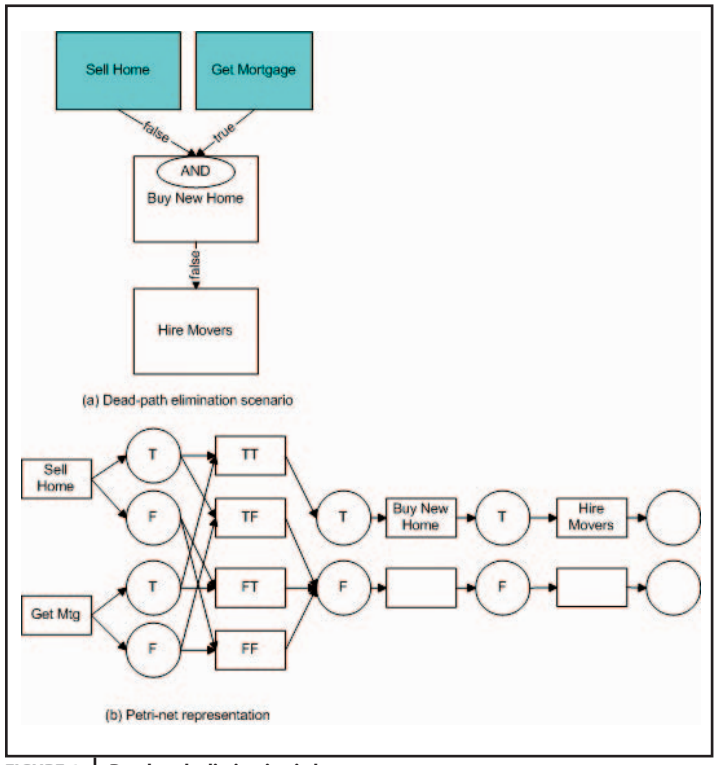


FIGURE 6 **Dead-path elimination in home move**

transition fires, it will consume the token, immediately disabling the other transition, thereby preventing it from firing.

Petri Nets and Dead-Path Elimination

A mountain of academic papers has been written about the use of Petri nets to model process control flow. "Fundamentals of Control Flow in Workflows" (see References below), for example, develops precise Petri net implementations of common process patterns such as AND and XOR splits and joins, as well the thornier "dead-path elimination," considered in this section.

Dead-path elimination is a technique used in languages such as BPEL and WSFL to bypass, or "pass through" activities whose preconditions are not met. For example, in part (a) of Figure 6, the activity *Buy New Home* waits for the parallel activities *Sell Home* and *Get Mortgage* to complete, and it executes only if both have a true result. If at least one is false, the execution of *Buy New Home* is skipped (i.e., don't buy a new home without financing and the sale of your old home), as is the execution of the successor activity *Hire Movers* (i.e., don't call movers if you aren't moving).

Part (b) shows the Petri net representation. Two crucial constructs used in this example are:

1. ***Color:*** The ability of a transition to branch to one of several places depending on the value of the token. In our case, the ability to follow different paths depending on whether "Sell Home" and "Get Mortgage" succeed or fail.
2. ***Lambda Transition:*** A transition that is strictly internal, used merely to drive control flow. A lambda transition fires immediately once it is enabled. In our case, lambda transitions are used to pass through the "dead path."

In the net, both the *Sell Home* and *Get Mortgage* transitions, representing the respective activities, branch upon completion in one of two paths: a true result leading to place *T*, and a false result to place *F*. A set of four lambda transitions then waits for and aggregates the results of the two activities: *TT* is enabled when both activities return true, *FF* when both are false, *TF* when *Sell Home* is true and *Get Mortgage* false, and *FT* is the converse of *TF*. Each of these transitions fires as soon as it is enabled. The *TT* transition passes a token to the place *T*, thereby enabling the *Buy Home* transition, and, further downstream, *Hire Movers*. *TF*, *FT*, and *FF* each link into the place *F*, which, in turn, leads into lambda transitions representing the no-op versions of *Buy Home* and *Hire Movers*, which execute instantaneously and do nothing. This Petri net, simply stated, functions as a synchronizing truth table, allowing the purchase of the home and subsequent activities only when it receives "trues" for all inputs, and skipping past that logic otherwise.

Petri Nets and BPM Standards

Of the major BPM standards, three – BPEL, WSFL, and BPMN – have roots in Petri nets. Each standard uses the notion of token passing to describe the semantics of control flow. The BPMN specification uses the token concept throughout. In WSFL and BPEL, the most striking Petri net–inspired discussion is "dead-path elimination."

Summary

For the BPM practitioner, theory, in the form of the Pi Calculus and the Petri net, actually matters! Why? First, because theory is mentioned so frequently in practical discussion of BPM, it is unavoidable: practitioners need to understand what the hype is all about. Second, theory injects much-needed rigor into a relatively immature BPM: Pi Calculus formalizes system interactions; Petri nets demonstrate precise control flow.

References

- R. Milner. "The Polyadic Pi-Calculus: A Tutorial" in F.L. Bauer, W. Brauer, H. Schwichtenberg, editors, *Logic and Algebra of Specification*. Springer. 1993.
- W.M.P. van der Aalst, "Pi Calculus versus Petri Nets: Let us Eat 'Humble Pie' Rather Than Further Inflate the 'Pi Hype'" Eindhoven University, Department of Technology Management.
- B. Kiepuszewski, A. H. M. ter Hofstede, W. M. P. van der Aalst. "Fundamentals of Control Flow in Workflows" Acta Informatica, 39(3):143-209, 2003.
- N. Kavantzas. "Aggregating Web Services: Choreography and WS-CDL" (presentation), Oracle, April 2004.
- *Workflow Patterns Website:* www.workflowpatterns.com
- *Petri Net Design Applet:* http://is.tm.tue.nl/staff/wvdaalst/pn_applet/pn_applet.html

About the Author

Michael Havey is an IBM consultant with ten years of industry experience, mostly with application integration. Michael is currently writing a book on BPM, to be published by O'Reilly later this year.

haveym@ca.ibm.com

Global Knowledge™
Experts Teaching Experts

Take our 10-minute assessment.

It's no secret that Red Hat Enterprise Linux is the most widely deployed Enterprise Linux solution in the world today. In fact, Certification Magazine recently named the Red Hat Certified Engineer (RHCE) program the highest quality certification in all of IT. Global Knowledge, a premier partner for Red Hat training, invites you to test your skills and pre-assess for Linux success! **Visit globalknowledge.com/lw for details.**

redhat PREMIER PARTNER

Oracle Web Services Manager

A platform to manage, secure, and extend enterprise Web Services

As departments and organizations in the enterprise adopt Web services, the importance of managing and securing, and in some cases, extending these services to partners, grows. Using a centralized platform to do it reduces the effort and cost involved, while upping security. Web Services Manager from Oracle is designed to meet the challenge of stitching together services in large enterprises into a single management system.

WRITTEN BY
BRIAN BARBASH

The Web services management functionality in WSM is built on the following basic components:

- ***Policies:*** Sets of rules, transformations, and/or constraints that are applied to inbound and outbound service requests.
- ***Gateways:*** These components are the enforcement points for policies and serve as proxies to Web services, isolating them from clients. Gateways can be put in front of any Web service, hosted in any environment, local or remote.
- ***Agents:*** Similar to gateways in enforcing policies; however, agents execute in-process on the server the Web service is hosted on and intercept all incoming and outgoing service traffic. They offer more tightly integrated control over individual operations on a service.
- ***Monitor:*** Collects data from gateways and agents and presents them to administrators in graphical format.

For purposes of this review, I have created a series of Web services for personal data management: a music collection service, a movie collection service, a workout log service, and a ledger for tracking basic expenses. WSM will be used to manage, secure, and monitor these services.

Gateways

As mentioned, gateways in WSM serve as proxies to Web services. Their behavior is similar to the Controller component of a Model-View-Control (MVC) application. A gateway serves as the entry point for service calls and routes traffic to the appropriate back-end Web service based on the content of the request document.

Establishing a gateway is as simple as assigning it a name and a URL. The URL will ultimately serve as the endpoint for all Web services managed by the gateway. Once created, each applicable Web service must be registered. WSM can read service descriptions directly from a WSDL file or from a UDDI registry.

Figure 1 shows the service registration screen for a gateway created to serve as the entry point for the personal data management services already created. In this example, the Workout Service will be imported from a WSDL file. WSM requires a service name, version, the URL of the WSDL document, and the protocol on which the service operates. Once imported, WSM creates a new service URL and WSDL document that point to the gateway, hiding the true endpoint of the service from the calling clients.

Notice that Figure 1 highlights one of the most powerful features of WSM: the ability to transfer between protocols. Out-of-the-box, WSM's gateways can transfer SOAP over HTTP(S) to a JMS or IBM MQ Series messaging system. The system can also be extended to support additional custom protocols. This capability provides the immediate benefit of exposing internal, heterogeneous Web services on a single protocol.

With all services registered in the gateway, WSM must be configured to route incoming service requests appropriately. The system provides an XPath-based content-routing mechanism that supports inspection of either the SOAP Envelope or a SOAP Attachment. (Note: When inspecting an attachment, its content must contain XML and be referenced in the SOAP Envelope according to the SOAP with Attachments specification.)

In the case of the gateway we just created, the content-routing instructions must be established for each of the registered data management Web services. For example, the routing instruction for adding a record in the movie collection service is based on the XPath statement:

ORACLE

Company Info
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
Web: www.oracle.com
Phone: 1.800.ORACLE1

http://www.oracle.com/technology/products/ias/index.html

```
soap:Envelope/soap:Body/wsj:Action =
  AddMovie
```

As shown in the XPath statement example, routing instructions are namespace-aware. WSM automatically presents an input field for each referenced namespace in the statement.

When messages are received by the gateway, content routing is done in the order in which the instructions appear in the console. This order may be adjusted manually to optimize processing efficiency.

Agents

WSM agents are in-process components that are installed directly into the application server and operate as a client or server. Client agents are used when an application acts as a client to other Web services. Server agents are used when an application is exposed as a Web service to calling clients.

Unlike gateways in which service calls are routed based on their content and/or operation, all requests to or from an application on a server with an agent installed are processed by the agent. On Java servlet containers, agents can be deployed as JAX-RPC handlers or as servlet filters. Out of the box, Oracle supports Apache Tomcat with Axis, BEA WebLogic, and Tibco BusinessWorks. The new release will include support for Microsoft IIS as well as IBM WebSphere.

Setting Policies

There's a high degree of flexibility in the policy administration in Oracle WSM. Each policy consists of a set of pipelines that in turn is a collection of individual steps. WSM comes prepackaged with steps for encryption, decryption, authentication, authorization, logging, quality of service monitoring, SAML and XML transformation. Customized steps can also be developed based on the WSM framework.

A policy consists of four separate pipelines:

1. Pre-Request Pipeline
2. Request Pipeline
3. Response Pipeline
4. Post-Response Pipeline

There is no restriction in the steps that can be executed in any of the pipelines. The product documentation recommends that enterprise-wide policy steps be maintained in the pre- and post- Pipelines, while service-specific policy steps be maintained in the request and response pipelines.

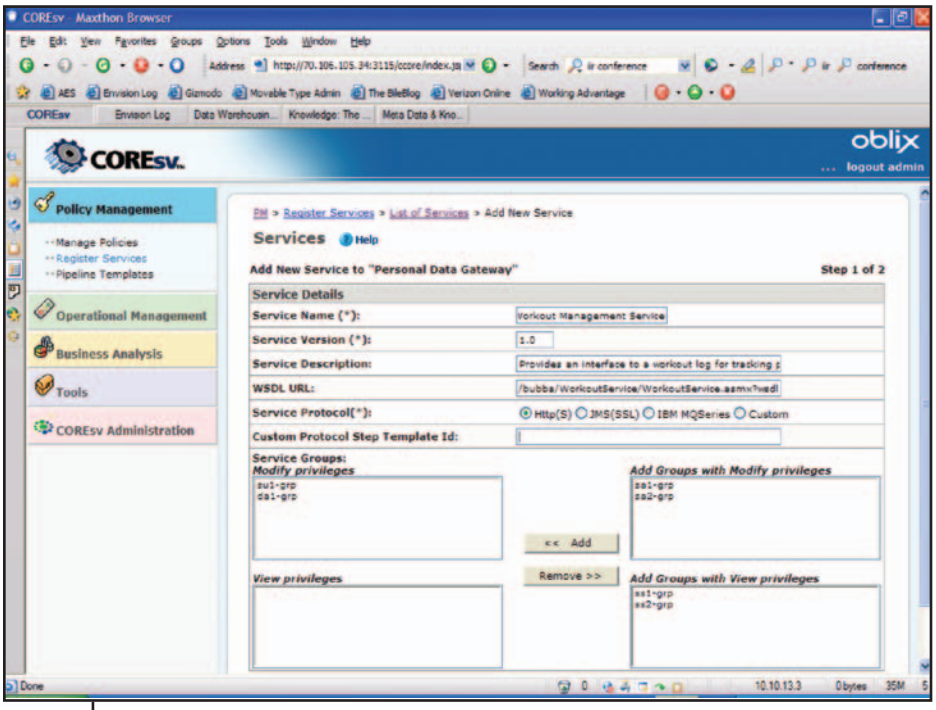


FIGURE 1 | **Service registration**

To illustrate, I've modified the expenses ledger service to use a slightly different document structure. The service itself will support both document structures. Instead of modifying the client for the new structure, however, I will insert steps to transform the incoming and outgoing document via XSL. To configure this in the system, WSM simply requires a valid URL or physical file location for the stylesheet.

Once configured and deployed, the calling client sends the original document to the server, the gateway intercepts it and transforms the document via the newly configured XSL pipeline step, routes the document to the appropriate service, gets a response, transforms the document back to its original form, and returns the result to the calling client.

Monitoring

WSM also provides a solid set of monitoring and analysis tools for managing services. Data is presented to administrators in a dashboard display that can be customized to target specific information. Alerts can be configured to identify specific service conditions. For example, WSM supports latency checks to ensure that managed services are operating according to predefined service levels. This is just one example of its wide variety of management and monitoring capabilities.

Summary

Managing, securing, and extending individual Web services applications in an enterprise is a daunting challenge that grows with the increased adoption of the technology. Oracle's WSM helps meet that challenge by providing a strong platform for defining and enforcing security policies. Its monitoring capabilities help organizations meet service levels and identify potential bottlenecks. Additionally its support for multiple platforms makes it applicable to many different environments. Overall, Oracle's WSM delivers on Web services management and administration and is a good option for enterprises.

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

Coordination and Transactions Are Key to Building an Operational SOA

Secrets of SOA design

Building an SOA usually means leveraging a loosely coupled-type architecture. While the benefits of a loosely coupled SOA with many services are apparent, the operational characteristics can be a nightmare. However, with a bit of planning, and the use of some standards, your SOA will be both reliable and functional.

WRITTEN BY
DAVID LINTHICUM

The key problem is to make many services, some you own and some you don't own, work and play well together. The objective is to leverage many services, and do it in a way that makes them appear like a single application, although the services could be running anywhere in and outside your organization. In short, making many appear as one.

Okay, now that we know what the problem is, how do we solve it? First, you need to remember that this is largely an architectural problem, and not something you can toss products and standards at. Indeed, you need to define the capabilities of your SOA based on the requirements of the domain, and back the appropriate solution set (standards, tools, technologies) into it. For instance, too many fine grained services and all of the technology and standards in the world won't help you.

What's key here, at least from an architectural perspective, is that you make the right calls in terms of what services are going to make up your SOA in support of critical business events. This is where most SOAs go off track, typically attempting to bind too many services, and at the wrong granularity. You can consider services that make up your SOA like links in a chain, each defining the reliability and performance.

Having stated that obvious problem, there are techniques, technologies, and standards that you can apply that provide your SOA with better capabilities to manage a loosely coupled and distributed SOA. Although they're just getting off the ground, I believe they're some of the most important emerging capabilities.

Look at Standards

Of course any weaknesses in Web services and SOAs always have a WS* standard associated with them, and this problem is no exception. For our purposes, the most relevant are: WS-Coordination, WS-AtomicTransaction, and WS-BusinessActivity. Note that the last two are dependent on the first, more on that below.

The *WS-Coordination* specification describes a framework for providing protocols that coordinate the actions of distributed services. Such coordination protocols are used to support a number of applications, including those that need to reach consistent agreement on the outcome; in other words, the ability to make many services function as a single service, and to do so through close coordination using standard communication and state mechanisms.

The WS-Coordination framework enables an application service to create a context needed to propagate an activity to other services. This is done through the registration of coordination protocols. The framework enables existing transaction processing, workflow, and other systems for coordination to hide their proprietary protocols and to operate in a heterogeneous environment. This specification describes a definition of the structure of context and the requirements for propagating context between cooperating services.

The problem WS-Coordination is looking to solve is that Web services increasingly tie together a large number of participants, forming large distributed computational units that are known as activities. In many SOAs, the resulting activities often leverage a complex structure, with complex relationships between their participants. Therefore, the execution of such activities often takes a long time to complete due to business latencies and user interactions. The use of the coordination framework isn't restricted to transaction processing systems; a wide variety of protocols can be defined for distributed applications.

> "You should end up with an IT infrastructure that's both adaptable and inexpensive through the notion of reuse"

The WS-Coordination specification describes a framework for a coordination service (or coordinator) that consists of these component services: an Activation service, a Registration service, and a coordination type. An *Activation service* enables an application to create a coordination instance or context. A *Registration service* enables an application to register for coordination protocols. A *coordination type* specifies a set of coordination protocols to use in a distributed application instance.

The *WS-AtomicTransaction* specification defines an atomic transaction coordination type for use with WS-Coordination. It defines three specific agreement coordination protocols for the atomic transaction coordination type: completion, a volatile two-phase commit, and a durable two-phase commit. Those building an SOA can use any or all of these protocols when building solutions that require consistent agreement on the outcome of short-lived distributed activities that either work or don't work – all or nothing.

The *WS-BusinessActivity* specification defines the business activity coordination type that leverages the WS-Coordination specification. The specification defines two specific agreement coordination protocols for the business activity coordination type: BusinessAgreementWithParticipantCompletion and BusinessAgreementWithCoordinatorCompletion. Developers can use any or all of these protocols when building applications that require agreement on the outcome of long-running distributed activities.

The essence of this problem, and the solutions, is the ability to make many different behaviors existing on all kinds of environments and at all locations appear to be a single, well structured, and reliable application. This is a problem we've been wrestling with for years, and have put specific types of technologies on it, such as transactional systems and state machines. There is really nothing new here, just new tools and standards.

With the advent of Web services we have both an opportunity and a problem. The opportunity is to build applications with reusable services that we may or may not have built ourselves. Thus you should end up with an IT infrastructure that's both adaptable and inexpensive through the notion of reuse. The management of distributed services carries its own set of issues, as discussed above. However, with a bit of good architectural forethought and the good use of standards and technologies, I think we can have our cake and eat it too.

About the Author

David Linthicum is the CTO of Grand Central Communications (www.grandcentral.com) and a leading expert in application integration and open standards. He has held key technology management roles with a number of organizations including Mercator and SAGA Software. David has written or co-authored 10 books, including the groundbreaking best-seller *Enterprise Application Integration* released in 1998. His latest book is *Next Generation Application Integration, From Simple Information to Web Services.*

dlinthicum@grandcentral.com

BizTalk Server 2004: Too Hot to Handle?

Where to successfully apply BizTalk Server technology

Recent trends in IT such as service-oriented architecture (SOA) and Web sevices, in conjunction with the still-increasing popularity of the .NET framework, put Microsoft's BizTalk Server in the center of attention for CIOs, CTOs, architects, and enterprise developers. Apparently everyone who is involved with the Microsoft platform now wants to implement BizTalk. Frequently we see that BizTalk is not implemented in areas where the benefits of the technology can be maximized. This leads to disappointment about the value of BizTalk, even though the problem is in the application of the technology instead of in the technology itself. In this article I will point out where, how, and when the third generation of BizTalk Server can provide real business value.

BizTalk Server Is Hot

In March 2005 BizTalk Server (BTS) reached an implementation base of 4,000 organizations. Microsoft states that these 4,000 implementations in four years make BTS the fastest-growing integration server on the market. Ever since it was first released, BTS has held a firm spot in the upper right quadrant of Gartner's magic quadrants for application integration and orchestration tools.

WRITTEN BY
LOEK BAKKER

It's no wonder BizTalk is the center of attention for anyone who has a Microsoft infrastructure or who develops, designs, or architects Microsoft-based solutions. However to fully grasp the benefits of BizTalk Server, we should know where to successfully apply BizTalk Server technology.

A Brief History of BizTalk Server

To fully understand what BizTalk is and what it can do for you and your company, we should delve into the history of the product.

The first version of BizTalk, BTS 2000, was the result of two separate server products on which Microsoft development teams were working. On one side was the development of the messaging infrastructure and the BizTalk Framework (a.k.a. BTF, a SOAP 1.1 extension and XML framework aimed at integrating e-commerce applications), and on the other side there was the development of an extension to COM+ called COM+ Scheduler, which we now know as the Orchestration Engine. Nine months before BTS 2000's shipment in December 2000, Microsoft decided to merge both development teams in order to create one single solution for application integration and XML messaging.

Fewer than two years later Microsoft released the next version of BTS, named BTS 2002. This version contained a few improvements over the BTS 2000 version, and was generally more for administrators than for developers and end users. Additionally BTS 2002 offered some new functionality such as an HTTP receive function and increased Web service support.

Then, in 2004 (as the name suggests), BizTalk Server 2004 (formerly code-named Jupiter, which Steve Ballmer affectionately termed "BizTalk on steroids") was released. BTS 2004

is the first version that is fully based on the .NET framework, and which is written in fully managed code. The differences between it and the 2002 version are tremendous. Any BTS 2000 developer could learn to operate the BTS 2002 version within a day, but the BTS 2004 version is a different story. Development tools are now integrated in Visual Studio.NET, thereby perhaps adhering less strongly to the principle "code less, configure more" that was used throughout the BTS 2000 and BTS 2002 products.

In today's version, we can still see the two pillars of the product: messaging and orchestration. However a big difference with the BTS 2000 and 2002 versions is that by default, orchestration is used within the messaging tasks, even if the messaging task is straightforward and does not need the advanced options of orchestration. There is a way to work around this, but it is not as straightforward as using an orchestration for this task.

> "BizTalk Server 2004 has gained its fair share of attention lately, and it has become the fastest-growing integration server on the market"

The Road to Choosing BizTalk Server

The key to assessing whether implementing a BizTalk-based solution will help your company is to look at it from an architectural perspective. At Capgemini we have developed the Integrated Architecture Framework (IAF) that defines architecture by means of contextual, conceptual, logical, and physical models, which progressively analyze an architectural need and refine a solution.

The architectural levels of the IAF are:

- ***Contextual:*** answers the "where" question. Delivers contextual information and key principles that define the architecture. Focuses on the company and its environment.
- ***Conceptual:*** answers the "what" question. Defines the architecture on a high abstraction level. Focuses on which business problem the solution/architecture should solve.
- ***Logical:*** answers the "how" question. Defines what services are needed to implement the desired architecture. Focuses on the solution for the problem stated at the conceptual level.
- ***Physical:*** answers the "with what" question. Defines the products and technologies with which the solution can be realized.

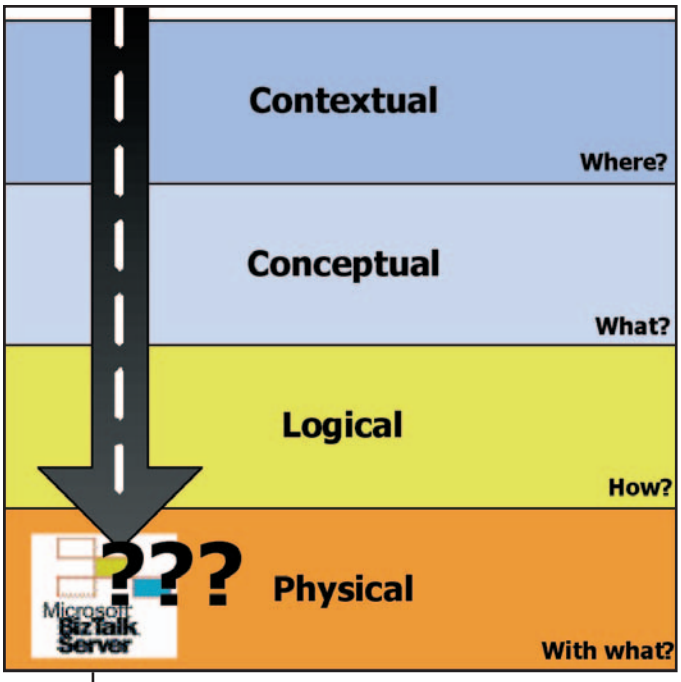


FIGURE 1 | The road to choosing BizTalk Server

Starting from the contextual level, we should analyze all levels of the architecture, and see where and how BizTalk fits into the picture. This should eventually lead to a well-founded choice for the BizTalk platform, or for an alternative.

What we often see is companies deciding to implement BizTalk regardless of whether analysis on the upper levels justifies this implementation. The right order to choose *any* technology at the physical level is to progressively analyze all levels, beginning with the contextual level. We must first understand the problem before we supply a solution. Now that we have our analytical framework, let's see how we can apply it to BizTalk.

Where

The .NET framework has come a long way since it was first introduced about five years ago. On the eve of the launch of the .NET framework version 2.0, even the open source community has embraced the (unofficial) .NET framework, called Mono. Even though .NET is now available for Linux and Unix, BizTalk is still seen as a product exclusively for the Microsoft/Intel platform. To a certain extent this makes sense, as BizTalk Server 2004 runs only on Windows Server, and relies heavily on Microsoft SQL Server for its internal databases.
BizTalk offers great opportunities to connect to other platforms, particularly through the broad application of open standards such as BPEL, SOAP, WSDL, XSD, and HTTP to name a few. Also, there are many adapters available to connect to other platforms. The use of open standards and the broad availability of adapters make BizTalk a great player in heterogeneous, mixed Windows-Unix-Linux environments. However the question remains as to whether it is wise to implement BizTalk as the only Intel-based system in a Unix or Linux environment.

If the *where* question tends to exclude BizTalk as a serious option, then you must have very strong arguments on the *what* and *how* sections to ultimately justify the choice for BizTalk.

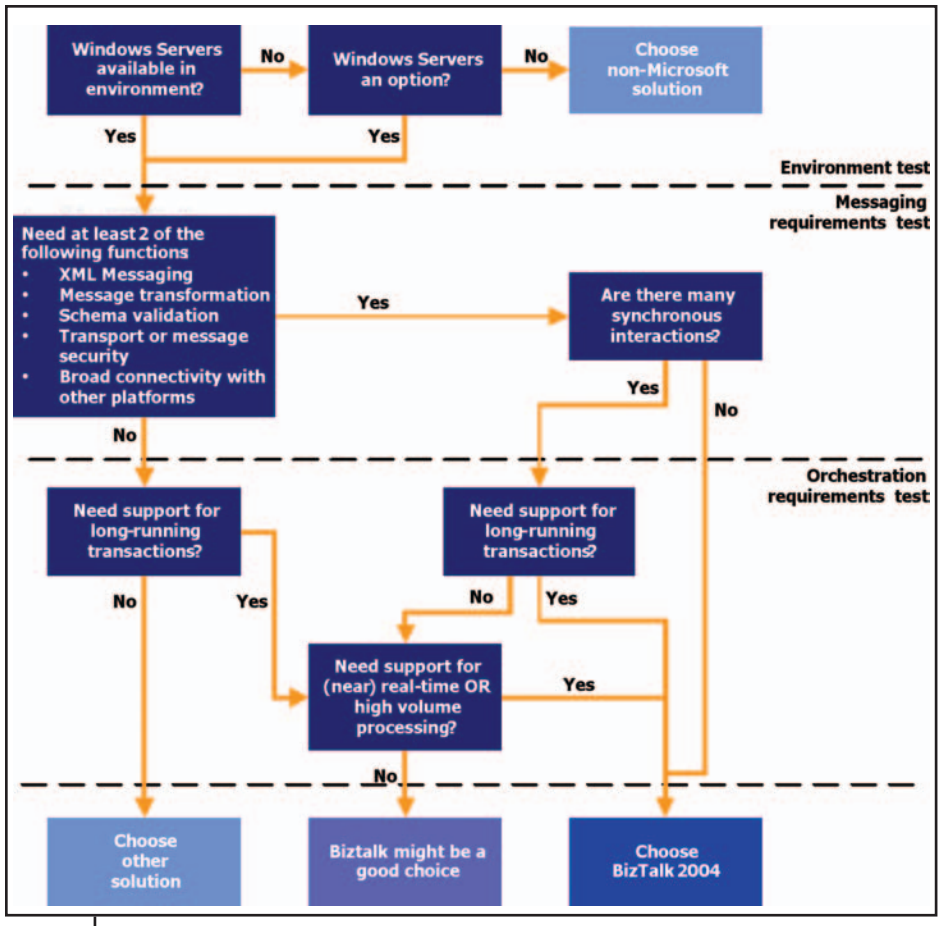


FIGURE 2 | **Decision tree for BizTalk 2004**

What

The *what* question focuses on the business problem that needs to be solved. It states the requirements for the solution. Typical business problems in these areas may be:

- "We need to shorten the time to market for new applications or automated support for business processes"
- "We want to automate the processing of claims"
- "We need to be able to exchange data with our trading partner"
- "We want to reduce operational costs"
- "We want to offer a new service to our customers"

Also, business problems may already be stated in the form of an envisioned solution:

- "We need to implement an SOA to provide business agility"
- "We must use EAI tools to connect our Billing application to our Order application to streamline our procurement process"

Note that business problems stated this way already hold an answer to the *how* question; we will describe this in the next section which discusses the logical level.

Since we are assessing BizTalk here, we should see if the problem stated matches up with the problems BizTalk aims to solve. BizTalk is a platform for:

- Application integration
- B2B messaging
- Business process automation

At the end of analyzing the conceptual level, we should have further narrowed the area into which BizTalk should fit. If the stated business problems are no where near the kind of problems stated here, then BizTalk might not be the right way to go. The conceptual phase sets the boundaries and constraints for the solution, which is outlined by answering the *how* question. Business value is determined at the conceptual level.

How

At the logical level, we often encounter popular, possibly hyped concepts such as SOA, Enterprise Service Bus (ESB), Business Process Management (BPM), and Enterprise Application Integration (EAI). BizTalk Server often gets associated with these concepts; therefore, it is tempting to already make the decision to choose BizTalk Server at this stage. This is the major pitfall for most failed BizTalk projects and implementations. To successfully solve a problem, you must understand the problem. To provide a solution, you must know *how* to provide this solution. This is why some implementations fail to meet the expectations from the business people. Thorough analysis of the solution is crucial, since BizTalk is very good at some tasks, average at others, and not suitable for some functions. It is still *too early* to choose BizTalk at this stage.

Key design decisions are made at the logical level. Some of the design decisions and constraints that are relevant to application integration, B2B messaging, and Business Process Automation solutions are:

- Are we going to need [XML] messaging?
- How are services or components going to interact? By method calls? By messages?
- Are there any long-running transactions?
- Is there end-user interaction?
- Will users perform tasks that require online, real-time processing?
- Is there any workflow involved?
- Are we working with sensitive information?

At this stage, you should get as much detail as possible on the solution design. The result of this analysis is the input for the next stage, where we decide whether or not to implement BizTalk.

With What

Eventually analysis of all other levels leads to the question: With what are we going to realize this solution? At the physical level, we should have a look at the functionality BizTalk has to offer and see how it fits into the architecture, requirements, and the solution we have analyzed so far. To provide a complete

view, we should also assess available alternatives.

Arguably the best way to start describing BizTalk's capabilities is to point out what BizTalk Server is *not*:

- ***BizTalk is not an application server.*** In particular, functionality such as support for atomic transactions is hard, if not impossible to deliver with BizTalk.
- ***BizTalk is not Human Workflow.*** BizTalk's key focus is on what is called technical workflow, not on human interaction. Human Workflow is present in the 2004 version, but at a level that will not be sufficient for any sophisticated workflow system. Usually we see that a BizTalk add-on like K2.NET is used for the Human Workflow part in BizTalk implementations. (It is uncertain whether the Human Workflow Services (HWS) will still be present in the 2006 version of the product, since a lot of aspects of Human Workflow will be implemented in the Longhorn operating system.)
- ***BizTalk is not an application framework:*** BizTalk just gives you the tools to implement an application framework like a Service Oriented Development Architecture (SODA), or an Enterprise Service Bus (ESB) for instance. But it is not an application framework as such.
- ***BizTalk is not an ideal option for synchronous interaction***. By nature BizTalk is a messaging engine that uses queues to process messages, and is optimized for asynchronous interaction. Synchronous processing through BizTalk is similar to chatting with e-mail; it is possible, but there are better ways to provide the same function.

Some of the core capabilities of BizTalk Server 2004 are:

- XML messaging through different open Internet standards
- Connectivity through third-party adapters
- Message transformation/mapping
- Message validation
- Business process orchestration
- Business rules repository and engine
- Secure and reliable messaging
- Long-running transactions
- Human Workflow Services for simple workflow tasks

This makes a BizTalk particularly good pick for application integration, B2B integration, and business process automation as we have seen.

At this point in the architecture study we are in a situation where we know the environment, we understand the business problem, and we know how we are going to realize the solution to this problem. Also, we know what BizTalk is good at and what it is not so good at. The decision of whether or not to implement BizTalk can be reached by using the decision tree in Figure 2.

The decision tree is primarily a quick help in the selection process for the right tool for your integration or process automation needs. It should not be considered a deterministic model to decide whether or not to use BizTalk. There is more to making the decision to implement BizTalk than merely looking at this decision tree, as we have seen. BizTalk Server should fit into the existing architecture, and a good cost-benefit analysis is the key to deciding the full business value of the solution.

Summary

BizTalk Server 2004 has gained its fair share of attention lately, and it has become the fastest-growing integration server on the market. This is not surprising since it offers rich functionality for integration and orchestration tasks. However, even if you already have a .NET environment, under certain circumstances there are other alternatives and options to consider. In this article I have presented an analytical framework to assess whether BizTalk Server 2004 is the right product for you, or whether you should consider alternatives. By applying this framework, and thus by bringing some common sense in selecting the right solution for a posed problem, we can prevent BizTalk from becoming too hot to handle.

About the Author

Loek Bakker is a senior consultant at Capgemini, the Netherlands. He specializes in architecture, SOA ,and Microsoft. NET. Within Capgemini he is a lead architect for BizTalk-based integration solutions.

loek.bakker@capgemini.com

Effective Business Process Management (BPM) Software Puts Business People in Control

Placing process design, execution, and management into the hands of the people who use it

Finally, executives on the business side of management teams are asking for technology that can be delivered, modified, and executed by business people. The promises of many enterprise application investments were never realized in large part due to the fact that the business side was disengaged from their IT investments, leading to large disconnects between expected capabilities and actual capabilities.

WRITTEN BY
JEFF MILLS

Conventional wisdom holds that you can't implement business processes involving human worker interaction and integrations with enterprise applications without a heavy reliance on IT resources. Throwing requirements "over the wall" to IT is where the breakdown occurs. How many CRM implementations have jury-rigged "workarounds" because necessary components proved to be "out of scope" (an issue too often discovered post roll out)? How many ERP solutions promise a Workflow capability that will solve all of your process needs only to find out that there are more employees "touching" processes that impact ERP performance than there are employees who will ever have that application on their desktop? Having said that, you will not be able to prevent all technology shortcomings with the business side fully engaged during the implementation.

Businesses are fluid. Processes change. Policies and procedures are revised. People and departments are restructured. Imagine if the people handing BPM requirements to IT could instead deploy their own requirements and manage their own updates.

An effective BPM technology empowers business people to design and execute a process that is flexible enough to allow the same business user to make changes to that process routinely. Many process management technologies allow business people to design a process, but few actually allow the business user to *own* the process by giving them the keys to the car. Business people possess the knowledge about how processes should flow and what business rules should be applied. They also know what changes need to be made over time. Shouldn't BPM technology, therefore, be designed to match the typical skills business people possess rather than depending on skills sets inherent to IT? Traditional process management technologies require business people to communicate their process needs to IT for them to manage, which leads to delays and "lost in translation" problems. Effective BPM tools put the power of design and implementation into the hands of business people.

Fundamental to a BPM technology is coming "out of the box" with a capability to manage human activities, such as approval management, document collaboration, and a variety of other manual tasks. Also fundamental to an effective BPM tool is giving business people the ability to task and manage human worker activity. Application integration and middleware technologies were created to automate 100% of a process and cut all manual activity out of the mix. BPM technologies were borne because you can't automate everything, and the human worker will always have a place even in the most optimized process. This has left application integration and middleware technologies scrambling to claim a human worker capability, lest they be marked as incomplete or outdated technology. Today's EAI technologies can task the human worker, but they fall well short of being effective because they have an IT dependency when developing and modifying each instance of manual activity.

An "out of the box" human worker interaction capability means granting business people the ability to task manual execution (by person, by role, by geography, by worker availability, etc.)

and the ability to interact with employees through common desktop tools (Word, Excel, Outlook, Acrobat, etc.). Most process management technologies require significant development efforts from IT to incorporate human worker interaction capabilities (new or updates). Effective BPM tools come pre-built to allow business people to design processes with the human worker in mind.

Effective BPM technologies must allow business people to create processes that go outside of enterprise applications and integrate with disparate systems. CRM, ERP, SFA, Help Desk, and other point applications advertise a BPM or Workflow capability built within their software. These modules offer a very light Workflow capability and are, most notably, limited to providing value within the embedded application only. Enterprise applications typically trigger processes that span across multiple departments and legacy environments. Just as in the ERP example cited above, effective CRM processes are also heavily dependent on back office support from people without CRM on their desktop. Effective BPM tools enable business people to design processes with organization-wide breadth and are not limited to driving value within a single application.

Last, effective BPM technologies are process-centric. Workflow technologies are document-centric and take the approach that you begin with a document and determine where you need the document to flow throughout the organization. This methodology is limited because while some processes contain documents, others include system-to-system data transfers or a human worker to perform a manual activity. Many process management technologies focus on the document and where it should go. Effective BPM tools allow business people to focus on the process itself and give the business user flexibility to define business rules to determine whether a document, a data transfer, or a human worker is appropriate for each step of a process.

> "Imagine if the people handing BPM requirements to IT could instead deploy their own requirements and manage their own updates"

True BPM provides enterprises with a process design and management capability that is put into the hands of the business people. It comes pre-built to allow business people to design, manage, execute, and change enterprise processes that include a combination of automation, document management, and human work.

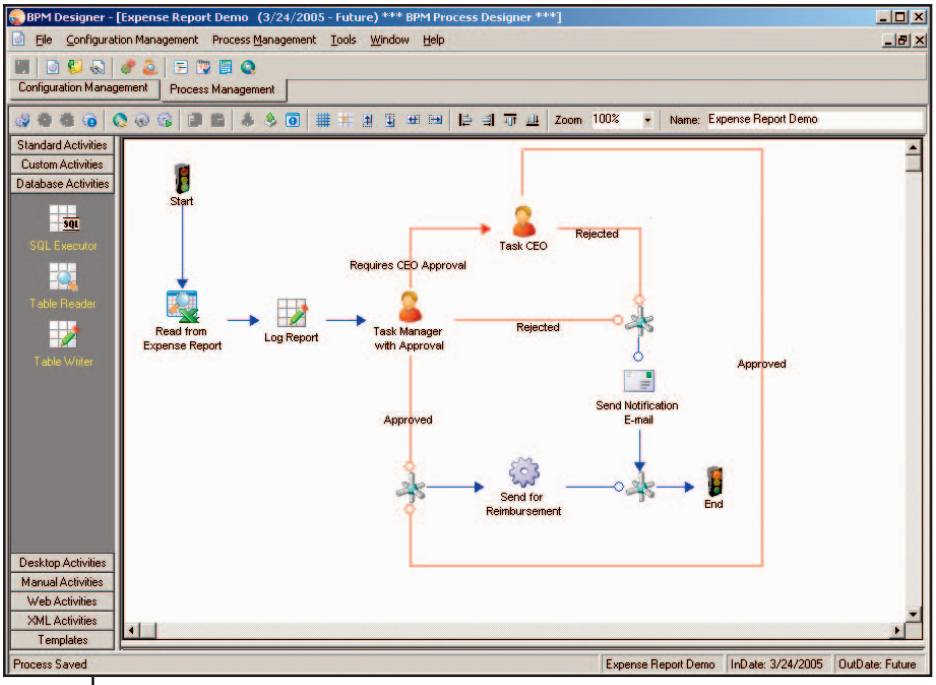


FIGURE 1 | **Easy to use Process Management Consule**

Key Takeaways

Business

- Effective BPM technologies eliminate the handoff of requirements to IT by allowing process owners (business people) to design *and* execute business processes.
- Enabling business people to design and execute their processes means faster deployments, faster responses to inevitable process changes, fewer process management resource requirements, and logically, greater ROI.

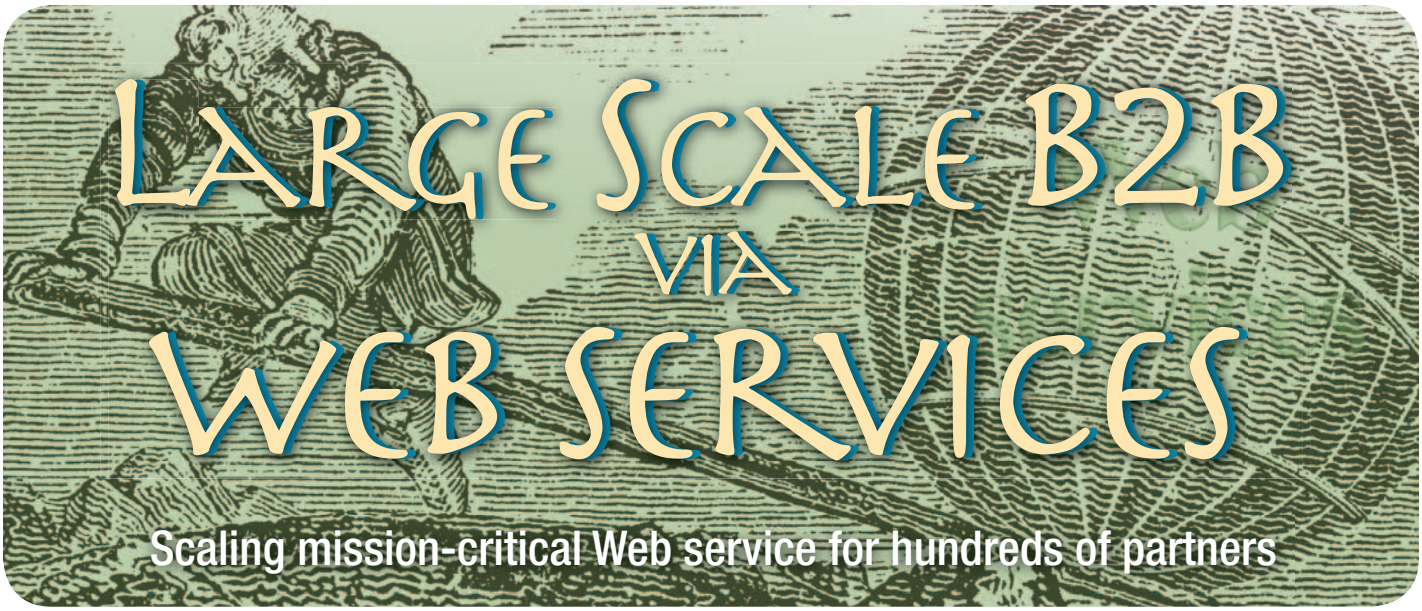
Technology

- Effective BPM technologies come "out of the box" with the business and application integration functionality required to support an end-to-end enterprise process, including business rules management, e-mail task notifications, ODBC database read/write capability, and offering a read/write capability from common desktop tools (i.e., Excel, Word, etc.).
- Putting the power of process design, execution, and management into the hands of business people means delivering BPM functionality through configuration, not code lessening the burden on IT resources.

About the Author

Jeffrey D. Mills is the vice president of channel development and partner enrichment at Bluespring Software. He began his career with 3M Corporation building eBusiness infrastructure. After 3M, Jeff joined Net Perceptions where he built analytics-based solutions for the industrial marketplace. Jeff is a graduate of Miami University in Ohio and is president of the Cincinnati chapter of the International Association of Microsoft Certified Partners (IAMCP). Jeff can be reached at jeff.mills@bluespringsoftware.com or at www.bluespringsoftware.com.

jeff.mills@bluespringsoftware.com



Large Scale B2B via Web Services

Scaling mission-critical Web service for hundreds of partners

Over the past few years Web services have grown from a conceptual draft based on the ideal of cross-platform programmatic interoperability to a formal specification and a vision of grand-scale distributed system architectures. Today we are putting that vision to work and designing systems with service-oriented architectures that are critical to our businesses. Amazon.com has been one of the forerunners in the adoption and promotion of Web services. Consistently striving to remain on the forefront of technology, Amazon has thoroughly embraced Web services - so much so in fact, that you can tie some serious revenue numbers directly to the external use of Amazon's Web services. For the remainder of this article I will outline for you just one of the avenues Amazon has taken to leverage Web services to generate new business.

WRITTEN BY
ALLAN WESSELS

Most of us are familiar with Amazon.com and have come to notice that other companies offer many of the products sold on Amazon; it isn't uncommon at all to browse direct product offerings from dozens of merchants such as Bombay and Sharper Image on Amazon.com. In fact third-party sellers play a critical role for Amazon in terms of breadth and depth in product offering, but how does the product information of these third-party sellers get there? Also, how are business-critical order, inventory, and price data exchanged reliably and in a timely manner for hundreds of merchants utilizing various platforms and technologies?

The answer: Amazon has developed a highly scalable SOA platform that leverages industry-standard technologies and incorporates the ability to conduct secure transactions, thus effectively allowing third-party sellers to fully control product listings directly on Amazon.com. A true service-oriented architecture allows merchants the ability to put their products in front of millions of Amazon customers; in return Amazon captures a commission on the sale and everyone is happy. The challenge of building a platform to handle hundreds of business partners with millions of transactions took some serious considerations into account.

Knowing that business partners would be utilizing various technologies to programmatically exchange data required not only a standard for the data format but also a security standard for the data transmission as well. By relying on XML schemas to define the structure of the documents to be exchanged (HTTPS as the transport protocol, and SwA [SOAP with Attachments] as the communication protocol), those key transport issues are addressed. Amazon publishes to its third-party sellers the XSD specification to each message Amazon sends and receives. A third-party seller has only to implement the XSD, extract their product data, transform it per the XSD, and post it to Amazon with a SOAP message over HTTPS.

Another design consideration involved the assumption that business partners would have a myriad of system designs

XSLmaker

The Smart Way to Transform XML to Web Pages

1. Define XML
2. Drop XML on Web Page
3. XSL and CSS are generated

Done.

100% WYSIWYG

Transform XML => Web pages

Features

Visual tool with Canvas	✔
Auto-generates XML tree based on sample XML data	✔
Drag-n-drop from XML tree directly onto the Canvas	✔
Generates XSL code & generates CSS code	✔

Advanced Features

Merge multiple XML sources	✔
Include live XML/RSS feeds in Web page	✔
Include other Web pages in Web page	✔
Use other Web languages with XSL in Web page	✔
Supply run-time input parameters to XSL Web page	✔

XSLmaker

The only comprehensive XSL development platform for efficient making of professional Web pages and Web applications from XML data. Easy and Intuitive.

FREE trial. Download at www.xslmaker.com

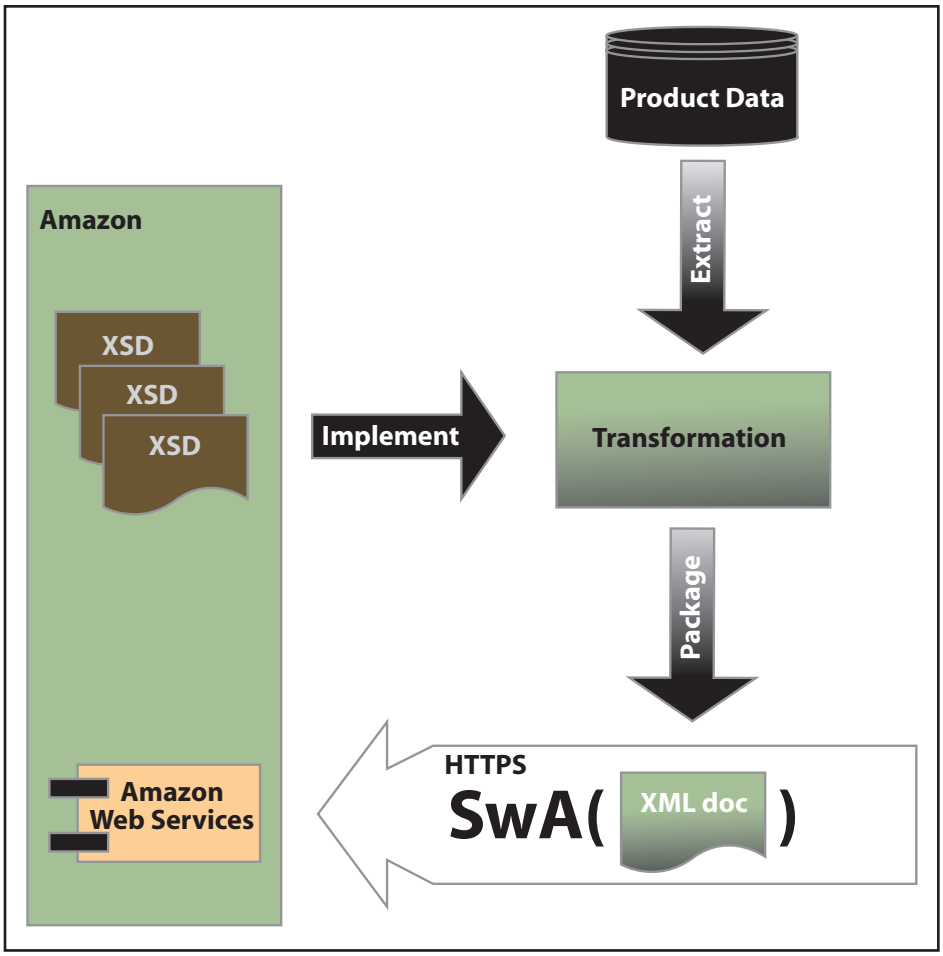


FIGURE 1 | Implementation architecture

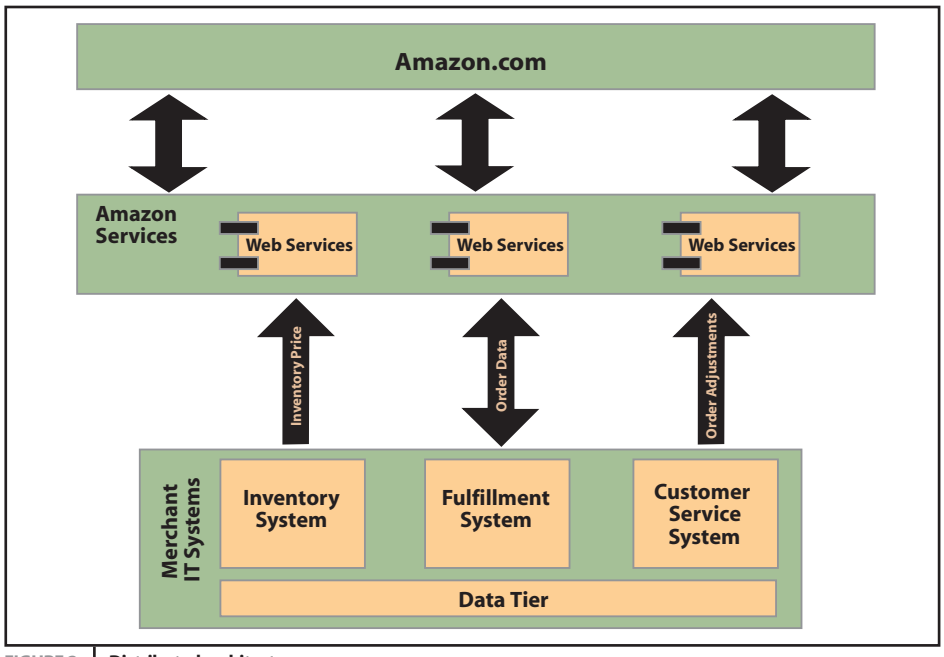


FIGURE 2 | Distributed architecture

that if addressed individually would require custom data definitions, connectors, and message exchange choreography for each partner. This forced the idea that the Amazon platform must be agnostic with regard to partner systems; nothing better lends itself to being agnostic than a service-oriented architecture. In some cases the disparity in partner systems can be so extreme that a system component such as fulfillment, which is critical to business operations, can be owned by a separate company at a different physical location. By encapsulating services in the Amazon interface the partner has significant flexibility in their integration design and can easily incorporate disparate systems.

Because real dollars are impacted with every message exchange, all message exchanges had to be secured and guaranteed. All exchanges utilize HTTPS and require three credentials for validation against an existing account. In order to ensure business-critical documents such as order reports are exchanged successfully an exchange acknowledgement has been incorporated. As displayed in the diagram below, the first operation is a service request for existing document IDs. The return value if null delays for 15 minutes and then requests again. If the return is positive it will contain an array of document IDs. At this point we step into the scope of a document exchange. By calling the getDocument service and passing a document ID, the return result will be the physical document represented by the ID, in this case an order report. Now the document provider (Amazon) knows a request has been made by the requestor (merchant partner) and has responded appropriately by sending the document, but does not know if the document has reached the requestor successfully. The requestor now needs to ensure the successful receipt of the document and perform the final operation of acknowledging to the provider that the document has been received successfully. Subsequently the provider can transfer the document ID from the "outbound" queue to the "sent" queue. In all, this design ensures the state

of a document is managed by the requestor who is responsible for follow up operations – in this case fulfillment of the order.

By implementing the system designs in Figure 3, Amazon has been able to successfully integrate product offerings from hundreds of merchants – from small domestic companies to large multinational corporations, Amazon's single platform scales to support them all.

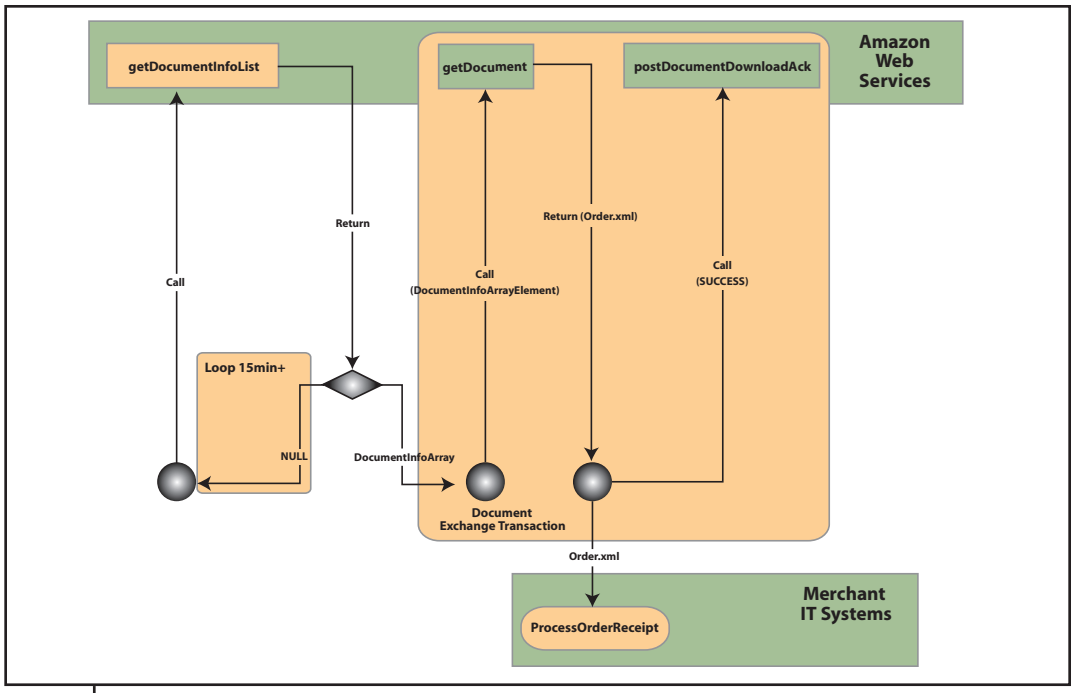


FIGURE 3 **Message exchange**

About the Author

Allan Wessels has over nine years of diverse experience in software development, from software engineering to project management, with stops at Sony Pictures, Microsoft, and most recently, Amazon.com.

wessels@amazon.com

The Keys to Successful SOAs

Aligning business and technology to achieve enterprise goals

Today most of the conversations surrounding service-oriented architectures (SOAs) focus on flexibility and breaking down applications into services: modular, reusable, componentized, with increased availability to the services as well as increased management of them. However, with these conversations comes the risk of getting sucked into a technology-centric vacuum in which consideration for the real business problems that customers need to solve might be neglected.

WRITTEN BY
MICHAEL LIEBOW

Undoubtedly there is a huge demand for the development and implementation of SOAs. Gartner predicts that by 2008 more than 60 percent of organizations will use SOA as a "guiding principal" when creating essential applications and processes. With this in mind, and as the spotlight on SOA grows ever brighter throughout the industry, companies in all sectors are posing the question, "What are the key things we need for successful SOAs?"

Before embarking on building an SOA, business and IT executives need to sit down and determine what business problems need to be solved. These are clearly going to vary by industry. For example, Telco and wireless service providers worry about customer turnover. Pharmaceuticals stay up at night trying to get new, life-saving drugs to market faster. Hospitals want to ensure that up-to-date patient records are available to doctors when needed during emergencies.

Solving real business problems using the flexibility afforded by an SOA is critical to helping customers transform to on-demand businesses that can quickly respond to rapidly changing market environments. An SOA can help companies do this by providing an industry-standard framework that is interchangeable, adaptive, and flexible, but most important, is closely linked to the business goals.

Business leaders who are not aware of the benefits an SOA can provide will likely lose a competitive edge in the marketplace as more nimble competitors take advantage of this new enabling technology. The business value that SOAs provide is so great that organizations are set to spend billions of dollars annually in just a couple of years on software and services to achieve these benefits.

Working through the steps of identifying a specific business problem, determining an approach to solve it, realizing ROI, and then moving on to the next business problem is a method that customers understand and prefer when purchasing technology and services. This approach is forcing the strategy to shift away from the practice of customers simply purchasing middleware, a database, management tools, or even general services from a single vendor, to customers demanding solutions based on IT components with the right combination of technology and services to solve specific business problems. Vendors that can't adapt to this new culture will have problems. They must improve themselves to sell large quantities of components and assets that fit into a defined ecosystem – based on industry standards – and scale these components to address additional business problems.

Customers need to approach building an SOA based on the needs of the business. A company, or more specifically an IT department, can't guess what new technology (such as Web services) will add the greatest value, so a detailed identification and prioritization of the services a business needs to develop or expose in order to support improved business processes must be developed. Once a business has determined its priorities and those priorities are

> "Every service should be created with the intent to bring value to the business in some way"

understood and shared by the company's business and IT leaders, the next step is to agree upon a systematic approach that will help them build a roadmap for implementing an SOA. In order to migrate to an SOA that will generate the greatest results in the most efficient manner, there are certain questions that must be addressed. In a recent paper, Ali Arsanjani, PhD and IBM's chief architect for IBM's SOA and Web Services Center of Excellence, outlined the following points for best practices in migrating to SOA.

- ***Adoption and maturity models.*** Where is your company at in the relative scale of maturity in the adoption of SOA and Web services? Every different level of adoption has unique needs.
- ***Assessments.*** Have pilots been conducted? Have you dabbled into Web services? How good is the resulting architecture? Should you keep going in the same direction? Will this scale to an enterprise SOA? Have you considered everything you need to consider?
- ***Strategy and planning activities.*** How do you plan to migrate to an SOA? What are the steps, tools, methods, technologies, standards, and training needed? What is the roadmap and vision, and how do you get there? What's the plan?
- ***Governance.*** Should existing API or capability become a service? If not, which ones are eligible? Every service should be created with the intent to bring value to the business in some way. How is this process managed without obstructing business goals?
- ***Implementation of best practices.*** What are some tried and tested ways of implementing security, ensuring performance, complying with standards for interoperability, and designing for change?

Once these questions have been addressed, the first phase of the SOA roadmap should be materializing. However, this is also the time when the architecture itself will need to be mapped out and a template for the architecture will need to be created, and design and architecture decisions will need to be determined for each of the following seven layers.

> "Business leaders who are not aware of the benefits an SOA can provide will likely lose a competitive edge in the marketplace as more nimble competitors take advantage of this new enabling technology"

Layer 1: Operational systems layer. This consists of existing custom-built applications, otherwise called *legacy* systems, including existing CRM and ERP packaged applications, and *older* object-oriented system implementations, as well as business intelligence applications. The composite layered architecture of an SOA can reuse existing systems and integrate them using service-oriented integration techniques.

Layer 2: Enterprise components layer. This is the layer of enterprise components that is responsible for realizing functionality and maintaining the quality of service of the exposed services. These special components are a managed, governed set of enterprise assets that are funded at the organization or the business unit level. As enterprise-scale assets, they are responsible for ensuring conformance to SLAs through the application of architectural best practices. This layer typically uses container-based technologies such as application servers to implement the components, workload management, high-availability, and load balancing.

Layer 3: Services layer. The services the business chooses to fund and expose reside in this layer. They can be *discovered* or they can be statically bound and then invoked, or possibly, choreographed into a composite service. This service exposure layer also provides for the mechanism to take enterprise scale components, business unit specific components, and in some cases, project-specific components, and externalizes a subset of their interfaces in the form of service descriptions. Thus, the enterprise components provide service realization at runtime using the functionality provided by their interfaces. The interfaces get exported out as service descriptions in this layer, where they are exposed for use. They can exist in isolation or as a composite service.

Level 4: Business process composition or choreography layer. Compositions and choreographies of services exposed in Layer 3 are defined in this layer. Services are bundled into a flow through orchestration or choreography, and thus act together as a single application. These applications support specific use cases and business processes. Here, visual flow composition tools can be used for the design of application flow.

Layer 5: Access or presentation layer. Although this layer is usually out of scope for discussions around an SOA, it is gradually becoming more relevant. I depict it here because there is an increasing convergence of standards, such as Web services, for Remote Portlets Version 2.0 and other technologies, that seek to use Web services at the application interface or presentation level. You can think of it as a future layer that you need to take into account for future solutions. It is also important to note

that SOA decouples the user interface from the components, and that you ultimately need to provide a complete solution from an access channel to a service or composition of services.

Level 6: Integration (ESB). This layer enables the integration of services through the introduction of a reliable set of capabilities, such as intelligent routing, protocol mediation, and other transformation mechanisms, and is often described as the ESB. Web Services Description Language (WSDL) specifies a binding, which implies a location where the service is provided. On the other hand, an ESB provides a location-independent mechanism for integration.

Level 7: Quality of Service. This layer provides the capabilities required to monitor, manage, and maintain QoS such as security, performance, and availability. This is a background process through sense-and-respond mechanisms and tools that monitor the health of SOA applications, including the all-important standards implementations of WS-Management and other relevant protocols and standards that implement quality of service for an SOA.

Several vendors are offering a variety of what they like to call SOA products. However, an SOA is the architecture of an IT environment that is designed and built to meet specific business needs; assuming an SOA can be installed out of a box is naïve. Instead, there is a plethora of different tools and services that an organization can seek to adopt on its path to successful SOA implementation. For example, a software foundation that is built on open standards, such as Web services and J2EE, allows customers to build and integrate applications within a service-oriented architecture, to create reusable services out of their existing Web services and packaged applications, as well as to combine services to link business process with software applications.

In addition, there is a variety of services that are available to help companies prepare for their SOA implementations, including:

- Assessment services are designed to help customers determine business processes that can be changed using an SOA and develop a plan to implement an SOA based on the existing IT environment.
- Planning, strategy, and execution consultations help customers develop a plan and architecture for SOA that provides a framework to execute business processes. These consultations help to define a reference model architecture for SOA building blocks that identifies, manages, and delivers business services, and develops an SOA transition plan to help customers achieve an SOA
- Some services are designed to help companies assess if there is value in using existing legacy systems and if so, how legacy systems can best be used within an SOA.
- Component business modeling services help customers map business processes across entire industries and break down a business into its component parts to better identify areas for process improvement. The flexibility that an SOA provides allows for business process improvements that can be implemented quickly.
- Enablement services help to identify how an SOA can help meet customers' business goals, infuse flexibility into IT assets, foster an alignment between business and IT, and help to avoid costly missteps by assuring that the technologies implemented will meet future needs.
- SOA design services help customers create a specific plan and architectural design for their SOA.
- Implementation services help customers build entire SOAs or deploy targeted aspects of a customer's SOA vision and roadmap.
- There are also services that provide ongoing performance monitoring, maintenance, and requirements verification, which helps to assure the SOA is proactively managed for optimal business value.

With all of this demand, it's important to clarify that SOAs can't be achieved by purchasing a single product that has been dubbed with the letters "SOA." SOA is not a product; rather, it is an architecture that bridges the gap between business and IT through a set of business-aligned IT services using a set of design principles, patterns, and techniques.

As a result, SOA is leading a titanic shift in how customers approach buying and implementing technology, thus creating the need for a massive culture change for vendors. Because of this culture change, many large-scale IT projects will ultimately be replaced with small building blocks based on components and reusable assets that can be scaled in a fraction of the time – and money.

The value of SOA won't be realized through the large individual projects that are typical of standard services engagements, at least not in the very near future. Instead, customers are focused on smaller engagements – months or even weeks in length – that address specific business problems such as helping insurance companies

streamline claims processing, helping telecoms produce a single customer bill, and helping retailers keep store shelves optimally stocked. Successful smaller engagements that solve the identified business problem will lead to multiple consecutive engagements with the same customers.

While there is no crystal ball that can tell a company what kind of ROI it will achieve on a project before it is deployed, anytime a company embarks on a new IT project there is pressure to convince the top executives that the technology investment will be a fruitful one.

Despite the relative novelty of SOA projects, many customers have achieved tremendous ROI in short periods of time, sometimes even before the implementation is complete. One example is Guardian Life Insurance Company of America. As the fourth largest mutual life insurance company in the United States, Guardian offers a variety of products through independent agents, including life insurance, disability insurance, and equities. However, each insurance offering had been running on a different IT system, which made it difficult for agents to access the information they needed to sell products and services to customers. Guardian built an SOA to combine these systems and provide its agents with a single view of their customers in an effort to increase sales. Guardian's SOA project was responsible for a 30 percent boost in agent productivity and achieved 100 percent ROI before it was fully completed, virtually paying for itself.

Another example of ROI that has stemmed from a successful SOA implementation involves a large credit card company. The company launched a Web services/SOA project that allowed it to shorten the billing cycle from more than 30 days to fewer than 30 days. Shaving just a few days off the billing cycle resulted in a payback of several hundred million dollars over a three-month period.

Finally, a telecommunications company found that its billing system wasn't reliably tracking the correct usage and billing amounts. As this scenario was largely due to its decentralized billing system, the company launched an SOA project to link its disparate systems into a single, unified system. One year after the SOA was implemented the company had saved half a billion dollars.

While these numbers are impressive, they are most certainly at the high end of the ROI spectrum. The key is to choose your first SOA project carefully, to really focus on a specific business problem and solve that problem, because improving it will result in the most immediate, clearest, and largest ROI.

Successful business process selection, planning implementation, and ROI aside, once you've completed your first SOA project, evolving the SOA across the company frees up IT resources and helps to ensure that investments in technology are focused on core capabilities aimed at growing the business. As companies continue to adopt and apply SOAs, it's worth noting that a complete SOA solution won't be delivered by a single vendor. Existing IT infrastructures are too complex and contain too many different technologies for any single vendor to credibly claim that products in the back of their truck will meet every need and fix every problem a customer has. Vendors will need to partner up or they won't survive this cultural shift. Customers will need a trusted partner to walk them through this transformation – one that will successfully partner with multiple vendors to create the technology components customers need.

About the Author

Michael Liebow is the vice president of Web Services and SOA for IBM Global Services and he views Web services as the critical enabling technology for IBM's On Demand vision. Michael has a broad background that makes him well suited to bring new concepts to market. He has held a variety of sales, marketing, and management positions within a diverse range of industries, including the high-technology sector, consumer packaged goods, media and entertainment, and advertising industries.

Join us at the Inaugural

Gartner Business Process Management Summit 2005

For the first time,experience:

Top speakers. Case studies. Insightful analysts. Great networking.
All for one of the fastest growing areas of today's business—BPM

June 6-7, 2005 • Hyatt Regency Crystal City • Arlington, VA

The First Summit to Help You Achieve Business Value through Process Excellence and IT Agility

Featuring an agenda built to address the top BPM issues of business and IT professionals, including:

- Justifying and measuring BPM
- Managing change and organizing for BPM
- Business Process Modeling and Business Rules
- Compliance, quality and customer-driven initiatives
- BPM technology and products

Steven Stanton
Author and pioneer of process-centered innovation and process improvement consultant

Thomas Malone
MIT Sloan School of Management, entrepreneur, inventor and strategist

Jeff Hiatt
Author and consultant on business process reengineering and change management

Jim Sinur
Vice president and distinguished analyst, Gartner

Gartner BPM Summit: Your Best Investment in 2005

gartner.com/us/bpm

Gartner
Business Process
Management Summit

Service Orienting BPM

Service-Oriented Process Management

Business process management (BPM) is a business management philosophy consisting of the idea that a business can be understood and managed solely in terms of business processes. BPM is process-centric and responsive to change in business requirements and objectives. Businesses are defined through dynamic processes that change almost constantly.

WRITTEN BY
NEERAJ KULKARNI

Workflow Management Coalition Terminology and Glossary defines a business process as follows:

- As being associated with operational objectives and business relationships such as claims process.
- As having a defined triggering (initiation) conditions for each new process instance (e.g., the arrival of claim) and defined outputs at its completion.
- May involve formal or informal interactions between participants; its durations may vary widely.
- May consist of automated activities and is capable of workflow management and/or manual activities that lie outside the scope of workflow management.

Data-Centric vs. Process-Driven Applications

The IT applications defined and implemented before the advent of SOA are mostly data-centric; that is, geared towards data capture, storage, and retrieval. The architecture is mostly technology driven and not process-centric. This resulted in applications being less adaptable to changes in business process or costlier to maintain and keep up with than changes in the business process. Often business processes were modeled to fit the applications (to save on costs of changing), thereby making them ineffective and inefficient. Dr. Michael Hammer, the process-reengineering guru, in his famous book *Reengineering the Corporation* said, "Rip it out and start over." Reengineering enabled better process definition in a given framework, but it did not give a dynamic and agile architecture. (More recently Dr. Hammer said, "I was wrong.")

For an enterprise to be effective and responsive to changes in the business process, the approach for enterprise architecture should be to:

- Externalize processes, separating them from applications and providing tools to simplify process design, implementation, and changes.
- Design applications in the form of services that are a part of the end-to-end process and have them access external data that the process needs.

This approach allows the business processes to interact with and coordinate the services' behavior for the execution and completion of the process and interaction with other processes. This is a paradigm shift from applications to processes, thus making processes the centerpiece of enterprise architecture (EA).

Enterprise Architecture

The definition of architecture from ANSI/IEEE Std 1741-2000 is: "the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution."

EA is a conceptual tool that assists organizations with the understanding of their own structure and the way they work. It provides a map for the enterprise and it's a route planner for business and technology change.

Normally, EA takes the form of a comprehensive set of cohesive models that describe the structure and the functions of an enterprise. The most commonly used perspectives of EA are business, application, information, and technology perspectives.

Service-Oriented Architecture (SOA)

Typically, SOA is equated to Web services, but that is not the case. SOA is an architecture style that promotes business process orchestration of enterprise-level business services. The three major elements of SOA are:

- **Services:** SOA models the enterprise as a collection of business services, which can be accessed across the enterprise. Services are self-contained in the sense that they perform specific functions and can be invoked using a standard protocol.
- **Processes:** Business processes orchestrate the execution of these enterprise services to fulfill required business functionality. Enterprise-wide processes essentially define enterprise operations.
- **Organization:** Organization owns all of the SOA artifacts and governs their creation, usage, access, and maintenance.

A service is defined and exposed to the outside world in terms of its interface. The service interface consists of a service name and set of methods that the service supports. Each interface is defined by a set of parameters (defining data required for interaction with the service) and a communication protocol used for data transfer and actual service invocation. The methods in the service interface are grouped together based on the business functionality requirements of the service.

Services provide the business logic as well

as the state management relevant to the business functionality. Services encapsulate the logic and data associated with a real-world process, similar to OO encapsulation. Since services are called across a network, the services should wrap a substantial body of application logic, thereby justifying the latency cost of network requests. A service should enable the performance of a complete function through a single request and fewer interfaces, rather than exposing many interfaces that manipulate small amounts of data.

SOA for Enterprise Architecture

Let's analyze the impact of SOA on all four perspectives of enterprise architecture.

Business Perspective

By using SOA it is possible to decompose business functionality into more manageable, reusable parts that can be independently designed, developed, and maintained. SOA enables the definition of the enterprise functionality in terms of services and processes. The business and application perspectives can be aligned better in the sense that it is easier to trace the application perspective back to the business perspective, and thereby easier to implement required changes in functionality. This facilitates the separation of business services that do not often change from processes – the fast-changing entities of a business model.

Business activity monitoring (BAM), the real-time monitoring of business events and transactions, becomes easier by separation of functionality and applications. BAM delivers real-time intelligence about integrated enterprise to management. BAM is typically integrated into the execution of business process.

Before even thinking of designing and implementing business functionality, it is essential to create a model of the business that identifies the fundamental entities and the business processes. Modeling tools using UML can be effectively used to model the business, the business process, and entities. UML enables the complete end-to-end modeling of business, from business process to implementation entities. During this process of business process modeling, services, and the interfaces to them for each process, are identified.

In an ideal case, an enterprise architect would like to engineer the business process and the applications from scratch, but that is not the reality. Mapping the required services onto an existing application is an important activity.

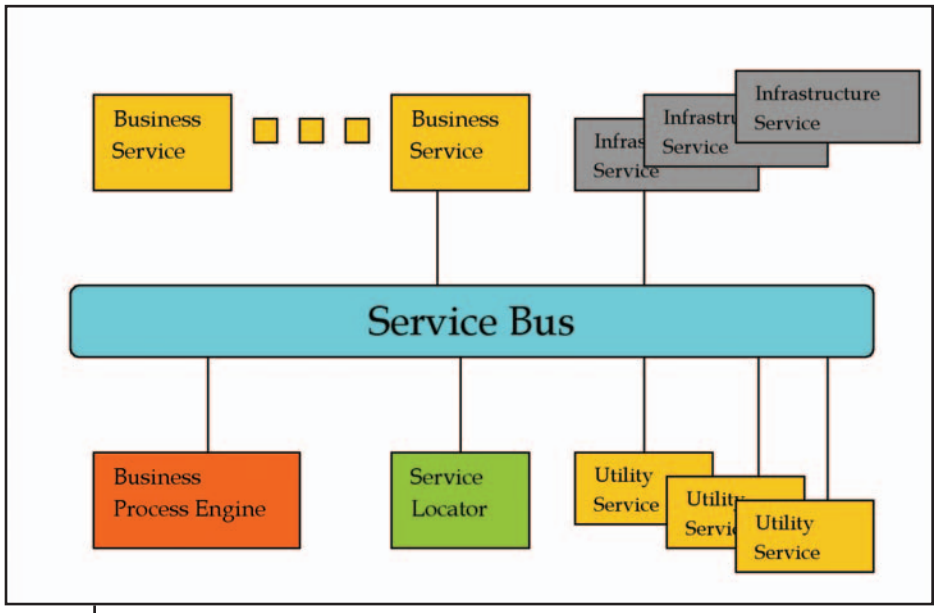


FIGURE 1 **Typical SOA application architecture**

If an application exists that supports required functionality, then the functionality needs to be wrapped to provide interfaces that can be exposed and used outside of the application. Service implementation will typically delegate execution to one or more existing applications and supplement it with new code.

Application Perspective

Every service can be independently designed, constructed, and maintained; hence, services provide a loosely coupled system landscape. The use of services makes the application portfolio easy to maintain. Figure 1 shows a typical application architecture that consists of services. The application architecture is composed of disparate services communicating with one another over the service bus. The salient features of this architecture are:

- ***Business Process Engine:*** Allows for the externalization of the business process. The introduction of a separate business process engine provides a separation of business process definition and execution from service implementation, thus allowing for loose coupling in the application architecture.
- ***Service Locator:*** Allows for externalization of the service location, thus supporting service location transparency.
- ***Business Services:*** Encapsulate the actual business functionality.
- ***Utility services:*** These are special kinds of business services that don't belong to a firm's core business, but can still be accessed by any client.
- ***Common Infrastructure Services:*** Provide system and infrastructure support for business services.

Business process modeling followed by design, development, deployment, and maintenance of a robust service-oriented application infrastructure often results in upfront investment, as compared to stand-alone application development. However, the maintenance of point-to-point communication between stand-alone applications results in higher maintenance costs.

Information Perspective

The SOA implementation introduces two data repositories:

- The data models to support service implementation(s)
- A data dictionary for service messages, the defining message semantics of the SOA

This separation of the information perspective simplifies the enterprise data and

information strategy. In an SOA implementation, the data model exposed by the enterprise application portfolio and used by internal and external processes is actually the messaging data dictionary. So there is a decoupling from the data model used by the services, hence the services can in fact use completely different internal data models as long as the data semantics they are exposing adhere to the semantic data dictionary of the enterprise. The messaging data dictionary is often much simpler than the internal data models used by the services, so integrating services through messages becomes significantly easier.

Technology Perspective

The technology perspective is the most important perspective because it forms the foundation for building enterprise scale services. It provides the framework for the development and maintenance of services. SOA requires developing an infrastructure that can be shared by many services delivering against diverse functional requirements.

It is almost impossible to throw away existing IT services and start on a clean state. In reality, services are created by integrating and extending existing applications. Components are aggregated to wrap existing applications into services.

Figure 2 shows a typical service implementation that exposes existing legacy system functionality. Like a typical EAI implementation, this implementation does not expose the functionality directly, but encapsulates the functionality in its own implementation. This enables:

- Extending the legacy system functionality without modifying the existing legacy systems
- Making the service more granular by combining the functionality of multiple legacy systems (or multiple interfaces of the same legacy system) and implementing additional functionality

The SOA approach differs from traditional EAI initiatives. Most EAI initiatives are driven by IT and lean more towards point-to-point connectivity of an enterprise application portfolio. SOA encapsulates existing applications into services, thereby facilitating the convergence of IT with business. Such an approach typically leads to an enterprise architecture blueprint for an organization that is aligned with business objectives without requiring the overhaul of existing applications. It is always possible to create business process through EAI using existing applications. But this leads to making the applications a part of the business process. A service as described above introduces a layer of abstraction between existing applications and processes. Figure 3 shows an example of a service-oriented architecture for an organization.

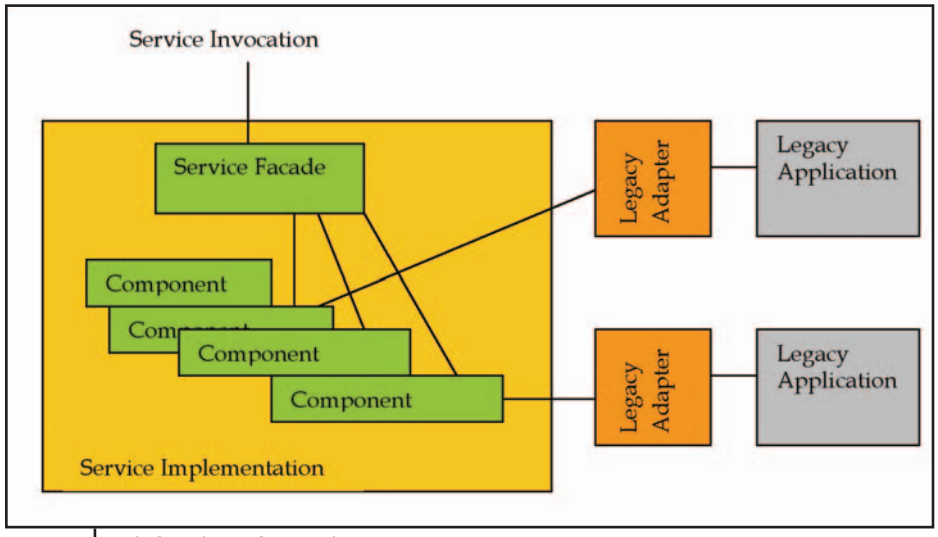


FIGURE 2 **Typical Service Implementation**

Orchestrating Service Interaction

The goal of an enterprise SOA is to build a flexible, durable infrastructure that makes it easy to integrate any set of applications at any time to solve business challenges. The SOA must make it possible to dynamically add and reconfigure services on the fly and to alter the logical flow between services as processes change. It must also be possible to use multiple interaction models between services. For example, it should be possible to use messaging semantics such as publish-subscribe to collect logging data from several sources while using BPM-style orchestration to implement long-running automated business process. In an event-driven SOA, a service is not aware of these interactions; it simply consumes and emits messages that represent events.

The enterprise service bus (ESB) provides the necessary unified middleware and the framework for hosting commercial and custom services. ESBs use message-oriented middleware (MOM) as a primary means of communication. In an ideal ESB, the messaging capabilities can be used without requiring the services to include messaging logic in their code. The messaging capabilities are configured outside of the applications.

ESBs also include intelligent routing capabilities through content-based routing. Most ESBs also come with service-oriented process engines that support complex, stateful orchestrations. In its 2003 Predictions Series, industry analyst firm Gartner Inc., said:

> *A new category of integration middleware called the enterprise service bus (ESB) has emerged to support the proliferation of service-oriented interactions between enterprise applications. An ESB is a standards-based integration backbone that combines messaging, Web services, transformation, and intelligent routing to reliably connect and coordinate the interaction of hundreds of application endpoints spanning a global organization.*

In the same report, Gartner predicts that a majority of large enterprises will have an ESB running by year 2005.

Data transformation occurs in several different places across an SOA. Data is normalized into canonical form as it enters the SOA

[Engage and Explore]

the technologies, solutions and applications that are driving today's **Web services** initiatives and strategies...

www.sys-con.com/edge2005

web services EDGE conference&expo

FALL SERIES

CALL FOR PAPERS NOW OPEN!*

Coming to a City Near You ►

Web Services Edge

Fall Conference Series

The Westin Washington, D.C.
Washington, D.C.
September 7-8, 2005

1

3 Dynamic Conference Programs

Targeting Major Industry Markets

20+ seminars within 5 tracks will address the hottest topics & issues:

- Web Services: The Benefits and Challenges
- Web Services Security
- SOA (Service-Oriented Architecture) and ESB (Enterprise Service Bus) Strategies
- Interoperability, Incremental Integration, & Open Source
- The Management Process in Developing a Web Services Strategy

Program Features:

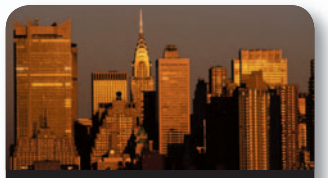
- Keynotes
- Tutorials
- Panel Discussions

Why Attend:

- Improve the return on your technology investment
- Develop & sharpen your strategy and identify key action steps
- Find new ways to reach and impress customers with Web services
- Maximize the power of your enterprise
- Protect your business from security threats
- Assess Web services as a viable option

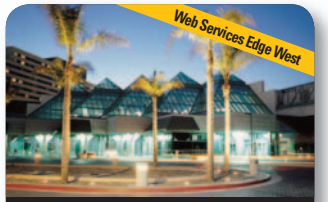
Attention Exhibitors:

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



Helmsley Hotel
New York, NY
September 19-20, 2005

2



The Westin Santa Clara Convention Center
Santa Clara, CA
October 24-25, 2005

3

Register Today! www.SYS-CON.com/Edge2005

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: grisha@sys-con.com

For Exhibit and Sponsorship Information ► Call 201 802-3066

Produced by SYS-CON EVENTS

and manipulated via transformation as it is routed between applications and services. The normalized data format across an SOA is mostly XML. All ESBs support XML transformation through configurable XSLT-based service.

Tracking, auditing and logging are very important functions in an SOA. This is commonly referred to as BAM. Auditing data as it flows within and between organizations is a legal requirement for an increasing number of processes. ESBs can provide several service-oriented capabilities and frameworks for tracking, auditing, and logging. An enterprise SOA shares data and business logic between departments within an organization and across organizations. It creates automated processes that span multiple security and administrative domains. Several issues surrounding security arise while implementing SOA. Most organizations and departments within an organization have their own schemes for authentication and authorization of users. Similarly, there may be different encryption requirements either at the data or the data channel level. The SOA infrastructure must be able to route data across multiple security domains and support various security schemes.

SOA, EAI, and Web Services

What is the difference between EAI and SOA? EAI designates a set of integration solutions, while SOA serves as an intermediary between well-integrated IT systems and well-defined business processes. SOA encourages the definition and deployment of generalized business function (a top-down approach driven by business needs), rather than point-to-point connections between legacy systems (a bottom-up approach for solving tactical integration issues).

Is SOA another name for Web services? SOA is an architectural style, while Web services represent a set of standards that enables various architectures, of which SOA is an important one. Web services can be used to create inefficient point-to-point integration code. With the support for Web services becoming widespread and most development environments providing built-in support for easy creation and deployment of Web services, it has arguably become much easier now to write poorly designed code.

The key principle behind an SOA is defining and exposing loosely coupled, normalized, and coarse-grained business functionality in a manner independent of underlying implementation. Executed properly, SOAs are well suited to application integration scenarios because they expose applications as a series of APIs that can be assembled with other APIs to construct a consolidated end-to-end solution.

The ultimate objective in defining an SOA should be to improve an organization's ability to respond to changing business models. Well-abstracted services provide a set of normalized business functions aligned with an enterprise roadmap of core strategic business process.

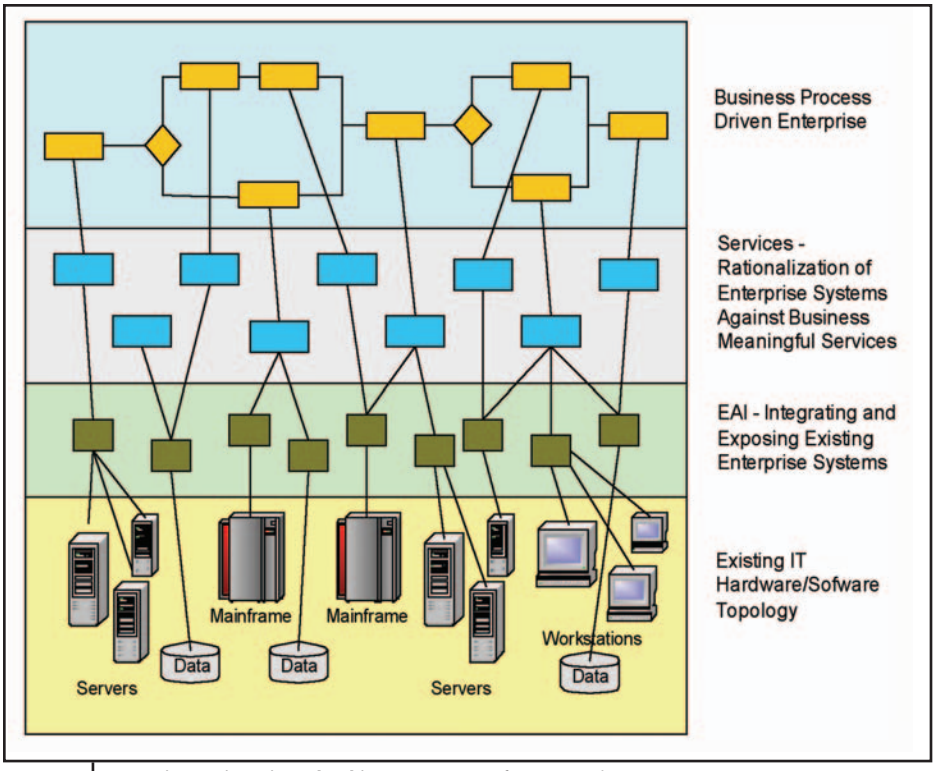


FIGURE 2 | **Enterprise service-oriented architecture** *Source: Infosys Research*

Conclusion

SOA bridges the gap between diverse applications and addresses the increasing need for the flexibility of BPM. The services layer, introduced by SOA, allows for direct mapping of business artifacts into an existing application portfolio. If implemented correctly, SOA can provide a highly effective approach to equip an enterprise with a set of appropriately abstracted and normalized business services. Web services standards are just a vehicle to help deliver SOA. The combination of these elements makes SOA a model as enterprises seek to increase business flexibility.

This article is adapted from an article in the SETLabs Briefings issue on BPM (Volume 2, No. 3), published by Infosys Technologies at www.infosys.com/Technology/technology_briefings.asp.

References

- Workflow Management Coalition Terminology and Glossary, Workflow Management Coalition, February 1999
- Microsoft Architecture Overview, Michael Platt, Microsoft Corporation (MSDN), July 2002
- Lublinsky, B. and Tyomkin, D. "Dissecting Service-Oriented Architectures." *Business Integration Journa*l. October 2003.
- Predicts 2003: Enterprise Service Buses Emerge, Gartner Inc., December 2003
- 21st Century Business Architecture, Howard Smith and Peter Fingar, BPMI.org, May 2003

About the Author

Neeraj Kulkarni is a senior technical architect with the Technology Consulting Group of Infosys Technologies Limited. He has several years of experience in product development and implementing projects that involve EAI and Web services.

neeraj_kulkarni@infosys.com

Subscribe Today!

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

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

Editorial Mission

Greater Collaboration and Efficiency Through Education

- ✓ ***ISSJ*'s editorial mission is to showcase proven solutions** that will guide, motivate, and inspire senior IT and business management leaders in the planning, development, deployment, and management of successful enterprise-wide security and storage solutions.
- ✓ ***ISSJ*** brings together all key elements of data storage and protection, and presents compelling insight into the benefits, efficiencies, and effectiveness gained by focusing on these two critical areas of IT simultaneously.
- ✓ ***ISSJ*** is an objective, critical source of information that helps storage and security managers make informed management decisions about what is working today, and what they need to plan for tomorrow, and is the only publication that focuses exclusively on the needs of IT professionals who are driving the enterprise storage architecture/infrastructure while pursuing and incorporating the latest security technologies.
- ✓ ***ISSJ*** achieves our mission by delivering in-depth features, practical "how-to" solutions, authoritative commentary, hard-hitting product reviews, comprehensive real-world case studies, and successful business models, written by and for professional IT storage and security practitioners.

SAVE 50% OFF!
(REGULAR NEWSSTAND PRICE)

Only $39.99
ONE YEAR 12 ISSUES

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

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

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

XML JOURNAL SUPPLEMENT

This Month

Use a Native XML Database for Your XML Data

Gregory Burd & Kimbro Staken

Developers tend to use the most familiar technologies. For data storage, that is the relational database. During design it's easy to see tables of data everywhere; however, not everything is relational in nature. When dealing with XML data or data easily expressed as XML, XQuery-based native XML databases (NXDs) present a viable and cost-effective alternative to relational databases, file system storage, or custom developed storage implementations. So, when is it time to consider an NXD?

Computational XSLT for Financial Statements

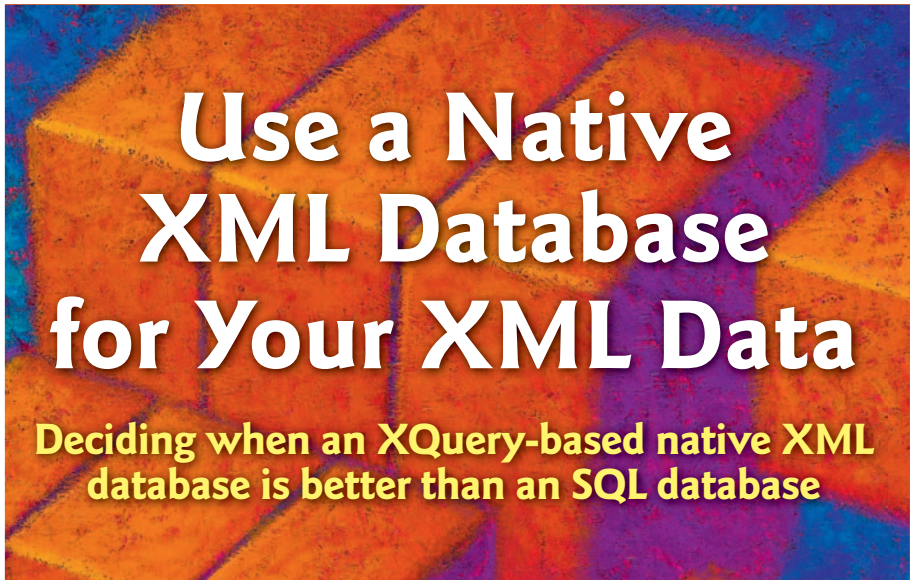
Ed Gimzewski

Schema languages like XBRL (the Extensible Business Reporting Language) can define the structure of a financial statement, and the data itself can be saved as an XML instance of the schema. The data is often processed further using formulas; for example, to verify balances and derive data for financial analysis. This processing is traditionally done in a non-XML application, but staying in the world of XML, I ask: How well can XSLT (the Extensible Stylesheet Transformation Language) do these calculations and what are the advantages?

Rx for Structured Product Labeling

Dan Dube

The long-term benefits of XML-based Structured Product Labeling (SPL) are creating many short-term headaches for drug makers as they scramble to comply with the Food and Drug Administration (FDA) deadline for electronically processing all drug-related content.



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

Use a Native XML Database for Your XML Data

Deciding when an XQuery-based native XML database is better than an SQL database


WRITTEN BY
GREGORY BURD & KIMBRO STAKEN

Developers tend to use the most familiar technologies. For data storage, that is the relational database. During design it's easy to see tables of data everywhere; however, not everything is relational in nature. When dealing with XML data or data easily expressed as XML, XQuery-based native XML databases (NXDs) present a viable and cost-effective alternative to relational databases, file system storage, or custom developed storage implementations. So, when is it time to consider an NXD?

Can native XML databases really provide a better answer for your data storage needs? In this article we'll examine some guidelines to help answer that question.

When to Consider a Native XML Database

1. Do you have thousands of XML files?
2. Is your XML data larger than 200MB?
3. Are you trying to build a hierarchy into tables?
4. Could your data change over time?
5. Have you spent more that $100 on books explaining SQLXML?

The File System Isn't a Database

The first two questions are practical in nature. If estimates indicate more than 1000 XML files or 200MB of XML data exist, the file system isn't the right tool for the job. File systems are not built to manage large numbers of files in a single directory or huge directory hierarchies. Managing concurrency, out of disk space conditions, and other common problems will plague your application unless you use a database.

Naturally Hierarchical

The third reason to consider an NXD really has to do with the impedance mismatch between relational databases and XML data. Fundamentally, XML data is hierarchical and is a poor match for a relational database's rows and columns. Relational databases have always had a hard time modeling hierarchies. You'll find dozens of workarounds for this, but no real simple and efficient solution. Any XML-to-relational mapping tool will have to pick one of these techniques and manage this common case as best as possible. Regardless of the solution, performance will suffer. The end result will also be more brittle over time. Changing the structure of the data – which is easy, natural, and useful to do in XML – forces a redesign of the relational database and changes to the mapping layer. A single mapping mistake could completely skew an entire data set. As an XML document structure changes over time, it's possible for attributes to become elements, and vice versa. Over time, incremental changes to the logical structure of the XML data can force physical changes to the database, and wholesale dump and reload may be required of the relational system to keep pace.

Structured Yet Flexible Data

The fourth question suggests that requirements change over time, something true of most real world business systems. Once a relational database schema has been set in stone, only the database administrator (DBA) is qualified to change it without disrupting services. The contract between a relational database and the program that use it becomes the weakest link in the system. As requirements change, the DBA will spend endless hours mitigating the issues that arise. Contrast that with XML databases. XML is both structured and flexible. Even XML documents conforming to a DTD or XML Schema maintain a high degree of flexibility when compared to relational schemas. Most NXDs will optionally validate document structure. Even when document validation is not enforced, XML documents maintain a high degree of implicit structure. Therefore in either case, XML documents are flexible and structured and as a result the contract between the database and the programs using it is not brittle. It can withstand change without requiring a DBA.

Data Mapping Is Wasted Time, Money, and Effort

The last question is really a reality check. Take a second to consider the amount of time and money you've spent trying to make a solution workable. The best thing you can do when digging a hole is to stop digging and get out of the hole. With that in mind, let's look at SQLXML. If you need to mix and match SQL data and XML data, it's not a horrible way to go. However, if you view it as a way to squeeze XML into a relational system in which you've already invested, you might want to reconsider. You are going to pay a performance penalty for every document stored in terms of CPU and memory. Your ability to query, index, and optimize will be impacted as well. Executing XQuery against XML data mapped into relational tables will be hindered by the non-native storage format. An optimized native XML database won't have the same penalty.

Let's take a look at how to use an NXD to learn more about its advantages over a relational database.

Examples and Code

Berkeley DB XML is an open source XQuery implementation built atop the Berkeley DB transactional database system. It supports optimized XML storage, XQuery query planning, massive scale and concurrency, and is available for download as source code for multiple platforms and as a Windows installer from Sleepycat Software (www.sleepycat.com). Berkeley DB XML has also been integrated with Stylus Studio if you're more comfortable using an IDE for development. Berkeley DB XML is readily available to anyone, so we'll use it for the following examples.

First let's create an in-memory container for XML documents. Follow along using the "dbxml" command line provided with Berkeley DB XML.

```
dbxml> createContainer ""
dbxml> putDocument myDoc <names><name>joe</name><name>fred</
name><name>jane</name></names>
dbxml> query collection('')/names/name[.='joe']
dbxml> print
<name>joe</name>
```

Line 1 creates the container, while line 2 places a simple document into the container. The third line performs a query on the container returning each document that matches the XPath query /names/name[.=joe]. Finally it displays what we returned by the query statement.

That's all quite useful, but let's say that you need to access your XML storage programmatically. Because Berkeley DB XML is a library, and as such is linked into your application just as any other library would be, it does not incur the overhead of client/server communication. You interact with Berkeley DB XML using one of the supported language APIs. The primary one is C++, as the product is written in C++. Java, Python, Perl, PHP, and TCL are all supported API languages. Many other languages are supported by third parties and are readily available on the Internet.

With that in mind, let's next try some simple C++ code that calls for the same thing as the dbxml commands used in the previous example (see Listing 1).

The underlined sections of the code relate back to the first example. The first underlined section creates a container. This container is called 'test.dbxml,' and is on disk, and rather than in memory. The next underlined section places the same simple XML document into the new container. The last underlined section issues the query, and the while loop equates to the print statement. Put it all together and you have essentially the same result as before. In Java the code looks much the same; again the underlined areas are the common key sections of code (see Listing 2).

As you can see, the API is fairly straightforward and similar across languages. You'll note that the C++ and Java examples create a database on disk rather than in memory simply by giving the container a name.

Let's move back to the dbxml shell and try a more complicated example. This time let's add a few documents, so we can explore the performance of the system.

First let's populate an imaginary parts database.

```
dbxml> createContainer parts
Creating document storage container
```

That created an empty container and opened it as the default container in the shell. We can now use the putDocument command to run our XQuery and insert the sample data.

```
dbxml> putDocument "" "for $i in (0 to 2999) return
  <part number='{$i}'>
<description>Description of {$i}</description><category>
  {$i mod 10}</category>
{if (($i mod 10) = 0) then <parent-part>{$i mod 3}</
  parent-part> else ''}</part>" q
```

> "Managing XML content can be easy, predictable, flexible, secure, and fast"

A single output line for each document will be printed as confirmation. The generated content looks like this:

```
<part number="999">
    <description>Description of 999</description>
    <category>9</category>
</part>
```

Every 10th document of the 3,000 generated will look something like this:

```
<part number="990">
   <description>Description of 990</description>
   <category>0</category>
   <parent-part>0</parent-part>
</part>
```

The numbers vary to allow more complex queries later. Let's dive in. First, let's find all part records that contain a parent-part, or «from the container named parts select all part elements that also contain a parent-part as a direct child of that element» (see Listing 3).

If you're only interested in the parent-part element, try the following:

```
dbxml> query collection("parts")/part/parent-part
300 objects returned for eager expression 'collection
  («parts»)/part/parent-part'
```

```
dbxml> print
<parent-part>0</parent-part>
<parent-part>0</parent-part>
...
<parent-part>2</parent-part>
<parent-part>2</parent-part>
```

Or if you simply need the value of the parent-part element, execute this query:

```
dbxml> query collection("parts")/part/parent-part/text()
300 objects returned for eager expression 'collection("parts")/
  part/parent-part/text()'
dbxml> print
0
0
...
2
2
```

To find documents without parent-part elements, see Listing 4. To find two specific parts by part number try the following:

```
dbxml> query 'collection("parts")/part[@number =
  1070 or @number = 1032]'
2 objects returned for eager expression 'collection("parts")/
  part[@number = 1070 or @number = 1032]'
dbxml> print
<part number="1070"><description>Description of 1070</
  description><category>0</category><parent-part>2</
  parent-part></part>
<part number="1032"><description>Description of 1032</
  description><category>2</category></part>
```

Listing 5 shows it's equally simple to find a range of parts.

Now let's explore the effect of indices. Your query times may differ as your machine is likely different from ours, but the improvement should be somewhat comparable.

```
dbxml> setVerbose 1
dbxml> query collection("parts")/part[parent-part]
Query       - Finished eager query execution,
  time taken = 2399.15ms
300 objects returned for eager expression 'collection(«parts»)/
  part[parent-part]'
```

Spending 2.4 seconds to examine 3,000 documents and find the matching 300 isn't bad for a file system, but this is a database engine. Indices should improve performance. Berkeley DB XML indices are specified in four parts: path type, node type, key type, and uniqueness. In this case we need to index the node elements to determine if something is present or not, and we won't expect this pattern to be unique. Let's try the query again, but this time with an index to optimize searches for documents containing a parent-part element, a "node-element-presence-none" index.

```
dbxml> addIndex "" parent-part node-element-presence-none
Adding index type: node-element-presence-none to node:
  {}:parent-part
dbxml> query collection("parts")/part[parent-part]
Query       - Finished eager query execution,
  time taken = 176.925ms
300 objects returned for eager expression 'collection(«parts»)/
  part[parent-part]'
```

From 2.4 seconds to just under 1/5th of a second seems like a worthwhile improvement. Let's try another query.

```
dbxml> query 'collection("parts")/part[parent-part = 1]'
Query       - Finished eager query execution,
  time taken = 215.387ms
100 objects returned for eager expression 'collection(«parts»)/
  part[parent-part = 1]'
```

The query executed in just over 1/5th of a second without an index. Since this query searches for a specific node's number, a "node-element-equality-decimal" index should help speed things up.

```
dbxml> addIndex "" parent-part node-element-equality-decimal
Adding index type: node-element-equality-decimal to node: {}:
parent-part
dbxml>  query 'collection("parts")/part[parent-part = 1]'
Query       - Finished eager query execution,
  time taken = 79.973ms
100 objects returned for eager expression 'collection(«parts»)/
  part[parent-part = 1]'
```

Wonderful, query execution time has been reduced by more than half, to under 1/10th of a second by adding that second index. Let's try one final example – the range query we tried earlier.

```
dbxml>  query 'collection("parts")/part[@number >
  100 and @number < 105]'
Query       - Finished eager query execution,
  time taken = 5917.83ms
4 objects returned for eager expression 'collection("parts")/
  part[@number > 100 and @number < 105]'
```

Six seconds is an eternity in database terms, so let's create another and try again.

```
dbxml>  addIndex "" number node-attribute-equality-decimal
Adding index type: node-attribute-equality-decimal to node:
  {}:number
dbxml>  query 'collection("parts")/part[@number >
  100 and @number < 105]'
Query       - Finished eager query execution,
  time taken = 85.99ms
4 objects returned for eager expression 'collection("parts")/
  part[@number > 100 and @number < 105]'
```

The index reduced the query time back down to less than a 1/10th of a second.

Conclusion

Pick the right tool for the job. An XML-centric design will benefit from a native XML database. Managing XML content can be easy, predictable, flexible, secure, and fast. Save yourself the headaches, time, and expense of solutions that are less suited to the task.

Author Bios

Greg Burd works as a product manager for Sleepycat Software and is responsible for managing Berkeley DB XML. He is a technology industry veteran with more than a dozen years of design, development, and product management experience at companies such as NeXT Computer, which was acquired by Apple Computer, and the JavaSoft group at Sun Microsystems.

Kimbro Staken is an independent consultant, author, and open source developer specializing in technologies for XML data management. He is one of the primary developers of the dbXML Core Open Source native XML database and a cofounder of the XML:DB Initiative.

gburd@sleepycat.com

kstaken@xmldatabases.org

Listing 1

```
#include <string>
#include <fstream>
#include "dbxml/DbXml.hpp"

using namespace std;
using namespace DbXml;

int
main(int argc, char **argv)
{
  try {
    XmlManager mgr;
    XmlUpdateContext uc = mgr.createUpdateContext();
    XmlQueryContext qc = mgr.createQueryContext();
    XmlContainer cont = mgr.createContainer("test.
dbxml");

    cont.putDocument("myDoc", "<names><name>joe</
name><name>jane</name></names>", uc);
    XmlResults res = mgr.query("collection('test.
dbxml')/names/name[.='joe']", qc);
    XmlValue value;
    while (res.next(value))
      cout << "Value: " << value.asString() << endl;
  } catch (XmlException &e) {
    std::cout << "Exception: " << e.what() << std::
endl;
  }
  return 0;
}
```

Listing 2

```
import java.io.*;
import com.sleepycat.dbxml.*;
import com.sleepycat.db.*;

class testSimple
{
public static void main(String args[]) throws
Throwable {
    try {
 XmlManager mgr = new XmlManager();
 XmlUpdateContext uc = mgr.createUpdateContext();
 XmlQueryContext qc = mgr.createQueryContext();
 XmlContainer cont = mgr.createContainer(«test.dbxml»);

 cont.putDocument(«myDoc»,
                  "<names><name>joe</
name><name>jane</name></names>»,
                  uc);
 XmlResults res = mgr.query(
     "collection('test.dbxml')/names/name[.='joe']",
     qc);
 while (res.hasNext()) {
     XmlValue val = res.next();
     System.out.println("Value: " + val.asString());
 }
    }
    catch (XmlException e) {
 System.out.println("XmlException: " + e.toString());
    }
    catch (Throwable t) {
 System.out.println("Threw: " + t);
    }
}

}
```

Listing 3

```
dbxml> query collection("parts")/part[parent-part]
300 objects returned for eager expression
'collection(«parts»)/part[parent-part]'
dbxml> print
<part number=»540»><description>Description of 540</
description><category>0</category><parent-part>0</par-
ent-part></part>
<part number=»30»><description>Description of 30</
description><category>0</category><parent-part>0</par-
ent-part></part>
...
<part number=»990»><description>Description of 990</
description><category>0</category><parent-part>0</par-
ent-part></part>
<part number=»480»><description>Description of 480</
description><category>0</category><parent-part>0</par-
ent-part></part>
```

Listing 4

```
dbxml> query collection("parts")/part[not(parent-part)]
2700 objects returned for eager expression
'collection("parts")/part[not(parent-part)]'
dbxml> print
<part number="22"><description>Description of 22</
description><category>2</category></part>
<part number="1995"><description>Description of 1995</
description><category>5</category></part>
...
<part number="2557"><description>Description of 2557</
description><category>7</category></part>
<part number="2813"><description>Description of 2813</
description><category>3</category></part>
```

Listing 5

```
dbxml> query 'collection("parts")/part[@number > 100
and @number < 105]'
4 objects returned for eager expression
'collection("parts")/part[@number > 100 and @number <
105]'
dbxml> print
<part number="101"><description>Description of 101</
description><category>1</category></part>
<part number="102"><description>Description of 102</
description><category>2</category></part>
<part number="103"><description>Description of 103</
description><category>3</category></part>
<part number="104"><description>Description of 104</
description><category>4</category></part>
```

WRITTEN BY **ED GIMZEWSKI**

Computational XSLT for Financial Statements

A new role for the Extensible Stylesheet Transformation Language?

Schema languages like XBRL (the Extensible Business Reporting Language) can define the structure of a financial statement, and the data itself can be saved as an XML instance of the schema. The data is often processed further using formulas; for example, to verify balances and derive data for financial analysis. This processing is traditionally done in a non-XML application, but staying in the world of XML, I ask: How well can XSLT (the Extensible Stylesheet Transformation Language) do these calculations and what are the advantages? Using this language to create data is called *Computational XSLT* and it opens the way to distributing financial formulas as a set of equivalent XSLT functions that are readable and run on any XSLT processor. No new standard is proposed but a new role for XSLT 1.0/2.0 or XQuery1.0 (the XML Query Language) is suggested, and this role is compatible with data in XBRL as well as with any other XML format.

AUTHOR BIO

Edmund Gimzewski has 15 years of experience in developing software for the financial sector and has been working with XML since its inception. For the last five years he has specialized in XML-centric systems for general software development and in defining and applying financial and other calculations. Edmund formerly worked as a scientist researching thermodynamics and has published nearly a dozen papers on the subject (see Thermochimica Acta).

Financial Statements: Structure and Data

The items in a financial statement (e.g., current assets and fixed assets) have labels that have an agreed meaning within the accounting standard being applied (e.g., US-GAAP). XBRL defines the items using XSchema and linkbases that use XLink (the XML Linking Language). The schema define the items and their types, and the linkbases contain additional information; for example, while the schema refers to an item by an ID, a presentational linkbase links that ID to a readable label for final presentation. Having defined the items, an instance document assigns data to the items for a particular company at particular points in time. In an XBRL instance document, this point in time and company label for an item's value is called the *context* of the item's value.

Formulas

Experts in financial data create formulas. These experts must also define the logic for handling missing data and other complications. Without that logic, the formulas can't be evaluated as intended. Formulas are used for data validation and analysis. There are many private and some third-party formula languages being applied to financial data. However there is common ground, as shown by the *XBRL Formula Requirements* initiative, which provides an analysis of the requirements and has use cases to show how the proposed formula language might work. The formulas are held in calculation linkbases.

XSLT As the Formula-Processing Application

With the input data in place as an instance of an XBRL or some other schema, calculations can be made using the formulas defined in linkbases or other XML or non-XML mark-up languages. The processing application that produces the calculated data usually works from formulas in non-XML scripts, but in the world of XML, an XSLT processor is a standardized application that can transform XML instance data to calculated data using readable instructions in the XSLT language. So to harness the power of XSLT, the missing link is to transform a set of formulas into an XSLT file.

The XSLT file then provides a way of sharing a readable, run-anywhere implementation of formulas and their processing logic. For example, such a file could be shipped with data (input plus calculated) to show how the calculated data was derived; or the user can apply the formulas to his own data in batch mode on a server, or individually in a browser or desktop application. Perhaps the biggest advantage of this approach is that it offers a relatively direct route from relational data to calculated data if it's used in conjunction with the XML extract-and-transform (via XSLT) support now offered by most database vendors (e.g., Oracle's XSQL API).

Objectives

Having made a case for XSLT for financial statement calculations, I will show the steps by which formulas in a schema are transformed into an XSLT file in which each formula has a corresponding XSLT function. The XSLT functions are then applied to input data to give calculated data. A simplified case study will show the essential components and then the complexity is raised to a higher level by introducing the real-world need to handle missing data. Finally, I will cover the performance of computational XSLT based

on my experience in applying hundreds of formulas to thousands of statements.

Although XBRL schemas, linkbases, and instances could have been used for the case study, for brevity, and to focus on the essentials, I have merged and condensed the XBRL schema and formula linkbase into a single pseudo-schema that lists the items for both input data and calculated data (with formulas). The simplified instance document has the period as the only XBRL *context*.

The work was originally done using XSLT 1.0 but I have moved it to XSLT 2.0 to exploit the support in XSLT 2.0 for XPath 2.0 sequences, unrestricted data structures, and Regular Expressions. References to XSLT *functions* should be understood to mean a function in XSLT2.0, or a template in XSLT 1.0 (or 2.0), or a function in XQuery 1.0.

Creating XSLT from Formulas

In terms of files with self-explanatory names, the components needed to create XSLT functions from formulas in a schema are as follows. Example files are listed in the source code and will be discussed later, but first we will introduce their roles.

- ***Schema.xml*** defines the structure of the financial statement in terms of input items and calculated items. A calculated item contains a formula used for the calculation; the formula refers to other items (input or calculated) in the schema.
- ***Instance.xml*** is where data is assigned to the input items defined in *schema.xml*; the data for an item is one or more values, each with a context (period). *Instance.xml* also contains the contexts.
- ***Compiler.xslt*** is hand coded and transforms the formulas in *schema.xml* into an XSLT file, *functions.xslt*, where each formula becomes an equivalent XSLT function. The "compiler" analogy refers to the analysis of *schema.xml* and the generation of equivalent callable XSLT functions. Each distinct *schema.xml* will be compiled to a distinct equivalent *functions.xslt* file.

To apply the formulas, the end user needs *schema.xml* to create a compatible *instance.xml*, and *functions.xslt*, whose functions they will call from their own XSLT. In the source code, *host.xslt* shows how the functions in *functions.xslt* are used.

The interactions of these components are shown in Figure 1, which emphasizes the two distinct roles of XSLT in the whole process: as a compiler to create *functions.xslt*, and for numeric computations when using *functions.xslt*. The compilation must be re-run after any change to a formula and is unlikely to be time-critical; but the numeric computations, which may be run in a browser, desktop, or server application, should be as efficient as possible.

Case Study: Simplified Use of Computational XSLT 2.0

The simplification is that missing (null) data should be treated as having a value of zero. The file *schema.xml* (Listing 1) has three input and four calculated items. When transformed to *functions.xslt* (Listing 2), a calculated item becomes a <xsl:function> element with the same name as the item. In *functions.xslt* the namespace prefix *formula* is used for these auto-generated formula-derived functions to distinguish them from the fixed helper functions (namespace *helper*) they call (see below). The input data in *instance.xml* (Listing 3) is assumed to be for a single company.

The structure of a formula function in *functions.xslt* is simple: it has one parameter, the *context_id*, which is simply a reference to the period of the statement being evaluated from *instance.xml*. It contains an XSLT variable for each distinct argument in the item's formula, and it returns the formula, evaluated as written. For example, *formula:F10* is the formula function for:

```
<item id="F10" formula="$F1 + $F2"
  type="calc"/>
```

and *formula:F10* contains variables $F1 and $F2, created like this:

```
<xsl:variable name="F1"
 select="helper:get_input_value
  ( 'F1', $context_id)"
   as="xs:double"/>
```

and it returns:

```
<xsl:sequence select="$F1 + $F2"/>
```

A more complex example is:

```
<item id="F13" formula="if( ($F10 -
  $F10_prev) gt
   0.01*$F10_prev ) then ($F10 + $F3)
     else ($F10)"
    type="calc"/>
```

In the corresponding function, *formula:F13*, the calculated-item variable *$F10* is populated by calling its identically named formula function with *context_id* as the parameter:

```
<xsl:variable name="F10"
  select="formula:F10( $context_id)"
    as="xs:double"/>
```

In the definition of item *F13*, the *_prev* modifier on *$F10_prev* indicates that the value should be taken for the previous context. Such references are common in financial formulas. The corresponding variable in *formula:F13* is created by:

```
<xsl:variable name="F10_prev"
  select="formula:F10(helper:get_
  previous($context_id))"
 as="xs:double"/>
```

Here the function *helper:get_previous(context_*

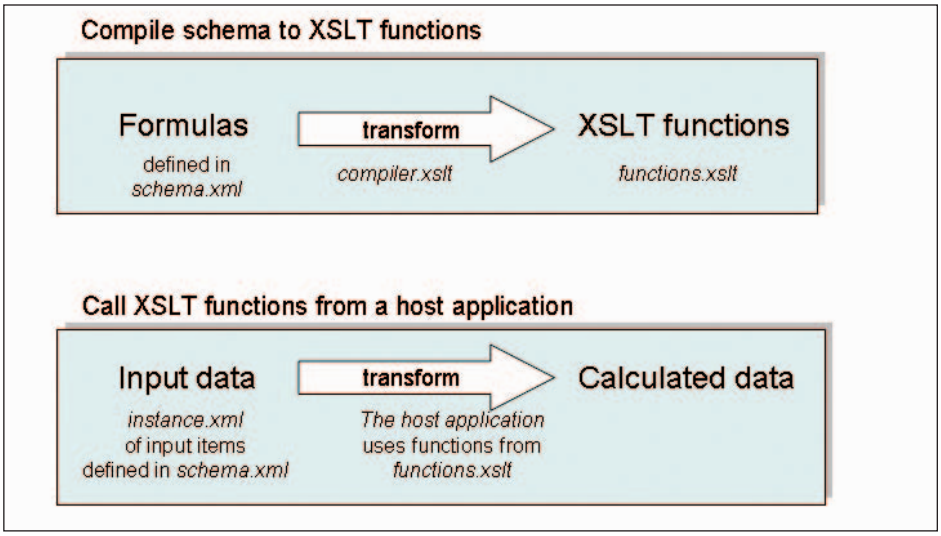


Figure 1 • Two roles for XSLT: creating and calling functions

id) simply returns the *context_id* for a period of one year before that passed to it.

In the source code, *compiler.xslt* (Listing 4) was used to create *functions.xslt* from *schema.xml*; file *host.xslt* (Listing 5) shows how the functions in *functions.xslt* are called, and the file *calculated_data.xml* (Listing 6) shows the result when *host.xslt* transforms *instance.xml*.

This is all relatively straightforward and the formulas in *functions.xslt* obviously have all the flexibility of XPath 2.0 formulas.

Dealing with Missing (Null) Data

Sometimes data isn't available (null) for input items either because the company didn't report it or because a formula needs input data from a previous period and no data are available (inevitable for a period previous to the oldest one). Financial data experts decide how their formulas should handle nulls. For example, the logic for handing nulls must define the result for this sample of expressions unambiguously.

```
10 + null; null + null; 10 div null; null
div 10; if (null ne 10) then (null + 20)
else (30);  if (25 gt null) ..; if(null =
null).., etc..
```

In what follows, the business logic for evaluating formulas is based on practices in my organization; other organizations may have different business logic, but the same technical processing should apply.

Our experts dictate that formulas have a type (see below), and for each type allowable expressions are restricted. These restrictions are required if the null evaluation logic is to be applicable. The revised schema, *schema_complex.xml* (Listing 7), has items with the following four types:

- ***Input type.*** The data is assigned to these in the instance document.
- ***Simple calculation type.*** "Simple" in that there are no comparison operators and the formula must evaluate to a number.
- ***Ratio.*** Has the form of *numerator* divided by *denominator*, where *numerator* and *denominator* are treated as sub-formulas with input or simple types.
- ***Conditional.*** Has the form *if (test is true) then return result from formula_if_true else return from formula_if_false***.** The test part has the form: *(test_left_hand test_operator test_right_hand)*. The sub-formulas *test_left_hand, test_right_hand, formula_if_true*, and *formula_if_false* are input or simple types.

Each calculated-type item also has the attribute *null_eval_rule*, which indicates how to evaluate a simple-type formula, or a simple-type sub-formula in a ratio or conditional type.

Null Logic

Given formulas restricted by these types, the main features of the logic are as follows, where numerical evaluation involves treating a null argument as zero. The reader need not concentrate on the details but should appreciate the growing complexity.

- For a simple type, if any argument refers to a previous period and is null, the formula returns null regardless of any other rule.
- For a simple type with *null_eval_rule*='null_if_all_null,' if there is any non-null argument, the formula is evaluated numerically, otherwise it returns null.
- For a simple type with *null_eval_rule*='null_if_any_null,' if there is any null argument, the formula returns null; otherwise it is evaluated numerically.
- For a ratio item type, *numerator* and *denominator* are evaluated like separate simple types that share the one *null_eval_rule*. If either returns null, the whole formula returns null; otherwise it is evaluated numerically.
- For conditional item types, *test_left_hand* and *test_right_hand* are evaluated as separate simple types that share the single *null_eval_rule*. If only one of *test_left_hand* or *test_right_hand* is null and the operator is *eq* or *ne*, the test result is evaluated as *true operator false*; otherwise, any null *test_left_hand* and/or *test_right_hand* is treated as a zero and the two are compared as values. Having evaluated the test to true or false, the whole formula returns either *formula_if_true* or *formula_if_false* treated as a simple type.

Applying the Null Logic

When the revised schema, *schema_complex.xml*, is transformed, the resulting *functions_complex.xslt* (Listing 8) has many new features to support the null logic. First, the XSchema type returned by the formula function has been changed from *xs:double* to *xs:double?*. The latter allows an empty sequence, the XPath 2.0 equivalent of null. Second, code has been added to help establish the null status of each argument and apply the rule(s) to get the resulting null status of the whole function. File *functions_complex.xslt* is annotated to highlight the new features.

All this may seem inordinately complex, but any formula definition language must define the logic for evaluating a formula in a format that a processing application can analyze and apply, and that is readable. For example, *XBRL Formula Requirements* recognizes the need for the flexible coding of such logic and proposes using a subset of ECMAScript or XPath 1.0. The processing application would then compile that script to its own executable code. However, in our pure XSLT approach, the formula function in *functions_complex.xslt* simply uses helper functions and some additional code in the same file to implement the evaluation logic.

How Well Does Computational XSLT Do on Real-Life Financial Statements?

As explained above, the speed of the compiler stage isn't critical but, for the record, even with rules more complex than the preceding, hundreds of formula functions are created in approximately 10 seconds. It is more important for the compiler to create fast XSLT. XSLT indexes (using *<xsl:key>*) make a massive difference, as does any function-call-caching in the XSLT processor. For example, a function can be called 50 times with the same arguments, but if the result of the first call is cached, the next 49 calls are read from cache. The Saxonica 8.1 parser used here supports caching.

A batch of 8,000 XML instance documents (one per company), each with data for 170 input items at 15 periods, was transformed to 8,000 files with 336 calculated items for the same periods. Some formulas referred to calculated items whose formulas referred to calculated items, up to seven levels above the input data. The null evaluation and period adjustment rules were more complex than those discussed, and annulization and formatting were applied. The 45 million calculations were completed in seven hours, which easily meets our requirements and compares well with other processing routes. The conditions were as follows. Software: Saxonica 8.1 JAR files were incorporated into a lightweight Java 1.4 application; the JVM ran on Windows 2000. Hardware: a PC with a 3GHz processor and 3GB of RAM.

Summary

Computational XSLT is a serious option for applying formulas to financial statement data. A computational XSLT file can be generated from formula definitions and can incorporate the generally complex null-evaluation logic for applying those formulas. The XSLT file provides a way of sharing a readable, run-anywhere

implementation of formulas and their processing logic. For example, such a file could be shipped with data (input + calculated) to show a user how the calculated data was derived, or to let users apply the formulas to their own data. Although not linked to any particular standard other than XSLT (or XQuery 1.0 if preferred), this approach is applicable to XBRL data. This XML-centric approach also offers a direct route to calculated data from data held in relational databases via the XML extract-and-transform (via XSLT) support from the major database vendors.

References

- *XBRL:* www.xbrl.org
- *XBRL Formula Requirements, Public Working DRAFT of Tuesday, 20 April 2004:* www.xbrl.org/technical/requirements/Formula-Req-PWD-2004-04-20.pdf *at Members' Area of* www.xbrl.org

Acknowledgements

The author thanks Fitch Ratings, Ltd., for permission to publish, and Otari Svanidze for his work in implementing the processes described in this article.

ed.gimzewski@fitchratings.com

Listing 1: schema.xml

```
<?xml version=»1.0» encoding=»UTF-8»?> <schema> <!--
==========================================
SCHEMA.XML - created by Ed Gimzewski, Fitch Ratings, 2005-01-12
==========================================
A pseudo schema that has the basic information that XBRL would
hold in an XSchema file plus a formula linkbase.
-->
<!-- Schema for simple case study -->
<!-- items for input data-->
<item id=»F1» type=»input»/>
<item id=»F2» type=»input»/>
<item id=»F3» type=»input»/>
<!-- items for calculated data.-->
<item id=»F10» formula=»$F1 + $F2» type=»calc»/>
<item id=»F11» formula=»($F10 + $F3) div $F2» type=»calc»/>
<item id=»F12» formula=»($F10 - $F10_prev) div $F2»
type=»calc»/>
<item id=»F13» formula=»if( ($F10 - $F10_prev) gt 0.01*$F10_
prev ) then ($F10 + $F3) else ($F10)» type=»calc»/>
</schema>
```

Listing 2: functions.xslt

```
<?xml version=»1.0» encoding=»UTF-8»?><!--
=====================================================
FUNCTIONS.XSLT

Created by transforming SCHEMA.XML with COMPILER.XSLT
=====================================================
-->
<xsl:transform xmlns:xsl=»http://www.w3.org/1999/XSL/Transform»
xmlns:xs=»http://www.w3.org/2001/XMLSchema» xmlns:xdt=»http://
www.w3.org/2003/11/xpath-datatypes» xmlns:compiler=»http://www.
fitchratings.com/compiler» xmlns:formula=»http://www.fitchrat-
ings.com/formula» xmlns:helper=»http://www.fitchratings.com/
helper» version=»2.0»>
<xsl:output indent=»yes» method=»xml» encoding=»UTF-8»/>
<!--================CONSTANTS AND KEYS================-->
<xsl:variable name=»mod_prev» select=»'prev'»/>
<xsl:variable name=»this_doc» select=»/instance»/>
<xsl:key name=»key_get_input_val» match=»/instance/*[name() ne
'context']» use=»concat(@context, '|', name())»/>
<xsl:key name=»key_get_contextFromID» match=»/instance/context»
use=»@id»/>
<xsl:key name=»key_get_contextFromEndDate» match=»/instance/
context» use=»xs:date(period/endDate)»/>
<!--================FORMULA FUNCTIONS================-->
<!--================Simple formulas================-->
<!--============================-->
<xsl:function name="formula:F10" as="xs:double">
<xsl:param name="context_id" as="xs:string?"/>
<xsl:choose>
<xsl:when test="empty($context_id)">
<xsl:sequence select="xs:double(0)"/>
</xsl:when>
<xsl:otherwise>
<xsl:variable name="F1" select="helper:get_input_value( 'F1',
$context_id)" as="xs:double"/>
<xsl:variable name="F2" select="helper:get_input_value( 'F2',
$context_id)" as="xs:double"/>
<xsl:sequence select="$F1 + $F2"/>
</xsl:otherwise>
</xsl:choose>
</xsl:function>
<!--============================-->
<xsl:function name="formula:F11" as="xs:double">
<xsl:param name="context_id" as="xs:string?"/>
<xsl:choose>
<xsl:when test="empty($context_id)">
<xsl:sequence select="xs:double(0)"/>
</xsl:when>
<xsl:otherwise>
<xsl:variable name="F10" select="formula:F10( $context_id)"
as="xs:double"/>
<xsl:variable name="F3" select="helper:get_input_value( 'F3',
$context_id)" as="xs:double"/>
<xsl:variable name="F2" select="helper:get_input_value( 'F2',
$context_id)" as="xs:double"/>
<xsl:sequence select="($F10 + $F3) div $F2"/>
</xsl:otherwise>
</xsl:choose>
</xsl:function>
<!--============================-->
<xsl:function name="formula:F12" as="xs:double">
<xsl:param name="context_id" as="xs:string?"/>
<xsl:choose>
<xsl:when test="empty($context_id)">
<xsl:sequence select="xs:double(0)"/>
</xsl:when>
<xsl:otherwise>
<xsl:variable name="F10" select="formula:F10( $context_id)"
as="xs:double"/>
<xsl:variable name="F10_prev" select="formula:F10(helper:get_
previous($context_id))" as="xs:double"/>
<xsl:variable name="F2" select="helper:get_input_value( 'F2',
$context_id)" as="xs:double"/>
<xsl:sequence select="($F10 - $F10_prev) div $F2"/>
</xsl:otherwise>
</xsl:choose>
</xsl:function>
<!--============================-->
<xsl:function name="formula:F13" as="xs:double">
<xsl:param name="context_id" as="xs:string?"/>
<xsl:choose>
<xsl:when test="empty($context_id)">
<xsl:sequence select="xs:double(0)"/>
</xsl:when>
<xsl:otherwise>
<xsl:variable name="F10" select="formula:F10( $context_id)"
as="xs:double"/>
<xsl:variable name="F10_prev" select="formula:F10(helper:get_
previous($context_id))" as="xs:double"/>
<xsl:variable name="F3" select="helper:get_input_value( 'F3',
$context_id)" as="xs:double"/>
<xsl:sequence select="if( ($F10 - $F10_prev) gt 0.01*$F10_prev
) then ($F10 + $F3) else ($F10)"/>
</xsl:otherwise>
</xsl:choose>
</xsl:function>
<!--================HELPER FUNCTIONS================-->
<xsl:function name="helper:get_input_value" as="xs:double">
<xsl:param name="id" as="xs:string"/>
<xsl:param name="context_id" as="xs:string?"/>
<xsl:choose>
<xsl:when test="empty($context_id)">
<xsl:sequence select="xs:double(0)"/>
</xsl:when>
<xsl:otherwise>
<xsl:variable name="val" select="$this_doc/key('key_get_input_
val', concat($context_id, '|', $id) )"/>
<xsl:sequence select="if (empty($val) or $val eq '') then xs:
double(0) else xs:double($val)"/>
</xsl:otherwise>
</xsl:choose>
</xsl:function>
<!--================================-->
<xsl:function name="helper:get_previous" as="xs:string?">
<xsl:param name="context_id" as="xs:string"/>
<xsl:variable name="context" select="$this_doc/key('key_get_
contextFromID', $context_id )"/>
<xsl:choose>
<xsl:when test="empty($context)">
<xsl:sequence select="()"/>
</xsl:when>
<xsl:otherwise>
<xsl:variable name="endDate" select="$context/period/endDate"
as="xs:date"/>
<xsl:variable name="adjustedDate" select="$endDate - xdt:yearMo
nthDuration('P1Y')"/>
<xsl:variable name="newNumericContext" select="$this_doc/
key('key_get_contextFromEndDate', $adjustedDate )"/>
<xsl:sequence select="if( empty($newNumericContext) or $newNu-
mericContext eq '') then () else $newNumericContext/@id"/>
</xsl:otherwise>
</xsl:choose>
</xsl:function>
</xsl:transform>
```

For Listing 3-8, go to www.sys-con.com/webservices/sourcec.cfm

WRITTEN BY **DAN DUBE**

Rx for Structured Product Labeling

Three headaches for drug companies to avoid in converting to XML systems

The long-term benefits of XML-based Structured Product Labeling (SPL) are creating many short-term headaches for drug makers as they scramble to comply with the Food and Drug Administration (FDA) deadline for electronically processing all drug-related labelingcontent.

Managing product information – shepherding huge volumes of content through multiple internal and external hurdles for drugs on the path from the research lab to the marketplace – is an ongoing challenge for pharmaceuticals. Although accurate and consistent product information is as vital as the drug itself, today's cumbersome systems for handling such information make it increasingly difficult to serve all channels consistently.

The FDA plans to complete the regulations, standards, and systems needed to switch labeling content from PDF to SPL, an XML schema, for prescription drugs by late 2005, and for all drugs by 2006. Without question, converting product-labeling documents to SPL is a major challenge. The question for drug makers is whether to treat the conversion of product-labeling documents to an Open Source digital format as a headache or an opportunity.

Faced with increasing requirements to put more information on packages and share it with pharmacists and doctors, drug companies are confronted with two distinct choices: 1) try to apply a band-aid by "enhancing" existing legacy systems to support minimal SPL compliance without analyzing the entire content supply chain, or 2) re-engineer business processes and the technology infrastructure by implementing XML-based content management systems to leverage the full benefits of content reuse and single-source publishing.

Pharmaceuticals As Publishers

Few external documents are as critical to a pharmaceutical company as product labeling. Closely regulated by the FDA and other international regulatory agencies, these documents deliver critical information to doctors, pharmacists, and patients about drug interactions as well as instructions on how to take the medicine. They also go far beyond product inserts; labeling documents often contain thousands of printed pages detailing the drug's chemical composition, the medical conditions it addresses, and the potential side effects or contra-indications.

Managing product information has become far more complex both inside pharmaceutical companies and outside in the healthcare system they serve. In today's global healthcare companies, numerous departments across the organization create and use product information for distinctly different purposes. New research and new uses for drugs often bring changes in labeling. For many companies, far-flung global operations make it difficult to coordinate all of the activities that affect labeling, despite ubiquitous information technology.

The complexity only increases when product information finds its way to the FDA, onto labels and Web sites, and into promotional literature. Many constituencies – physicians, pharmacists, researchers, regulators, and patients – access the information through a variety of channels and depend on it to guide them in making critical decisions. As time has passed, the systems for gathering product label information and disseminating it to pharmacists, doctors, and the public have grown more complex and more susceptible to error.

All of these factors greatly multiply the potential for errors, inconsistencies, and liability. For example, a drug produced in multiple formulations requires separate packaging and labeling information for each formulation. Should labeling for one formulation mistakenly turn up in a different one, the pharmaceutical company could face a variety of adverse consequences, including an FDA recall. Similarly, if a company updates a labeling change in the package insert but not on its Web site, a physician or a patient could inadvertently make important decisions on the basis of faulty or incomplete information.

Just like a publishing company, pharmaceuticals have to create, manage, publish, and distribute content. The major publishers have already discovered what SPL is forcing big pharmaceuticals to figure out – that they can drastically reduce the cost of their content supply chain by implementing XML-based content management systems.

Why XML?

In 1999, the FDA began allowing the electronic submission of product labeling information in Portable Document Format (PDF) files that enabled the agency to process, review, and archive labeling

AUTHOR BIO

Dan Dube is director of business analysis at Innodata Isogen, a provider of content supply chain solutions. Innodata Isogen (www.innodata-isogen.com) optimizes content supply chains, helping clients realize significant cost savings and productivity gains from content-related operations, achieve better outcomes, and compete more effectively in demanding global markets.

content electronically. In response to the recommendations of the Institute of Medicine and the National Committee on Vital and Health Statistics as well as mandates in the Medicare Modernization Act of 2003, the FDA created a new role for electronic labeling information – to support health information management technologies such as electronic prescribing and the electronic health record (EHR).

When the FDA determined that using PDF documents would no longer adequately support those initiatives, it published the last electronic labeling rule in late 2003 that required the submission of the content of labeling in electronic format for marketing applications for drugs and biologics. Working with interested parties in Health Level Seven (HL7), a standards development organization, the agency adopted a proposed standard called Structured Product Labeling for describing the content of prescription drug labeling in an XML document.

An SPL document consists of an XML document that contains the text and images in an approved prescription package insert (e.g., the labeling content), along with additional information for machine processing of label content (e.g., header information and data elements). A set of files collectively referred to as a stylesheet converts the SPL XML file to a human-readable format. The stylesheet displays the information in the XML file in a consistent format for viewing.

SPL offers numerous advantages over PDF, enabling, for example:

- The automated exchange of information among disparate IT platforms, applications, and interfaces
- A more efficient exchange of changes to labeling content
- The comparison of text and specific data elements
- The exchange of information needed for other submissions, such as drug listing, eliminating redundant data collection, and improving efficiency
- Reuse of common components of content between labels and other relevant documents (e.g., promotional literature, packaging, Web site content, package inserts for different formulations, and submissions to regulatory bodies outside the U.S.)
- The rendering of labeling content into multiple output formats (e.g., PDF and HTML) from the same XML source file

Originally designed to meet the challenges of large-scale electronic publishing, XML has emerged as an increasingly important means of exchanging a wide variety of data over the Web and elsewhere. XML is a simple application-neutral mark-up language designed to provide information in a structured format. One of its primary benefits is its ability to store any kind of structured document that can be shared between different operating systems. Most important, it also preserves the narrative structure of SPL documents so they can be published in standard presentation outputs like PDF and HTML that can be read by both humans and machines.

By separating content from the limitations of various formats, XML supports the reuse of the same information in various documents and channels. Pharmaceutical companies can maintain product information in a single source format, ensuring that product information remains consistent whether the output is a package insert, packaging, a Web site, marketing material, or a submission to the FDA. For example, a change in product indications made in one section would automatically be published to all appropriate outputs both print and electronic.

Moreover, XML's uniform formatting and the centralization of content control it affords lets companies search all their product information more efficiently and use it more effectively in all contexts. Given XML's excellent track record of supporting single-source publishing systems in other industries, it has the potential to not only speed up FDA review of initial labeling and post-approval labeling changes, but to help the agency communicate accurate information easily and rapidly to physicians, pharmacists, and patients.

Caveat Implementor

However, the road to SPL compliance is also filled with potholes that could derail their efforts.

Our experience implementing XML-based systems for companies in the pharmaceutical/life sciences and other industries has uncovered three pitfalls that often plague organizations deploying new content management and publishing systems. On the other hand, we're convinced that organizations that adopt a careful strategic approach to SPL compliance, one that embraces XML and its potential while avoiding these pitfalls, will seize a clear competitive advantage and drive significant improvements throughout their organization.

Inefficient Processes

- ***Using highly skilled people on manually intensive tasks.*** To achieve maximum efficiency, drug makers need to be sure that their new XML system lets them use highly skilled staff and subject matter experts in the areas they are most adept at such as making sure that writers don't waste time troubleshooting issues with document formatting.

Organizational Boundaries

- ***Make sure that departments aren't duplicating efforts across departments.*** For example, there may be overlapping content development requirements between labeling preparers, R&D (which usually initiates work requests), marketing (which needs current information when creating sell sheets), and packaging (which usually needs to distill key information from "Instructions for Use" to meet packaging requirements).

Technology Limitations

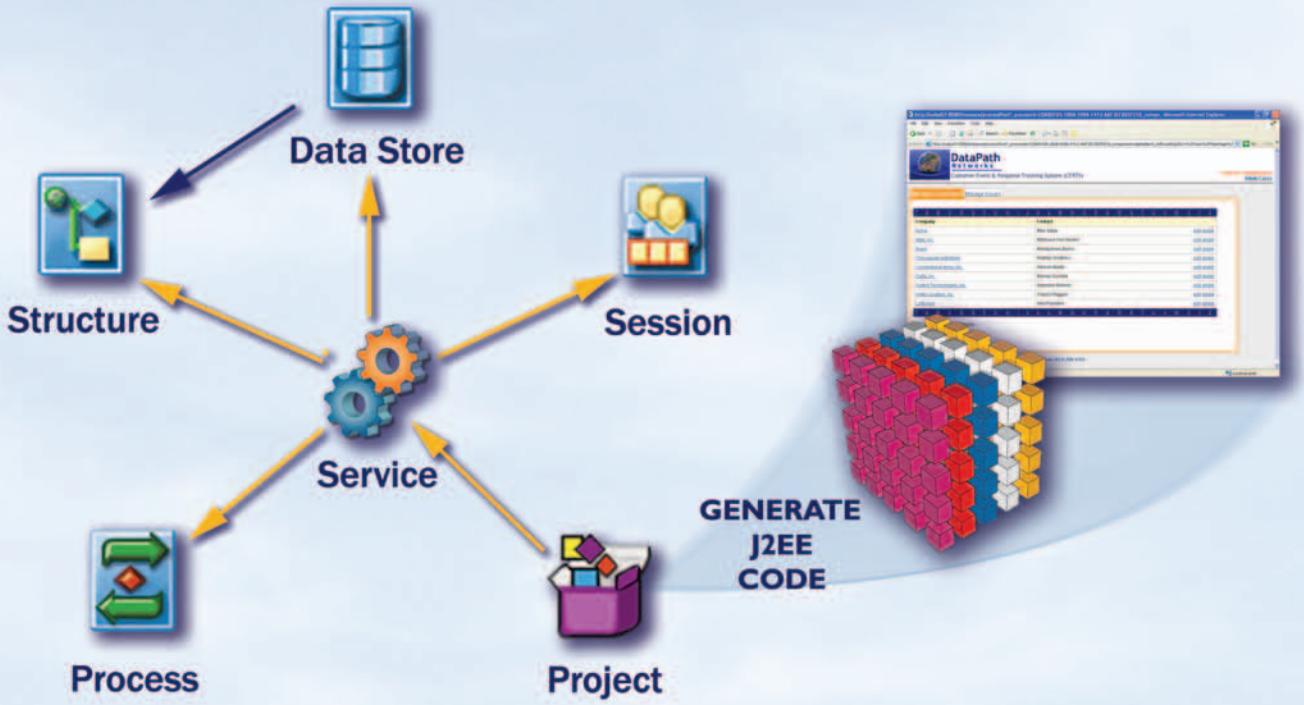
- ***Don't get bogged down by proprietary systems.*** They make it difficult to migrate legacy content, metadata, links, and automated workflows into a new system.
- ***Beware of "standards-based" systems that manage content in proprietary ways.*** A corollary of the previous pitfall; investigate how a content management system manages linking relationships and reuse between content components. Many systems that purport to be XML-compliant actually use proprietary function calls and API "extensions" to XML that constrain the environment and create a closed architecture. In essence, this establishes a "vendor lock-in" and makes it extremely difficult to migrate to other systems in the future.
- ***Don't use tools for something they weren't designed for***, such as using MS-Word for final composition.
- ***Patchwork integration of systems.*** One of the key benefits of the SPL initiative is the use of XML as the foundation. Leverage the power of XML to incorporate a standards-based approach for communicating information between disparate systems.

Separately, each of these traps can severely impair an organization's efforts to build an efficient content supply chain. Collectively, they make it all but impossible. But if pharmaceuticals fail to address them, they will miss many of the benefits that come with converting their documents to SPL.

ddube@innodata-isogen.com

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

Never mind what your mother told you. In your world of multiple platforms and databases, it's not only *okay* to talk to strangers, it's *essential*. That's where z/XML-Host™ comes in. z/XML-Host makes it Easy for your mainframe 3270 applications to talk with any other platform using XML and SOAP-based Web Services. This includes .NET and Java applications--even Microsoft Excel can access your mainframe data Easily. And since z/XML-Host runs directly on your mainframe, you can meet strangers on the same day you complete the Easy installation.

Visit **www.illustro.com/strangers.htm** and learn more. Even download a fully-functional version of the software to try for Free.

Internet-enabling your mainframe with illustro's z/XML-Host? **Now you're talking!**

Works with Microsoft .net Java Microsoft Office

illustro™ SYSTEMS INTERNATIONAL, LLC

See The Light.™

www.illustro.com • Toll-free (866) 4-illustro or (214) 800-8900