SYS-CON
NOW INCORPORATING XML JOURNAL

SOA Web Services JOURNAL

January 2006 Volume 6 Issue 1

SOA Governance Best Practices – Architectural, Organizational, and SDLC Implications
Taking the management of services to the next level
pg. 16

Multicore Systems
Ready or not, here they come
pg. 26

Building Blocks of SOA Governance
Establishing demand and supply centers is a reliable approach for SOA governance
pg. 38

Successfully Planning for SOA

Taking a strategic approach to realizing SOA

PG. 10

RETAILERS PLEASE DISPLAY UNTIL MARCH 31, 2006
$6.99US $7.99CAN
01
71486 03420 9

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

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

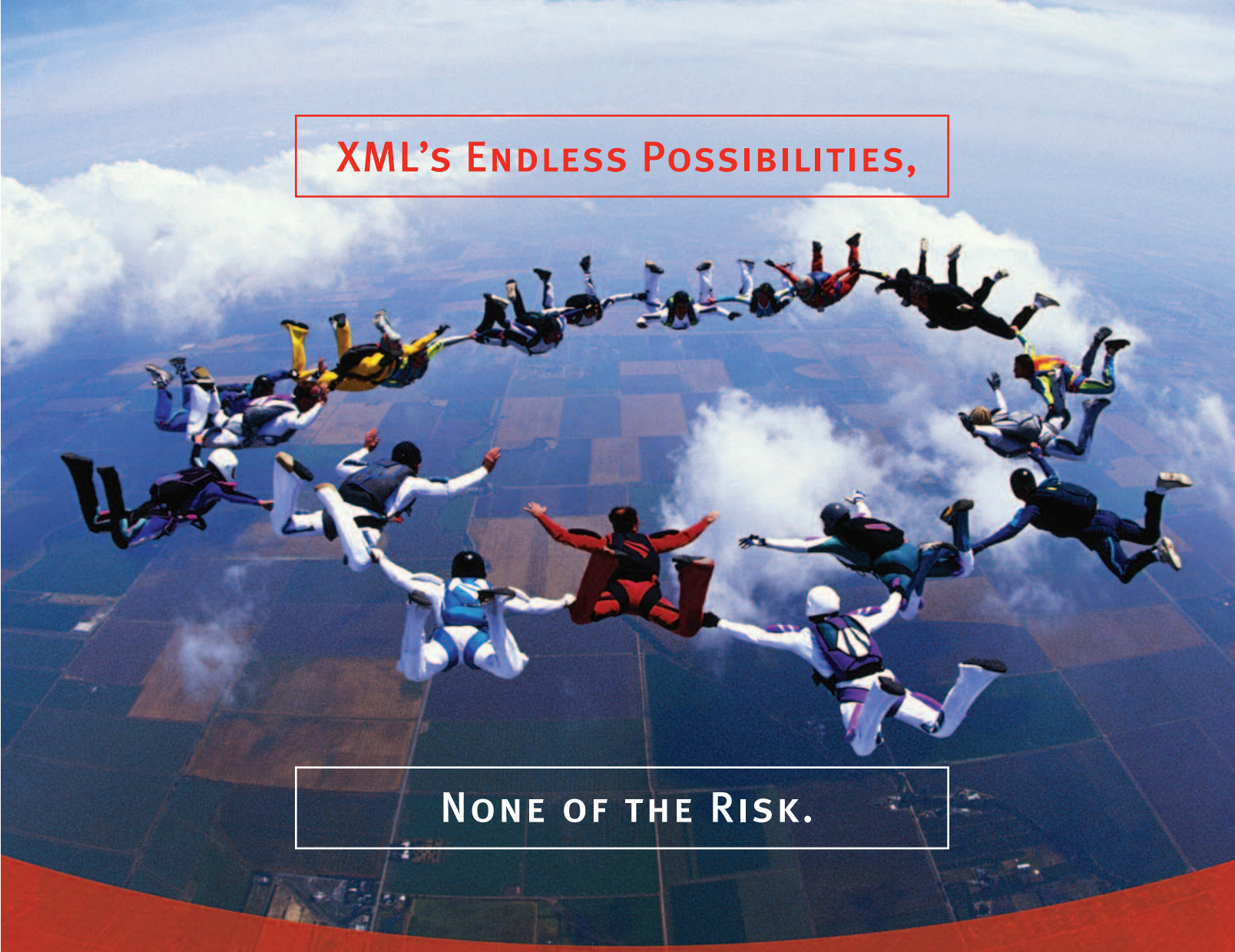
Unless of course, you've discovered the power of Wily. No other software offers our level of insight. Which means you'll be able to monitor web services transactions in real time, find the root cause of problems fast and optimize performance across the entire SOA application environment — end-to-end.

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

Get Wily.™

1 888 GET WILY | wilytech.com

wily technology



Forum XWall™ Web Services Firewall - Reinventing Security

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

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

FORUM SYSTEMS™ – THE LEADER IN WEB SERVICES SECURITY

SOA Web Services Journal

January 2006 Volume 6 Issue 1

XML Journal

Inside WSJ

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

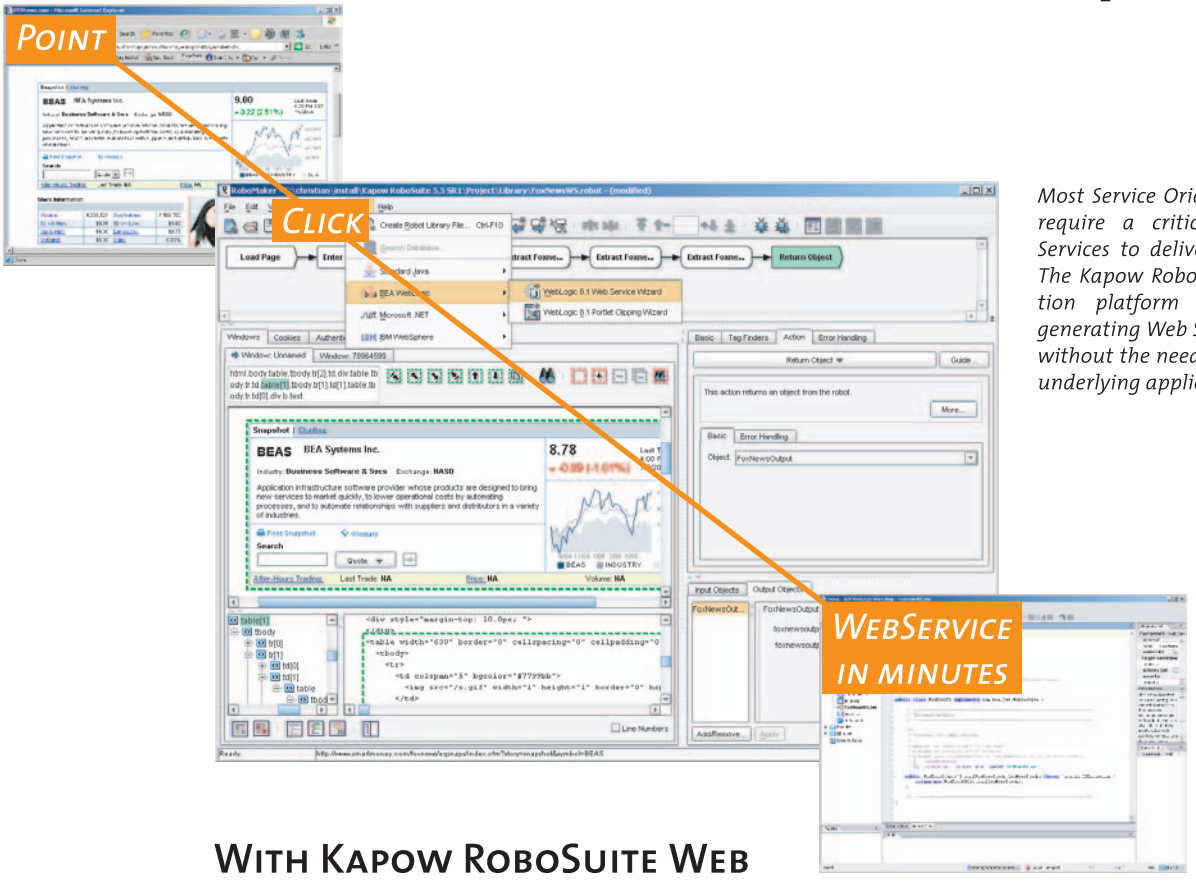
Support *No more finger pointing!*
Web services require support at the API level – a real burden for most organizations. Mindreef SOAPscope makes it easy by allowing customers and support staff to share Mindreef package files that contain complete problem data

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

Leverage existing IT to generate Web Services

[in minutes]

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



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

With Kapow RoboSuite Web Integration Platform you get

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

kapowtech.com

kapow TECHNOLOGIES

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

Who's in Charge Here?

WRITTEN BY
SEAN RHODY

If you're old enough, you probably remember the whole episode with Al Haig in the White House, saying "I'm in charge here" during the period when Reagan was shot. He wasn't really, but it's a good illustration of the concept of confusion, and how different people react to situations. How is this relevant to SOA and Web services? That's an excellent question.

As in a country, an organization implementing SOA needs governance and direction. A simple example may help shine a light on this subject. Let's suppose a division of a large company implements a new security service. Since it's only one division (let's suppose the company has some 50 divisions), they size the service to suit their needs only, and use a small server (maybe two for redundancy). Now, the service is successful, and other divisions start to use it, and so on. Pretty soon the service is up and down like an elevator, and when it is up, the response time is slow.

This illustrates why SOA and Web services need a governance model. Our hypothetical division implements the service for its needs, without considering the bigger picture. However, one of the goals of any SOA is reuse of services and reduction of redundancy, so any service that is successful needs to be designed for reuse. One of the key issues, of course is who pays for it. Why should that single division shoulder the burden of maintenance and equipment for the other 49 divisional needs?

Key concepts in the governance of SOA are cost and control. These two concepts go hand in hand – you can't expect control without expecting to bear the cost of the service. As services become more widespread and usage increases, organizations have to deal with scalability challenges, increased maintenance, and eventually with version-control issues.

Another key challenge is managing the reduction of complexity. Everyone is in favor of reducing the number of different ways to do the same task, as long as you don't touch MY way of doing things. One division of a large corporation surveyed its various applications and identified about 1,000 services. Then they did some comparisons and realized that because of redundancy, they could reduce the service count to fewer than 200. Over 80 percent of their software portfolio was redundant.

When things are that way, they are that way for reasons. Some have to do with specialized systems for special purposes – silos of useful functionality, but limited scope. It's not uncommon for organizations to have three or four or more systems for every real service – in telecommunications for example, it's often the case that they add a new billing system to account for new ways of doing business, rather than wait for changes to the old billing system. These are the problems SOA is designed to solve, but the key issue is governance. You may have heard the expression "You can have it fast, you can have it cheap, or you can have it quickly, but you can't have all three." Somewhere along the line the decision of whether we need it right now, or whether we will stick to a service interface and make changes has to be made. Fortunately, SOA allows us to put a façade in front of any new system to make it conform to the needs of the organization. In this fashion, it allows an organization to add new capabilities quickly, without compromising the overall integrity of the architecture. It also enables reduction of redundancy, without the pain of intermediate states of a system.

In order to achieve this nirvana – this heaven of software and services, there has to be order, and management and governance. The corporate equivalent of Al Haig must stand up, and say "I'm in charge here." However, in order for it to work, he must REALLY be in charge – and stick to his guns long enough to effect change.

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

SOAWebServices JOURNAL

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

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

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

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

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

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

Worldwide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ
Newsstand Distribution Consultant:
Brian J. Gregory / Gregory Associates / W.R.D.S.
732 607-9941 - BJGAssociates@cs.com

For list rental information:
Kevin Collopy: 845 731-2684,
kevin.collopy@edithroman.com;
Frank Cipolla: 845 731-3832,
frank.cipolla@epostdirect.com

SYS-CON MEDIA

Ruling Out Services

WRITTEN BY
AJIT SAGAR

Ask 10 people the question: What is SOA? You will most likely get 10 different answers. Chances are that in more than 50 percent of the cases, the word "Web services" will be a part of the answer. Another 20 percent will talk about process orchestration, XML, integration, and so on. All of these answers definitely describe either the elements of SOA or the components used for the implementation of SOA. One of the technology paradigms that does not instantly come to mind though is "business rules."

The association of business rules technologies with service orientation is rather new, although the technology itself has been around for a while. Business rules and process orchestration make a rather interesting combination. A few years ago the two areas had substantial overlap, and in a project it was sometimes hard to determine which one to use to solve which problem. A couple of years back a client had engaged us in a project that involved a rather elaborate proof of concept to determine whether the combination of process orchestration business rules made sense in their organization. The project was part of a larger initiative to service enable their legacy applications. Since the whole team was new to the technology, often questions arose regarding what to model where. The process orchestration vendors provide the ability to execute conditional logic within the process. Of course, this should be limited to logic that is routing, branching, etc. – not logic that is associated with actual calculations, decisions, and complex computations. The BRMS (Business Rules Management Systems) vendors, on the other hand, allow you to do the opposite – execute flows in the midst of processing complex logic. What can't be achieved (in either product) through the basic constructs can be done by using the programming language and scripting plug-ins that are available.

The market has matured quite a lot since then. It is not that the features have been removed – indeed, the products offer more, not less. It is just that the space that is occupied in the tiers of an SOA is much more clearly defined. Process orchestration is for the orchestration of business logic (implemented in SOA as discrete services). It allows us to separate the actual execution of the service from the context in which it is executed.

The service that is orchestrated through process orchestration performs business logic. If that business logic is composed of complex computations based on decisioning, it lends itself very well to the paradigm of business rules. All of the business rules engine (BRE) vendors in the market specialize in modeling such logic during design and executing it very efficiently during run time. Also, the interface to this logic is accessible as a well-defined service in an SOA. If you are designing a business process, essentially you will end up with discrete tasks. The tasks that lend themselves well to complex conditional logic are excellent candidates for business rules. BREs also provide powerful features for designing and modifying the rules in "business terms," which helps an SOA achieve the business agility objective.

Thus the question that may come to mind is: Why don't we see business rules mentioned whenever there is a reference to SOA – such as orchestration? Well, though business rules can be used to implement services in an SOA, they are applicable to a certain kind of service. In other words, there are other paradigms for creating services – object-oriented programming language constructs constitute one of the most prevalent. On the other hand, process orchestration orchestrates all services, whether they are business related or not.

> "Process orchestration allows us to separate the actual execution of the service from the context in which it is executed"

About the Author

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

ajitsagar@sys-con.com

Other companies in this magazine spent a lot of time on pretty ads. As you can see, we did not. We spent our time hiring the best people and training them to deliver outstanding support for your website. We spent our time building a state of the art datacenter and staffing it with people who care about your website like it's their own. Compassion, respect, credibility, ownership, reliability, "never say no," and exceed expectations are words that describe our service philosophy. From the first time you interact with us, you'll see what a difference it really makes. And you'll also forgive us for not having a pretty ad.

WEB HOSTING • MANAGED DEDICATED SERVERS • COLOCATION • VPS • ECOMMERCE • BLOGGING • EMAIL

Successfully Planning for SOA

Taking a strategic approach to realizing SOA

As you continue to develop your IT architecture, it becomes clear that the route to achieving real business benefits requires a fundamental change in the way you think about system design. In this article on services-oriented architecture (SOA), I'll share with you helpful tips, insights, and a domain model to help you plan this change, and ensure the success of your SOA implementation.

SOA: A New Way of Thinking

WRITTEN BY
DAVID GROVES

Albert Einstein once said, "The significant problems we face cannot be solved at the same level of thinking we were at when we created them." In today's enterprise computing, this suggests that the challenges IT faces in delivering successfully to the business cannot be overcome without changing the way we think about IT. For developers and enterprise architects alike, SOA provides a structure for that change. The question then becomes: How do we migrate to that new level? How do we prepare for such a fundamental change? How do we do so in the most cost-effective, least organizationally traumatic way possible? The answers begin with proper planning.

SOA is not so much a technology as it is a way of thinking. It is bold agenda for infrastructure reformation and represents a culture shift in how we employ technology and work together. Its sudden popularity is less a product of hype than it is recognition of SOA as an evolution towards providing closer alignment between our business and our IT systems. Also, that evolution is stunning and far reaching in its implications for the success of our enterprise.

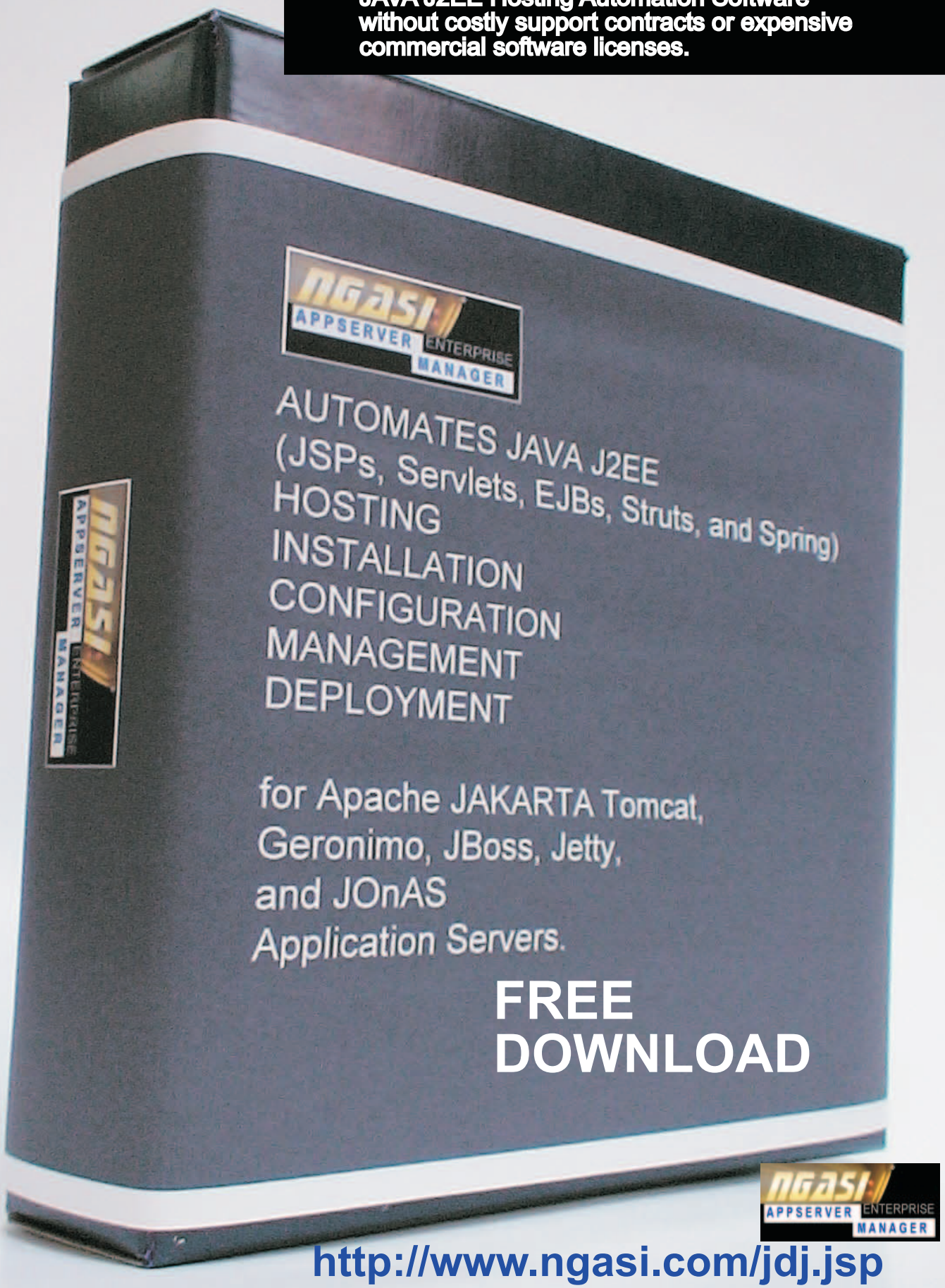
> "SOA maps IT functions to your business processes, thereby enabling business improvement over time"

What Is SOA, Really?

Service-oriented architecture is an IT strategy that organizes the discrete functions contained in enterprise applications into interoperable, standards-based services that can be combined and reused quickly to meet business needs.

A service is a module of code, governed by a service level agreement, which can be accessed via a standards-based interface. Each service represents a piece of functionality that maps explicitly to a step in a business process. Services can either be written from scratch or mined by exposing modules of existing system functionality from previously "silo-ed" applications.

Over time, a catalog of services can be built up, allowing business functionality to be fluidly accessed and reused across many different systems. In this way, SOA can eliminate duplicate data, rekeying of information, and human error at the tactical level while enabling strategic change – for example, by creating a single view of the customer, and in



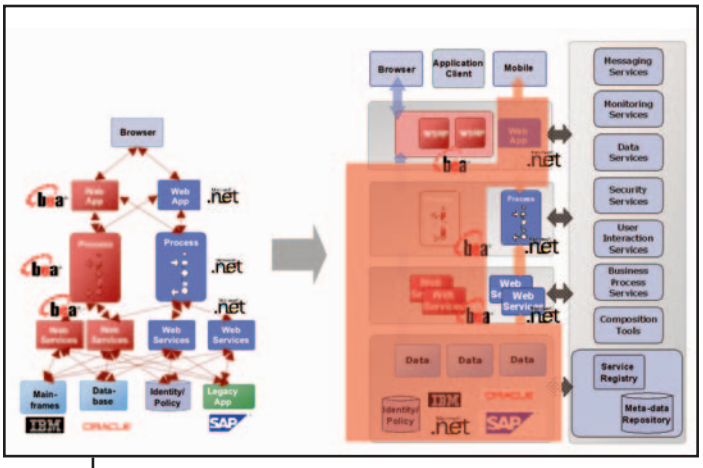


FIGURE 1 | From silo-ed applications with point-to-point connections to SOA on Service Infrastructure

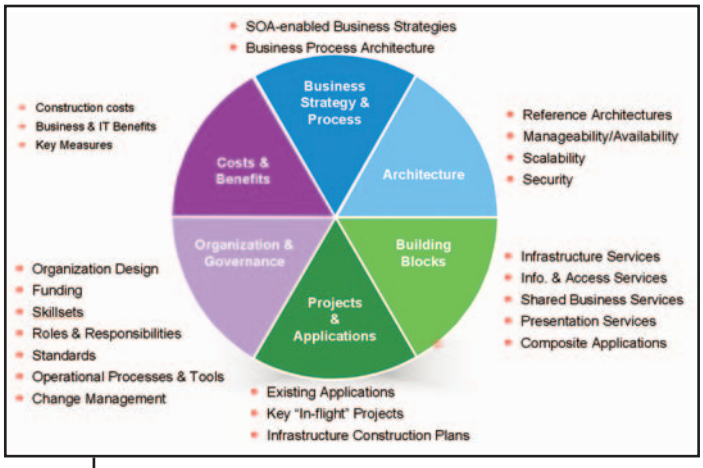


FIGURE 2 | BEA SOA Domain Model for best practice SOA planning and implementation

the process opening up new possibilities for cross-sell, up-sell, and offering services for a more attractive user experience.

New Infrastructure Paradigm

Part of the paradigm shift of SOA is a move from application infrastructure to service infrastructure. Prior to SOA, applications were organized into "silos" with point-to-point connections. SOA uses the same back-end application engines and middleware, but leverages a converged Service Infrastructure Layer (Figure 1).

Implementing SOA

As you begin to implement SOA, use the following formula:

1. ***Think Strategically, Execute Tactically:*** Start by realizing a single-core process with simple, agnostic services that span multiple business units
2. ***Top-down:*** Identify the services required to support this single-core process
3. ***Bottom up:*** Identify functions in existing systems that could be exposed as services to support this process
4. ***Infrastructure Services:*** Identify common supporting functionality requirements
5. ***Expand Slowly:*** Technically challenging projects may be undertaken concurrently after initial projects have proven successful
6. ***Build an Application Catalog:*** On a project-by-project basis, harvest and reuse service modules, thereby reducing your cost curve over time
7. ***Focus on Benefits:*** Phase projects in order of ROI, in stages that include plateaus for consolidation

Using the BEA Domain Model To Plan Effectively

To be successful, SOA requires IT and the business to work together in new ways. As you plan for SOA, you will need to create an effective balance between the technical and nontechnical elements. To this end, BEA has developed a domain model (Figure 2) to help guide you through planning the six key areas that must be given equal consideration to ensure a successful implementation. The first three (Business Strategy and Process, Architecture, and Cost and Benefits) are a good place to begin the planning process.

> "Ensure measurability by setting your baseline at the beginning of your implementation, and avoid the need for backfilling down the road"

Business Strategy and Process: Mapping SOA to Your Business

SOA maps IT functions to your business processes, thereby enabling business improvement over time. This mapping process is best expressed as follows:

- ***Analyze:*** Study processes and identify the supporting functionality required
- ***Develop:*** Harvest functionality from existing IT assets, develop new functionality, and ensure that all services have clear service-level agreements.
- ***Utilize:*** Orchestrate services into processes, measure alignment with strategy, and identify opportunities for optimization

Architecture: Defining Long-term Needs

Establish a reference architecture for your IT organization. This is not a current state diagram, but a long-term vision to build against, which should incorporate a two-to-three year architectural vision of where your business is heading. Take the time to define your architecture's guiding principles and policies, but be wary of making those guidelines an end unto themselves. The flexibility and modularity of the SOA system should take first priority.

Cost and Benefits: Demonstrating Immediate Business Value

SOA is designed to hit the ground running, and it is important to prioritize the development of services in a cost-benefit order, so that your SOA shows ROI from the start. With careful planning, "start-up" costs can largely be absorbed into existing budgets. Over time, the reuse of service modules ensures a lower start-up cost for each new business application. Ensure measurability by setting your baseline at the beginning of your implementation, and avoid the need for backfilling down the road.

Using the SOA Maturity Model

BEA's SOA Maturity Matrix (Figure 3) will help you to monitor your SOA rollout, thus helping you to monitor your progress against the different phases of development. This matrix is divided into three stages: exploring, expanding, and Exploiting. To assess your architecture's maturity level, you may use BEA's Online Self-Assessment Tool (www.bea.com/soa).

> "Take the time to define your architecture's guiding principles and policies, but be wary of making those guidelines an end unto themselves"

Conclusion

It has been my goal in this article to offer you some pointers for organizing successful SOA planning, and my belief is that in taking this approach you can achieve the smoothest SOA rollout possible, and move your organization on to the next level of your development and business agility. For more information on BEA's SOA solutions, please visit www.bea.com.

References

- *BEA Domain Model Whitepaper (PDF):*

About the Author

David Groves is Americas SOA Practice Lead at BEA Systems. His industry experience includes IT strategy, enterprise solutions consulting, program management, and delivery methodology. He has provided consulting and service delivery management to many of BEA's enterprise customers in the Finance, Telco, and Government sectors. David is focused on developing BEA's service offerings and capabilities, with a particular interest in SOA and IT strategy.

david.groves@bea.com

SOA: Separating Myth from Reality

Using the facts to lead the SOA journey

There is still enough hype around service-oriented architecture (SOA) that it's difficult to get a handle on the realities of implementation. While most IT managers now understand what an SOA can do for IT, many still need help figuring out how to get started. New questions about management, reuse, and legacy applications are also emerging as more and more companies think about how to adopt SOA solutions.

The myths and mixed messages surrounding SOA hinder companies from starting on their path to SOA adoption, and in many cases actually prevent companies from realizing that some SOA principles may already be in place. Here are some of the questions that we have been addressing with our customers to help them move from thinking about SOAs to actually creating a plan.

WRITTEN BY
MARK POTTS

What do I need to get started – services, software, hardware?

SOA is not a "rip and replace" strategy; rather, it is a way for enterprises to leverage existing investments, manage them better, and become more flexible and aligned with the real needs of the business. This journey is not facilitated simply by buying a new set of products or adopting new technologies. Realizing the value of SOA requires expertise and governance from the formal design process through implementation, provisioning, operation, and change – i.e., the entire life cycle. The vital starting point, therefore, is not technology, but rather identifying the business-specific services in an organization.

Due to the different focus – business-driven tops-down rather than technical bottoms-up – many customers require SOA expertise in the beginning of the process in order to map out a strategy that makes sense for the particular company and its goals. This is why so many large vendors like HP are offering services specific to SOA as a part of their offerings. The appropriate management tools that link the development life cycle to the operational life cycle become essential as an organization embarks on SOA-based projects.

How would implementing an SOA benefit my company?

A key benefit of adapting an SOA approach is the alignment of IT investments with a company's business strategy. As a result, companies can preserve legacy systems by exposing them as business assets without the need for replacement or significant modifications. SOAs also facilitate a company's ability to develop new business capabilities at lower cost, thereby meeting the needs of the business in the required time frames when change is driven by required change or new opportunities.

How difficult are SOAs to manage?

Because of the loosely coupled nature of SOA, an SOA implementation does bring additional management concerns to the forefront. Among them: managing the service integrations as well as the component parts; managing security across organizational or implementation boundaries; and managing policies for deployment, execution, and life cycles. Management is critical to realizing the business benefits of SOA and addresses the nonfunctional aspects of SOA, which are where the majority of the costs lie (security, configuration, financial, and QoS). Managing the nonfunctional aspects of SOA also extends to issues such as compliance and IT governance, which are also critical as SOA is adopted more widely across IT and reuse and integration cross organizational boundaries.

If I am already using Web services, do I need to think about SOA?

There is a relationship between Web services and SOA, but the use of Web services does not constitute an SOA or deliver its benefits. SOA is an architectural approach, while Web services are an implementation choice for adopting an SOA, in which interfaces are based on standardized Internet protocols. The

> "While the complexity of SOA can raise questions, it's worth the time to investigate how this architectural solution can bring new power to your enterprise"

service-oriented approach does not limit interoperability to Web services – it can be implemented using any kind of service-based technology. In that way, full interoperability is only possible through an SOA, and it's the only way to fully realize the benefits of implementation.

Am I going to be able to justify this adoption from a business perspective?

The key to justifying SOA adoption is to realize that the business justification is not tied to the architecture, but rather to the ability to serve customers faster, address compliance requirements efficiently, and to become a more agile in supporting the business objectives and goals using IT. One starting point is to evaluate which business processes generate the most return on investment for your company and prioritize your SOA deployment to make these processes more efficient and more customer-aware. Also, because SOAs are not built upon a "rip and replace" strategy, investment is not geared toward buying more and more technologies, but instead leveraging existing assets and investments for better leverage, standardization, and governance across the business.

Now that I'm planning for SOA, can I incorporate legacy applications into the plan?

Because of the cost and inflexibility associated with legacy systems, they usually need to be "modernized" to be incorporated into an SOA. An evaluation will be needed to determine if the code should be refactored, modularized, or if the application could be replaced with a commercial application. Additional design work will then need to follow to build out a modernized application that is agile enough to integrate with other applications within the SOA.

Can SOAs solve all of my integration problems?

No. Ultimately, it is the information integration strategy underlying an SOA that solves integration problems. The SOA itself is not the true solution; rather, it's a better way to achieve a desired goal. Integration problems are solved through a number of actions related to adopting an SOA: carefully planning for an SOA, assessing business requirements, establishing governance, and enacting an underlying information integration strategy. It is only when these steps have been taken that an enterprise can expect to have its integration problems solved.

Are all SOA solution components completely reusable?

One of the major benefits of the set of architectural principles that define an SOA drives organizations towards a greater level of reuse and consistency. The amount of achievable reuse increases over time once newer services are designed using existing services in an organization. Determining which services are the most appropriate for reuse is where having insight into which business processes align the business goals most closely with IT is crucial.

In the end, IT professionals need to be armed with the most accurate information available to determine how to begin the SOA journey. While the complexity of SOA can raise questions, it's worth the time to investigate how this architectural solution can bring new power to your enterprise.

About the Author

Mark Potts is a fellow and chief technology officer within HP's management software business. Prior to joining HP, Mark was the founder and chief technology office of Talking Blocks, which HP acquired in September 2003. Talking Blocks was a software company that provided products for managing SOA and Web services.

mark.potts@hp.com

SOA Governance Best Practices – Architectural, Organizational, and SDLC Implications

Taking the management of services to the next level

The fact that you're reading this article means that you are probably planning a service-oriented architecture (SOA) initiative and recognize that some level of governance is required in order to be successful. If you are like most people in this position, you are also somewhat confused as to the meaning of SOA governance. Governance is the current buzzword, and combining governance with SOA creates a phrase that every independent software vendor (ISV) wants a piece of. How do you sort out what is marketing hype from what is truly valuable and relevant to your organization's SOA efforts?

WRITTEN BY

BRENT CARLSON

ERIC MARKS

Governance Scope Within an IT Organization

Much of the hype around SOA governance has been focused on *operational* governance. Defining, tracking, and managing factors like service-level agreements (e.g., average response time, peak response time, average throughput, peak throughput) and authorization policies (e.g., users from organization A are allowed to invoke this service while users from organization B aren't) are clearly important once the pieces of an SOA get up and running within an organization's IT infrastructure.

However, while operational governance and management is necessary for a successful SOA initiative, it is not sufficient. For an organization to effectively define and implement an SOA (and not simply implement a series of point-to-point services masquerading as an SOA, but in fact creating another layer of technology spaghetti), it must extend SOA governance back to the *development* and *architectural* perspectives. To be successful with SOA, you must find a way to bind these perspectives together as seamlessly as possible to enable effective information flow in both directions: from architecture to development to operations, and vice versa. Let's investigate each of these governance perspectives in turn.

Architectural Governance

Architectural governance at the enterprise architecture (EA) level involves three key elements: 1) making core decisions about business or technological functionality within the enterprise, 2) sufficiently documenting those decisions so that downstream consumers (the teams responsible for developing and deploying services and applications) can quickly understand and make effective use of those decisions, and 3) reviewing the project-specific application of those decisions. In order for an EA team to execute these tasks, it must have at its disposal an effective way to disseminate the knowledge assets it produces, to track and understand which knowledge assets are being applied to specific projects, and to document the review of those project-specific decisions.

Design-Time (Development) Governance

In many ways, Software Development Life Cycle (SDLC) governance within an SOA initia-

more relevant results than Google's appliance.

context searches mean better searches than Google's appliance.

your best choice for the most efficient searching of XML.

find your element at **whatthexml.com**.

element

tive is a reflection of decisions made at the EA level. Decisions about the scope and granularity of business services to be implemented and the technical approach to be used in implementing those services must be applied to specific service production or consumption (i.e., application development) projects. However, SDLC governance extends beyond appropriate application of EA guidance to the actual analysis, design, implementation, and testing of the resulting services and/or applications required by the IT project at hand. With respect to service production, SDLC governance involves the progressive "hardening" of the service as it progresses through its requirements definition, design, implementation/unit test, and integration/system test phases to eventual deployment in the operational environment. When applied to service consumption, governance may involve both internal project-specific reviews (e.g., have the appropriate services been selected, have requirements for new services been identified) and external reviews from the perspective of service providers (e.g., does the use of this service within this application conform to enterprise-specific or government-mandated privacy rules, does the service implementation contain open source components and if so, are the components used in a manner such that enterprise-specific intellectual property is not compromised).

Operational Governance/Management

Operational governance/management within an SOA involves applying appropriate business and technical policies (e.g., which groups and users are allowed to invoke a particular service, what are the minimum throughput and response time expectations required of a service) to deployed services. Business policies are often implemented within an SOA by an Enterprise Service Bus or SOA Fabric integrated with the enterprise's authentication and authorization infrastructure, while technical policies are typically monitored by a services management platform. The cumulative set of governed technical policies is often referred to as a service-level agreement (SLA). Examples of SLA-level technical governance elements within an SOA are:

- Average throughput
- Peak throughput
- Type and description of committed SLA
- Availability
- Consuming service clients
- Hardware and software configuration
- Fault history
- Alert thresholds

> "SOA, in conjunction with other loosely coupled architectural approaches, forces IT organizations to recognize that teams producing services are not likely to be the consumers of those services"

Political/Organizational Aspects of SOA Governance

How do we map these governance disciplines into an organization's structure and roles? Because of the loosely coupled nature of SOA, SOA governance is a new discipline that has implications for existing corporate and IT institutions as well as for new organizational structures and processes (and the politics associated with those structures and processes). Proper focus on what governance is, how it can be achieved, and its implementation can help make governance a valuable and necessary function to support your SOA migration.

SOA governance has an impact on current IT governance processes. Some of these processes include the budgeting and project approval process, portfolio management activities, and ongoing oversight of projects to assure budgetary compliance. Applying governance to SOA activities is critical because there may have to be changes to the normal IT governance processes for budgeting and portfolio management.

Think about the budgeting process of your organization. That budgeting process has a tremendous impact on the behavior of various organizations and their IT representatives. If there is no budgetary control of projects to influence them to adopt SOA and reusable services as their fundamental design concepts, then projects will go their own way as driven by the requirements of that particular business unit or project. The same goes for the portfolio management process. If there is no mechanism to surface SOA and reuse opportunities for all projects and then apply budgetary pressure to converge them toward an SOA, then they will similarly go their own way. SOA governance, budgeting, and portfolio management are ways to influence behavior of business units, as well as the IT and business personnel within them, to more aggressively support SOA and reuse.

Enterprise architecture processes may undergo similar changes given the advent of an SOA initiative in an organization. Often the architecture process and organization will have to be restructured to accommodate the requirements of an SOA initiative because the skills, roles, and functions of an EA team are not completely appropriate for an SOA initiative. Think about the process of architecture as two tiers of activities: one tier is the architecture strategy and goals, followed by the definition of the elements, standards, and organization of architecture to accomplish those goals. The second tier is the application of architecture to funded projects, the acquisition or implementation of various technologies and standards, and the enforcement of compliance to the enterprise architecture goals.

These are two related yet distinct processes, and often they are not as interdependent as CIOs would like. Think about the cases where there is a chief architect or central architecture group at corporate headquarters,

and then also present are the solution architects assigned to projects. They actually build systems and implement technologies and standards. Who has the most direct bearing on the architecture that ultimately is implemented in a given organization? Naturally it is the person assigned to the budgeted project that was sponsored by a specific business unit that ultimately funded the project. The behavior associated with enterprise architecture is similarly related to the organization and processes used to achieve the goals of SOA, architecture compliance, portfolio management, and budgetary compliance.

Influencing SOA Behavior

Determining and shaping the behavior patterns that will sustain an SOA effort through time is often the work of change management specialists. However, it is easier to conceptualize the role of behavior in SOA if we examine the intersection of major influences on organizational and individual behavior. Some of these influences are the following:

- Corporate culture
- Major decision-making processes
- Budgeting processes
- Incentives and penalty structures
- Compensation linkages to corporate goals and mantras
- Portfolio management processes
- Architecture process (definition, acquisition, implementation)
- Architecture practice (solutions development)
- Corporate performance metrics, such as return on invested capital (ROIC), revenue and market share growth, cost controls, etc.
- Promotion and advancement criteria

All of these factors influence behavior of organizations and individuals within an enterprise. In the context of an SOA, however, there are a few key factors that determine how effective an SOA can be without fundamental changes to the organization and processes of an organization. To understand these essential factors, we must locate the "center of gravity" of an SOA. Understanding certain key organizational and process relationships will help with the organization of SOA governance, the design of SOA governance processes, as well as understanding the behavior and incentive models that may be required in order to implement SOA governance. For example, the relationship of the budgeting process to project execution and implementation helps determine how effectively budgeting oversight impacts the resulting architecture of IT systems. Without proper budget-to-project alignment, teams will be inclined to build project-centric services that don't necessarily fit the broader needs of the organization's SOA. The relationship of the acquisition/procurement process to enterprise architecture is another key relationship that has a tremendous impact on the resulting architecture of an organization. Does the acquisition process reflect the goals and standards of the EA organization? If not, how can it be changed to better reflect it?

> "SOA governance has an impact on current IT governance processes"

Another critical relationship is the relationship of EA to project or solution architecture – in other words, connecting architecture via governance to downstream activities (more on this in the next section of this article). If there is a disconnection between enterprise architecture and the architecture that is designed at the project level, then there is the possibility of a disjointed architecture.

These important relationships all point to how sociopolitical forces and organizational forces converge to either facilitate or hinder SOA governance. There are no easy answers to these challenges. However, understanding how these organizational tensions either help or hinder SOA governance will point to a path to implementing appropriate organizational institutions and processes to achieve SOA governance objectives.

Production/Distribution/ Consumption: Separation of Concerns Within SOA Governance

SOA, in conjunction with other loosely coupled architectural approaches, forces IT organizations to recognize that teams producing services are not likely to be the consumers of those services. Unlike traditional application development, SOA is built upon the premise that a set of services can be employed within a wide range of applications. In other words, SOAs depend upon the separation of the production and consumption concerns within the IT organization, and a distribution vehicle that allows service producers and consumers to communicate and collaborate with each other. Let's define these production, distribution, and consumption concerns with a bit more detail:

- ***Production:*** Identification of and governance over the development and maintenance of existing and newly defined candidate reusable services
- ***Distribution:*** Publication of those services for widespread dissemination to potential service consumers
- ***Consumption:*** Discovery of and governance over the appropriate use of services within application development projects

While it may be technically feasible to execute the responsibilities within these concerns through manually defined and managed procedures, the reality is that for IT organizations of any size to build out a successful SOA, both political/organizational issues such as those discussed in the previous section, and appropriate tooling such as a services repository/registry that automates both service governance and service distribution are essential. The following sections of this article highlight how a services repository/registry can be employed to support key best practices within the production/distribution/consumption life cycle inherent within SOA.

Production Best Practice: Pragmatic Service Definition

Services within an SOA cannot be developed in a "bottom-up," ad hoc manner. Bot-

tom-up development of services is inherently driven by immediate project needs – how do I solve this specific problem with a specific implementation (often driven by the influence of existing applications and their behaviors masquerading as true business requirements). What happens when an organization defines and implements its services with this mindset? The service layer simply becomes YALOT (yet another layer of technology) – more spaghetti code of a different form that didn't improve our business process flexibility, but simply implemented a monolithic application in a different technology.

> "If there is a disconnect between enterprise architecture and the architecture that is designed at the project level, then there is the possibility of a disjointed architecture"

However services also cannot be defined solely in a "top-down" manner. Top-down business process analysis left to its own devices leads to either "analysis paralysis" – continual refinement of a model hoping to reach perfection (which never comes), *or* "Big-Bang" projects – trying to define and implement everything at once, usually with disastrous consequences (most typically a combination of "death march" projects and cost and schedule overruns).

Ultimately, what organizations need to make progress in SOA is to develop a coarse-grained business model driven by key business processes (not all of them, but only a representative set of high-priority processes to begin with). Architects and business analysts should collaborate to build this model using those processes to extract and define a normalized set of functions, then grouping those functions together based on behavioral affinity (read components and interfaces for those of you who are UML centric) as a strawman set of initial target service definitions.

At this point, we have a useful framework for the "real work" – detailed analysis, design, and implementation of the services we need for our current set of prioritized projects. Based on business process (and project) prioritization, we identify the needed services from our business reference model and formalize the service definition for these prioritized services. Ideally, each service should be driven by the requirements extracted from at least two separate processes – designing a service based on a single use case is very likely to result in a fragile and narrowly defined service that will not be flexible enough to meet our next set of prioritized projects. Our formalization efforts are likely to result in modifications to our business architecture – which is just fine! We can iteratively enhance our architecture as we make progress towards service implementation.

Production Best Practice: Recommended Service SDLC Governance/ Review Checkpoints

Now that we have our first set of services defined, we need to build and deliver them. Developing services within an SOA (i.e., for purposes of reuse across multiple applications) usually requires more of the production team than a single-use component, module, or object. In order for a service to be considered reusable, it must be maintainable, discoverable, and consumable. *Maintainability* introduces such concepts as version control (which we will address in more detail later in this article), models and other design documentation, and requirements traceability (why was the asset implemented in this way from a technical and business perspective). *Discoverability* forces us to consider how we help potential consumers of this asset find the asset in a timely fashion – via keywords, domain taxonomies, and mapping to models, for example. *Consumability* involves looking at the asset from the point of view of the downstream project planning to use the asset: Is there a user guide, a well-documented API, sample client code, and other artifact available to help the user rapidly understand how to apply this asset to the project at hand? Are dependencies on other assets (and to prior versions of this asset) specified and easily navigated?

How do we achieve these high standards for our services under development? By establishing standardized governance/review checkpoints throughout the service SDLC. We recommend that at a minimum, organizations should review services under development at these points in the SDLC:

- ***Requirements Complete:*** All business requirements documented and initial service definition specified (ideally as WSDL), allowing reviewers to validate the service against its business architectural context
- ***Design Complete:*** Implementation approach defined with sufficient documentation (e.g., UML design models completed, relevant legacy APIs identified) to allow reviewers to validate design against technical and application/integration architectural contexts
- ***Implementation Complete:*** Service implemented and deployed in a test environment, with sufficient supporting documentation (e.g., sample client code, automated/manual test cases and test results, usage guide) to enable a potential consumer to understand the service and to trust its quality and stability

Other review points may also be appropriate based on organizational needs and objectives. However, don't overwhelm your development teams with process for the sake of process: you will quickly instill a revolt of the masses if you force seemingly arbitrary hoops for developers to jump through in the process of completing their work. Your objective should be "just enough process" – don't overwhelm your project teams with unneces-

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

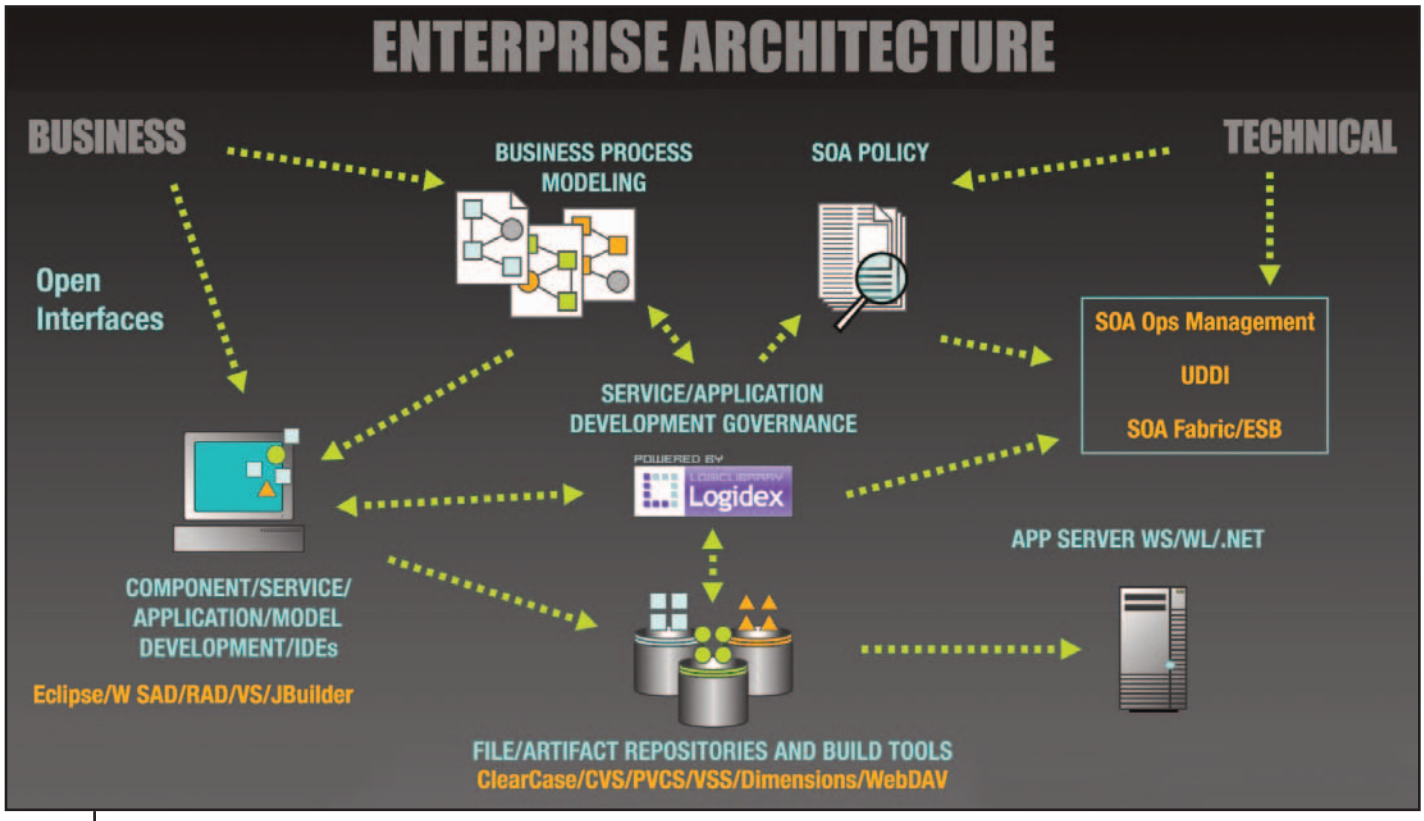


FIGURE 1 | **SOA process and procedures – SOA playbook**

sary workload, but rather provide enough guidance at key points in the production and consumption life cycles to make sure things stay on track. You are very likely going to have to iteratively reach the right level of process for your organization – start with as lightweight a process as you think will work, and then add process steps only as you find need for them. A well-designed services registry/repository can assist in automating these governance processes, thereby reducing the "organizational friction" that could otherwise hinder people from "doing the right thing."

Production Best Practice: Versioned Services Governance

Because services (like components) are meant to be used in more than one application, organizations need to plan for the incremental enhancement of their services over a long deployment lifetime. In effect, organizations planning to build a robust, stable, and extensible SOA need to treat their services as "products."

What does treating a service as a "product" mean to our IT organization?

1. Each produced service must have a regular and well-defined release cycle. This release cycle needs to occur often enough to meet consumer needs on a timely basis, but not so often as to churn existing consumers. Typically a release cycle of somewhere between three and six months is appropriate for most organizations, and allows them to meet new service needs without unduly disrupting existing applications. As multiple versions of a service are released, consider defining these life-cycle states for your services:
   - *Under Development*: Available for requirements gathering and application development team planning purposes
   - *Production:* Mainline version for use in new development
   - *Retired:* Still in use by existing applications but not allowed for use by new apps
   - *Obsolete:* All applications should be migrated off this version; version metadata is maintained for traceability/audit purposes only
2. Services must preserve backward compatibility wherever possible. Deprecation techniques (where obsolete operations are identified as such and notice is given to consumers that those operations will be removed from service interfaces in future releases) give existing consumers time to migrate to newer service releases. Service providers should provide n-1 version support at a minimum – all services provided in the prior version (except those marked as deprecated) should be preserved intact in the current version. In addition, consider providing a "grace period" where both service versions are deployed to allow consumers to make any necessary changes to integrate the new service version. Dynamic run-time binding techniques via Web services management infrastructure (e.g., service proxies or UDDI-based late binding) can also simplify the migration process from old to new service version.

3. Mechanisms for gathering requirements from current and potential "customers" need to be established by the enterprise architecture and service review teams. Consider establishing a "product manager" role within these organizations, one that manages the aggregate set of business requirements for the service and works to prioritize requirements with its current and potential consumers.

Again, a well-designed services repository/registry can help organizations manage service versions over their lifetimes, with automated notifications, embedded discussion forums for requirements gathering and analysis, and filter-based search capabilities that expose services to potential consumers based on service state (e.g., new application project developers should not be able to search for "Retired" or "Obsolete" services).

Distribution Best Practice: Service Distribution via Services Repository/Registry

Now that we have a set of broadly reusable services produced through our application of the aforementioned best practices, how are we going to get them into the hands of our application developers? This takes us back to the discoverability and consumability aspects of services production. Simply put, unless your services are all as simple as the ubiquitous Stock Quote example so often used in articles discussing Web services, WSDL is not enough. Syntactic definition does not equate to semantic understanding. Potential service consumers need ready access to supporting artifacts (e.g., usage guide, sample client code) to make the service consumable to them. The service also needs to be discoverable – wrapping the service with metadata that allows the user to search for useful services using varying techniques and user interfaces. \A well-designed services repository/registry goes a long way to helping IT organizations to efficiently deliver services to potential consumers. At a minimum, such a repository/registry should support both browser-based access and deep IDE integration to enable users with varying roles to discover the right services. For example, a business architect will likely feel most comfortable using domain terminology searches within a browser, while a designer or developer would prefer a UML-based visual search mechanism within their preferred IDE.

> "Syntactic definition does not equate to semantic understanding"

Consumption Best Practice: Service Usage Registration and Traceability by Application Development Projects

The third leg of the SOA governance stool – consumption – comes into play as we begin to build, deploy, and maintain applications based on our previously produced and distributed services. Application-based tracking of service consumption is essential for a number of reasons: to support internally defined and externally imposed business governance mandates, to simplify the process of ongoing impact analysis and change management as the SOA matures, and to provide a quantitative ROI based on real service usage statistics back to the C-level within the enterprise.

Let's take a quick look at governance mandates. Business-level governance (through the form of government regulations such as HIPAA, SOX, and Basel III) is increasingly making its way down to IT. As a result, increasing numbers of auditability and traceability requirements are being applied to the IT organization, and these requirements cannot be met without some form of service usage registration mechanism (and again, for IT organizations of any size, this registration mechanism needs to be automated through the services repository/registry).

Remember also that our services will change over time as new requirements are identified (as discussed above in the "Production Best Practice: Versioned Services Governance" section). Existing application teams need to be kept abreast of planned and implemented changes to the services they are using, both to participate in requirements feedback and to prepare for the eventual obsolescence of back-level services as new service versions are deployed.

Finally, since enterprises are not in business to serve IT but rather it's the other way around, our organization's C-level executives are certainly going to expect a quantifiable ROI from any SOA initiative. Without direct traceability over service usage, it becomes arduous at best and impossible at worst to assemble such a quantifiable ROI based on service use and reuse. On the other hand, if usage registration is built right into the services repository/registry, quantifiable ROI is as simple as running a periodic report.

Summary

Don't think managing your services operationally is enough. Just because you can keep tabs on a service's execution doesn't ensure that the service is really supporting the overall business goals of the SOA. Traceability back to the business goals/priorities through EA to SDLC to operations will make SOA successful in the enterprise. Also, don't minimize organizational impacts that may be needed – monolithic, project-centric funding models are not likely to work in the loosely coupled world of SOA.

About the Author

Brent Carlson is vice president of technology and cofounder of LogicLibrary, a provider of software development asset (SDA) management tools. He is the coauthor of two books: San Francisco Design Patterns: Blueprints for Business Software (with James Carey and Tim Graser) and Framework Process Patterns: Lessons Learned Developing Application Frameworks (with James Carey). He also holds 16 software patents, with eight more currently under evaluation.

brent.carlson@logiclibrary.com

Eric Marks is founder, president, and CEO of AgilePath Corporation, a service-oriented architecture (SOA) and Web services consulting firm based in Newburyport, MA. Marks is a software and technology veteran with 18 years of experience with firms including PricewaterhouseCoopers, Cambridge Technology Partners, Novell, Electronic Data Systems, StreamServe, Ontos, and Square D/Schneider Electric.

emarks@agile-path.com

Are You SOA New School, or Old?

Why you need to know the origins of your technology

It has come to my attention that there are really two kinds of SOA technology vendors out there, old school and new school – each offering very different approaches to solving the SOA problem. I'm not going to mention any particular vendors, but you guys can guess who they are.

WRITTEN BY
DAVID LINTHICUM

Keep in mind, what's important here is that not any particular approach or technology is correct, but that the approaches and technologies you employ match up with your requirements and business expectations. However, it's also important to understand exactly the type of technology you're going to leverage, including its core – that's the point I'm attempting to make here.

Old-School SOA Vendors

Old-school SOA vendors are those with "legacy" integration or application development solutions that "SOA-tized" their stuff, basically adding Web services interfaces, orchestration, governance, etc. I've been here before – you're working with an existing product and existing customers, and suddenly a new wave comes along. You have to keep up, thus you add another layer onto your existing product to make it a bit more acceptable to the market, and also maintain backward compatibility with your existing customer base. It's not easy.

Thus, these guys layer on top of the existing stuff or creating another layer of abstraction, hiding the core with new interfaces, management layers, and perhaps a repository. Some of the common characteristics of old-school SOA are:

- ***The need for their underlying and legacy core technology in order for their SOA technology to function.*** This is a sure sign that there are dependencies on the older coretech. This is not bad unto itself, but could cause performance issues as you work down to the core from the abstraction and back again. You'll find that some of the existing vendors do this, but some do not, so make sure to try before you buy.
- ***The product cannot deal with coupling, only cohesion.*** This means that the SOA solution relies on an asynchronous messing system for internal communications – in other words information-oriented integration – and can't deal with behavior or service-oriented integration. Thus, reusing service through this type of infrastructure can be problematic, if not impossible.
- ***The need for backward compatibility with an existing base of users.*** Again, this is not a bad thing necessarily, but if your SOA product is trying to maintain a portion of itself as a proprietary subsystem, that could limit development going forward.

The upside to old school is that you're in a new market space with a minimal amount of redevelopment. That is the upside for the vendor, however. Moreover, these are typically well-known and established players.

The downside is that you could have an archaic product at the core, and it may not handle the new ways in which we're aligning systems today, at the service level and not at the information level alone.

New-School SOA Vendors

New-school SOA vendors are those with new products. Typically they are start-ups with fewer than 100 or so customers. They don't need to SOA-tize their products because they are typically built from the ground up with SOA in mind, doing both service- and information-based integration. However, there are advantages to starting with a new canvas, and some drawbacks as well.

The upside to new school is that you're getting a product that's built specifically for the application to an SOA. Moreover, these new guys typically solve a particular portion of the problem, such providing just the registry, development environment, or federated identity. They don't offer "all-in-one SOA solutions," which is good. To date, I've not seen a product that can do it all. The downside is that you're dealing with a new company that is going to change a lot in the next few years, in good ways and bad ways.

Conclusion

So, which school should graduate from? At the end of the day, it's really a matter of your problem domain and the problems you're looking to solve, that's for certain. However, I can say that the new school products have more applicability for the SOA space since they are specifically designed for that purpose. The old school products just seem like quick reinventions, many of which are limited by the existing core product. When building an SOA, you don't want limitations – it's hard enough as is.

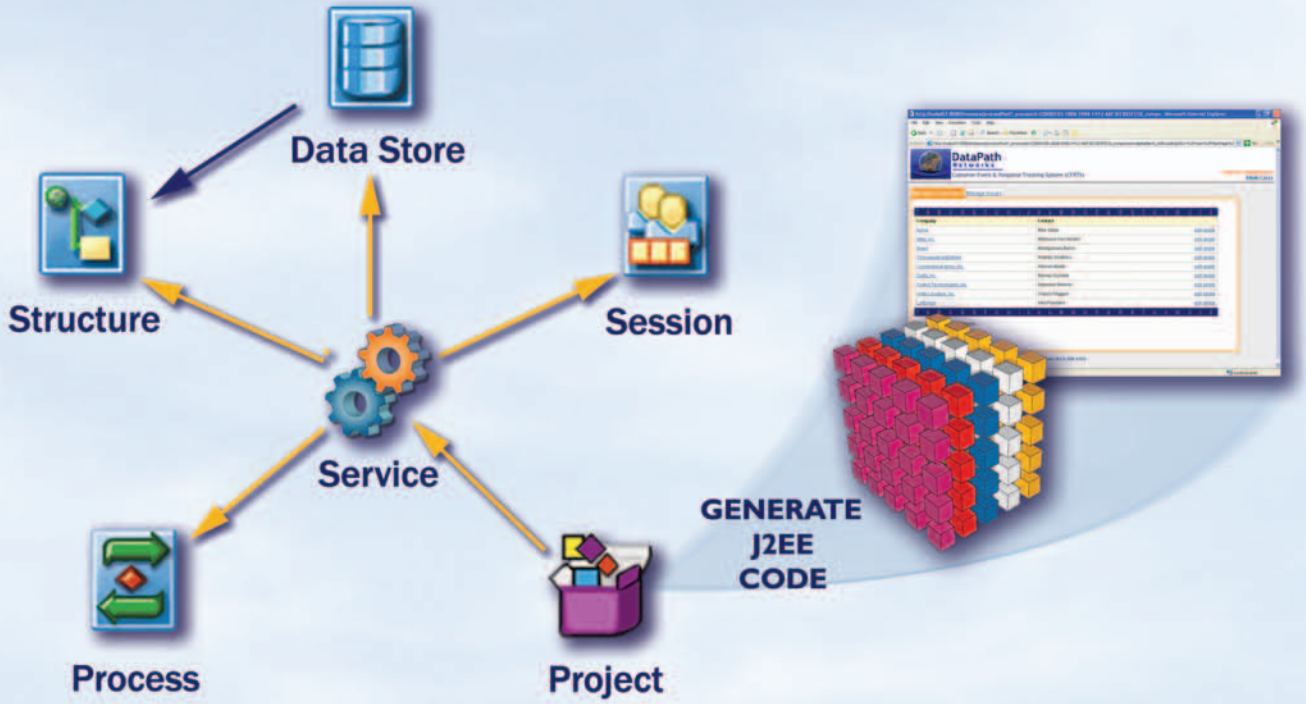
About the Author

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

linthicum@att.net

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

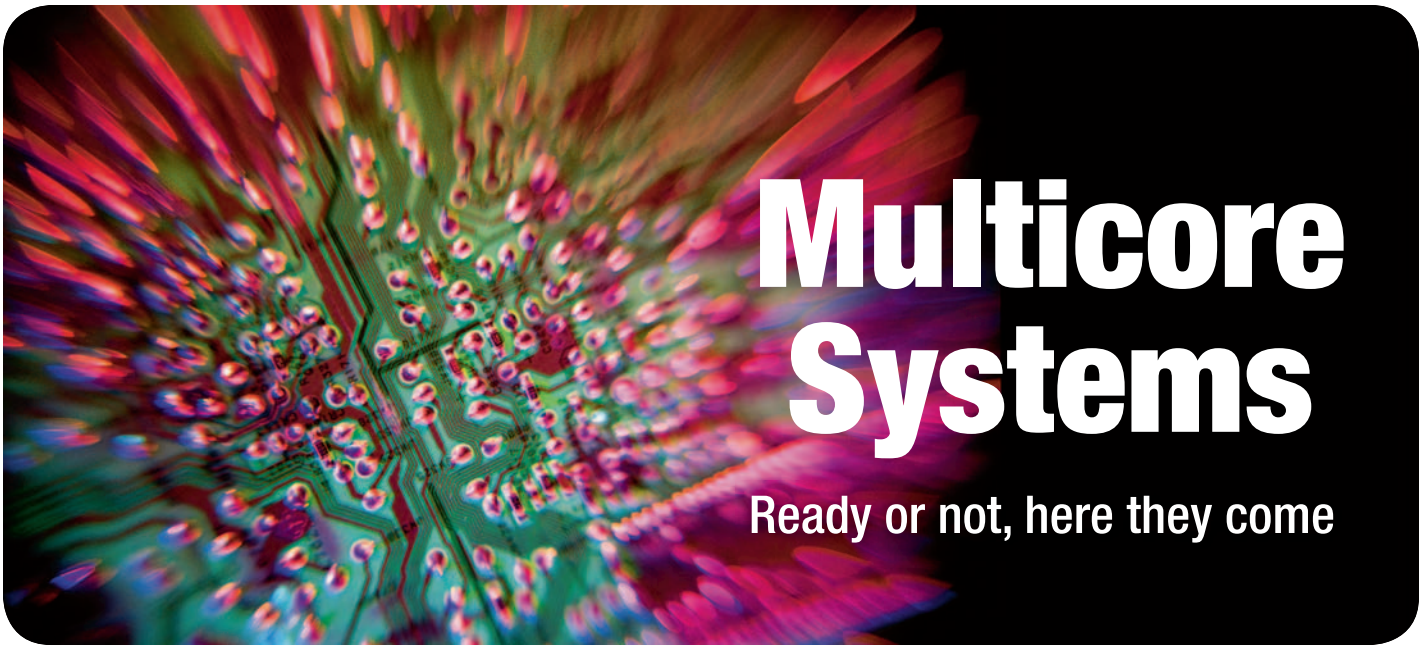
"Developing with Skyway Software gives us a much more rapid response time to our business community, it has a lower TCO, and it accelerates SOA and Web Service adoption throughout BAT."
— Kevin Poulter, British American Tobacco, a $30 billion powerhouse

White Paper — Case Study — FREE Evaluation Download

www.skywaysoftware.com

Skyway software

(813) 288-9355



Multicore Systems

Ready or not, here they come

Every major chip manufacturer has delivered or announced a roadmap for multicore chips that have multiple CPUs on the same piece of silicon. Systems developers are now designing these chips into their entire product line. For Java platform developers, Symmetric Multiprocessing Systems (SMP) should be hidden well below the hardware abstraction layer, but not all applications will get equal benefits from SMP without understanding what's going on under the hood.

WRITTEN BY
BOB PASKER

This article discusses strategies for achieving the best bang for the buck out of SMP systems. We'll look at design patterns for parallel programming and locking, debugging and profiling, large memory footprints and the effects on garbage collection, as well as tuning and capacity planning.

Java was originally designed as a language for set-top box applications and later became a vehicle for the HotJava browser. Even though Java had first-class support for multithreaded programming (GUI applications have long been multithreaded), application and app server developers continued, with good reason, to treat threads as scarce and expensive resources because the overhead of context switches, per-thread memory, and synchronization was quite noticeable in the sweet spot of server technology – one to four CPUs with up to eight gigabytes of memory. Applications that could not fit into a single instance were deployed as clusters of such instances. Even larger SMP systems, such as Sunfire and Power series boxes, are typically divided into four CPU partitions because of their NUMA architectures and the GC problems associated heaps greater than 4 gig. The literature on scalable Java applications is therefore filled with design patterns and sample implementations of worker thread pools to reduce the number of actual threads in a system, culminating in the addition of of Java5's java.util.concurrent.ThreadPoolExecutor (www.onjava.com/pub/a/onjava/2004/09/01/nio.html http://gee.cs.oswego.edu/dl/cpjslides/nio.pdf http://www-128.ibm.com/developerworks/library/j-jtp0730.html). Rather than being hidden well below the JVM abstraction layer, threads have become an integral and ongoing design point and a tuning headache for developers.

With the advent of better hardware interconnects and multicore chip technologies, much larger flat-memory SMP machines are becoming available, and the debate over threads and threading for scalable Java applications is being reviewed. Our experience at Azul with hundreds of different end-user applications and ISV products running on our Java Compute Appliances has show that although these applications work perfectly on our Java-licensed platform, many of these apps have inherent scalability problems that prevent a single instance from fully utilizing an entire system. The ability to host up to 120 JVMs on the largest appliance concurrently notwithstanding, we have developed a number of strategies to scale a single instance of Java server application.

These strategies span the gamut from configuration changes to devising entirely new algorithms, and I will even propose some additional changes to the Java class libraries to make them scale better.

Since Azul appliances have two orders of magnitude – more CPUs and memory,

the first order of business is to increase the amount of hardware resources available to the application instance. This consists of nothing more than modifying the app server configuration to increase the number of thread-constrained resources: worker threads in the thread pools, adding more MDB listeners, servlet pool instances, etc., and restarting the server. Applications react favorably to this kind of app server tuning, and they take only minutes to implement. There might also be application-defined resources that can be expanded, so take a look at how you use threads and memory to see if there are similar configuration changes that can be made.

Another easy way to take better advantage of a large SMP system is to increase the heap size of the application well beyond the rule-of-thumb numbers (www.128.ibm.com/developerworks/eserver/library/es-was-zseriesfaq.html#ques5, http://java.sun.com/docs/hotspot/gc1.4.2/faq.html, http://dev2dev.bea.com/pub/a/2004/01/chow_deisher.html). Azul appliances have such a large memory footprint that allocating dozens of gigabytes of memory doesn't seem unreasonable. The caveat of large heap sizes for Java applications has been that garbage collection has traditionally subjected applications to infamously long pauses, resulting in unacceptable application response time. Over 90 percent of the large applications we see have heap sizes of fewer than two gigabytes. The way large applications get away with such small heaps sizes is by running multiple copies of the application in a cluster, either on the same machine or on different ones. Such "heap partitioning" has the effect of splitting the heap into small pieces, and a garbage collection (GC) pause in any one server doesn't affect the users on other member of the cluster.

We have only seen a handful of systems with heaps that exceed five gigabytes. These large-memory applications must have their GC parameters tuned to the nth degree, and often, small software or configuration changes can require a lengthy retuning cycle. Interestingly, we have seen two applications with almost 100 gigabyte heaps, and those are on large NUMA SMP systems with very expensive 4 gig DIMMs. The reason they can run with such large heaps is because the heap is large enough that over the course of a workweek, the application never uses enough heap to provoke a full GC cycle, and the application must be restarted every Saturday! On the Azul platform, such shenanigans are unnecessary because of our Pauseless Garbage Collection (PGC) technology that eliminates response time–busting GC pauses (www.azulsystems.com/products/whitepaper_abstract.html). Because of PGC, the max heap size for an Azul JVM is 96 gigabytes, and applications can fully utilize the entire heap. Since the Azul appliances also have an abundance of processor cores, GC will run concurrently with the application, on a set of parallel GC threads, and not take valuable processing cycles away from the application. Increasing the maximum heap size is also a simple configuration fix that can be done on the command line.

> "Multicore systems will eventually transform the landscape of Java development to one that is significantly more scalable than what is currently available"

"Add more memory" (or in Java's case "add more heap") has long been a maxim of smart performance engineers, but what to do with it? For J2EE applications, the additional memory can be used to increase the number of threads to handle more work, and to increase HttpSession and ResultSet caches to reduce the amount of time waiting for the database to respond with a corresponding reduction in database load (www.javaperformancetuning.com/tips/jdbc_caching.shtml). Most systems also have application-specific caches or off-the-shelf caching products that can be enlarged to improve cache hit rates and lower database I/O rates.

Once the configuration has been adjusted to increase the amount of resources available to the application, the instance may still suffer from internal resource constraints. Often this turns out to be one or more synchronized methods that become a point of contention because JVM implementations use pessimistic locking to implement synchronized methods. The Azul platform can also help here because it has support for Optimistic Thread Concurrency (OTC) that permits multiple threads to enter a single synchronized method (www.azulsystems.com/products/whitepaper_abstract.html). Rather than trying to prevent conflicts with pessimistic locking, OTC will detect and repair actual memory conflicts between threads. Wherever there are no memory conflicts between threads, OTC will permit all of the threads to be in the synchronized method simultaneously. The result is that applications will see more concurrency in the application, and the need for fancy, fine-grained, multilevel or reader/writer locking strategies is therefore reduced, thus resulting in faster development time and less application tuning. However there are also some pitfalls to watch out for as well. In particular, be careful about performing I/O or long-running tasks in a synchronized method held because this will reduce concurrency in the application by making waiters wait longer for the holder to release the lock. Doing I/O inside a synchronized method also defeats OTC (www-128.ibm.com/developerworks/java/library/j-threads2.html). The Azul porduct also includes a JVM dashboard that will display the wait time and queue depth for the locks in the system. By looking at the top contended locks, you can zoom in on which parts of your application needs to be examined and improved to get more concurrency and therefore more work done.

> Increasing the maximum heap size is also a simple configuration fix that can be done on the command line

Another type of resource starvation problem stems from an application's inability to take on more work. For example, we encountered an Interactive Voice Response application with only a single thread performing accepts on a ServerSocket. The entire application was therefore bounded by the single-threaded accept rate. The single accept thread was sufficient to saturate the eight-way box because the work arrival rate could exceed the service rate, but with so many processor cores available to process transactions on Azul, the queuing paradigm was inverted and the arrival rate was much lower than the service rate. Therefore adding more load increased the response time even though there were plenty of idle cores due. The solution was to modify the application to launch multiple accept threads on different ports, and once this was accomplished, many more sockets could be handled, which meant more work came into the system, and more processors were loaded up with work. The lesson here is to look not only for contention bottlenecks in your application resulting from locks, but also bottlenecks that might be used to move work into or out of the system.

Once you have provided the application with all of the processor and memory resources it needs, removed as much contention as you can, and made sure that enough work can get into and out of the system, it's time to start looking at improving application algorithms. The theoretical basis for this step is known as parallelization, and a taxonomy has been developed to describe various levels of it. The most parallel algorithms are known as "embarrassingly parallel" (EP), meaning that no particular effort is needed to segment the problem into a very large number of parallel tasks. Probably the most well-known EP problem is Set@Home (and its cousins), but Monte Carlo simulation is frequently encountered in commercial applications to assess risk and predict returns of various financial instruments. Certain sorting and searching algorithms are parallelizable, so look for opportunities to split such problems up into smaller pieces (see *Sorting and Searching* in www-cs-faculty.stanford.edu/~knuth/taocp.html).

For non-J2EE applications, there are a number of clever technologies for parallelizing algorithms, not the least of which is Doug Lea's Fork/Join Framework, which has been shown to be much more efficient than java.lang.Thread, and it includes techniques for both work stealing and efficient push and pop operations (http://gee.cs.oswego.edu/dl/papers/fj.pdf). Because many large Java applications have dynamic workloads, divide-and-conquer algorithms must be careful to only use free resources, rather than taking over the entire server. Indeed, Azul provides the ability to set minima and maxima for the processor count guaranteed to each VM, so on that platform it would not be possible to starve other applications. Nevertheless, within a single JVM, it would be possible to starve other threads. java.lang.Runtime.availableProcessors()gives a static view of the number of processors available to the JVM, but we are working on extensions that will give an instantaneous view of free resources available to the JVM.

For J2EE applications, it's not possible to spawn threads inside, for example, a servlet or an EJB, and the JSR for an App Server Work Manager now seems to be dead (http://jcp.org/en/jsr/detail?id=237). Developers will have to resort to something like MDBs to pawn work off on other threads. J2EE apps often have transaction types that can be decomposed into smaller pieces that can be performed simultaneously, such as searching multiple databases, prewarming a database or object cache, or processing a large batch of work. By decomposing these transactions into smaller pieces and then generating messages for an MDB, you can gain additional parallelization. This mechanism could be expensive, however, and you should ensure that the additional cost of generating and running the MDBs is worth the effort versus performing the operations in-line to avoid thrashing.

One of the difficulties sysadmins experience with J2EE is the amount of time it takes an application to start up inside a container. It should be possible for the application server vendors to take advantage of SMP systems by multithreading their startup sequences. One caveat for application developers however, is that the current startup is likely single threaded, and there may be unexplored dependencies in the application code that will behave improperly when multithreaded. Indeed, this is a more generic problem: anytime you write a component that is not explicitly single threaded (such as an EJB or a servlet inheriting from SingleThreadModel), be careful to make sure your code is multithread safe!

Within the Java class libraries, there are numerous algorithms that could also benefit from parallelization – specifically those associated with searching and sorting of collections. For example, the current implementation of Collections.sort() is single-threaded (http://java.sun.com/docs/books/tutorial/collections/algorithms/index.html). Imagine, however, if the sort algorithm were sensitive to the size of the problem and the available system resources to divide the sort up into smaller chunks and merge the results. If the complexity of the sort algorithm is O($f(n)$), then dividing the problem into p pieces running on u threads results in a wall-time execution of $O(f(n/p)/u) + C(p)$), where $C(p)$ is the cost of dividing the problem into p pieces and merging the results. It is easy to see that as n grows, the benefits of divide and conquer increase so long as no other part of the system is inadvertently starved. Indeed, the cglib package already provides alternative parallel sorting algorithms that could serve as the basis for an improved implementation of Collections.sort() (http://cglib.sourceforge.net/apidocs/net/sf/cglib/util/ParallelSorter). Hopefully, the Java Community Process (JCP)

will take on the task of parallelizing the Java class libraries.

In Java5, we have the addition of Core XML Services for managing XML documents. Although parsing XML is not necessarily parallelizable, it would be possible to create an XML parsing and XML transformation pipeline, consisting of a set of threads connected by queues. The threads in the pipeline would be a lexer, a SAX parser, and a DOM filter. In the parser, the last element in the pipeline is an application-specific processing module that would make callbacks to the application. In the transformer, the last element in the pipeline would be an XSLT engine. By running each pipeline thread on a separate core, large XML documents could be parsed and transformed in much shorter wall-time, which is important for Web services–based systems.

Last, let's take a look at the internals of the JVM itself. The Hotspot VM already runs the JIT in a separate thread, which means that the VM will continue running bytecodes at the same time as it is generating native object code. However since every method is separately compiled, JIT performance will improve when multiple threads are used by the JIT. Azul's JIT (derived from Hotspot) already takes advantage of this, and as JIT heuristics improve, the penalty for throwing out and regenerating code will be reduced, so we will see JIT heuristics improve as well. To the extent that a JVM has to share internal data structures such as class data structures across threads, we will see JVM vendors improve overall JVM performance by further internal turning for increased concurrency.

As we have seen, multicore systems will eventually transform the landscape of Java development to one that is significantly more scalable than what is currently available. The impact will be felt at development time by requiring developers to be more cognizant about concurrency and starvation issues, and to look for ways to increase parallelism in their applications; during load testing and tuning, by requiring developers, performance gurus, and sysadmins be savvy about how resources are used in the application or app server; and even by the app server vendors and JVM writers, who are responsible for making sure that their technologies can take as much advantage of multicore systems as well. The great benefit of multicore systems is that they will be much more scalable than their one to four CPU brethren and require a smaller number of physical boxes to satisfy the same load. This will save space, energy, and management time, and match today's CIO's goals of improving overall data center consolidation.

About the Author

Bob Pasker is the deputy CTO of Azul Systems. He has been designing and developing networking, communications, transaction processing, and database products for 25 years. As one of the founders of WebLogic, the first independent Java company (acquired by BEA Systems in 1998), he was the chief architect of the WebLogic Application Server, which today still dominates the market. Bob has provided technical leadership and management for numerous award-winning technologies, including the TribeLink series of routers and remote access devices, and the TMX transaction processing system.

bob@azulsystems.com

> "The lesson here is to look not only for contention bottlenecks in your application resulting from locks, but also bottlenecks that might be used to move work into or out of the system"

Service Taxonomy and Service Ontologies Deliver Success to Enterprise SOA

An aid to linking the business and IT architecture through service classification

A lot has been written on the approach to service-oriented architecture (SOA) migration. Although they are referred to by many names, there is the *strategic approach*, which is of high quality and so is also costly and initially less responsive because of the analysis involved up front. Then there is the *organic growth approach*, in which services are developed on an as-needed basis within the context of projects, which is responsive, but leads to redundancy and the lack of vision leads to unmanageability later. Finally, there is the *hybrid approach*, which attempts to take the best of both of these worlds. It is so very important that the business analysis is not cast aside when developing the SOA through this hybrid approach to migration.

What we will discuss here is just one facet of ensuring a successful migration to an SOA, that of *Classification Analysis* and the use of service *taxonomies* and *ontologies* to define these classifications. Where we will differ slightly from other discussions is in our use of classification of services to help provide the necessary bridge between the complexities of services and that which the business understands of the enterprise.

WRITTEN BY
MARTYN HILL

All Services Are Not Created Equal

At the end of the day we have a lot of functional assets, business processes, and data locked up inside legacy systems. We do not want to reinvent the wheel for new business requirements; thus, through SOA we are attempting to expose those functional assets for reuse. A typical large enterprise can expect to have 100+ course-grained sets of services. However, many of these services are likely to be processes or composites of other lower-level services, which in turn could be reliant on data services, and so on. The end vision could end up with thousands of services to be managed (not counting versioning). A large catalog with no structure, as shown in Figure 1, is not conducive to reuse by either the business when developing requirements or IT when implementing new features or composite applications.

The Need for a Service Taxonomy

In order to manage the complexity and potentially the sheer number of services, it is quickly apparent that we need to first "horizontally" manage the classification of services.

In the beginnings of the focus on SOA it was typical to only talk in terms of a simple set of horizontal service types: shared business services, shared application services, and shared data services. While this is still good for articulating the concept of SOA, it is necessary to go far beyond this for the actual implementation and management of services.

Figure 2 shows a simple expansion of the horizontal classification of services. Although there are many published versions of this, it is not a "one size fits all" and should vary by scope and enterprise size. What is important is that the need is recognized and a suitable model adopted. An SOA implementation will mature and the organization will become more sophisticated in its understanding. It is therefore important to have the ability to recognize the need for change and adopt change in the classification models selected.

Some of the benefits that may be experienced from a horizontal taxonomy such as this are:

- Help in defining the complexity of service implementation and hence assistance in estimating the cost of implementation and/or change
- Garnering a better understanding of granularity of services and hence how composable they may be
- Assisting business in understanding the level of reusability when considering new

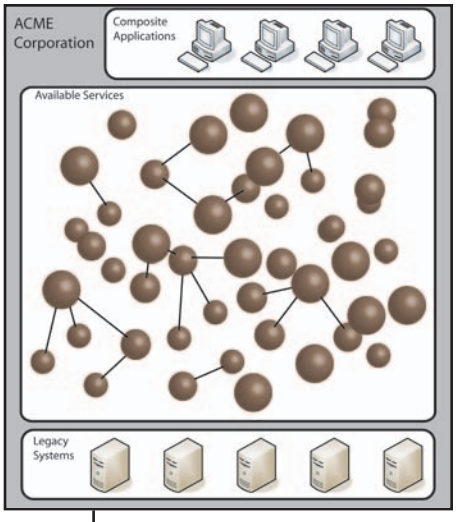


FIGURE 1 **A service catalog without any formal structure**

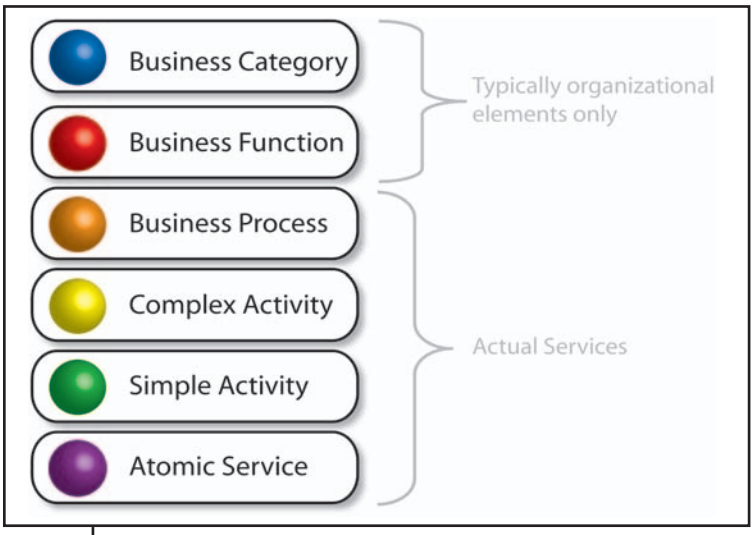


FIGURE 2 **A simple horizontal taxonomy**

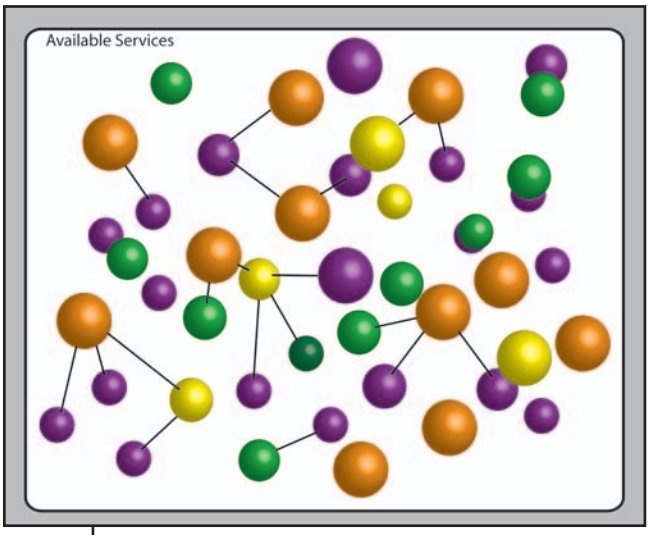


FIGURE 3 **Introduce the horizontal taxonomy – a stepped improvement**

requirements perhaps targeted at the use of a particular service
- Aid in analysis of identifying higher-level composites and façade services from atomic services to proactively populate the service catalog
- Specifying principles for service interaction, such as "services may only utilize services from the same or lower layers," etc.

If we apply this horizontal taxonomy to our previously unstructured service catalog, as shown in Figure 3, we now have achieved a much better understanding of our service capability.

This still does not provide enough clarity and we need further "vertical" classification to assist in simplifying the service catalog.

The Need for Service Ontologies

When we suggest the use of "vertical" classification, it is not really a single set of columns applied to our original taxonomy (i.e., just another taxonomy), but what is needed is a more rich, semantic, and hierarchical understanding of the services available, specifically in terms of their relationship to the business.

There is a lot of discussion about using ontologies to help map semantics of data between different models. Although this is a valid point, there is a far more fundamental set of benefits to be realized in using ontologies to provide a vertical classification of services:
- A common language is established that makes the technical and business problem domains more understandable
- Because understanding is generalized and removed from the context of potentially multiple source systems, a virtual model is being established that promotes interoperability
- Achieves a greater simplification of the development of business requirements
- An enterprise-wide understanding of the functional architecture of the business is established
- Shows the progression or lack thereof of service catalog population, and hence helps plan and prioritize the population of the service catalog
- Establishes understanding of when a service is overstepping its bounds
- Makes identification and management of service ownership much simpler

It requires comprehensive business analysis to form a model of the functional domain model and business processes that constitute the enterprise. This is no easy task and why when it comes time to develop this understanding, many SOA strategies balk at this point and turn back to the *organic approach*.

Don't give up! Take onboard an incremental analysis of the enterprise. Also, many vertical industries already have useful information or functional domain models that can accelerate the development of initial levels of service ontology, such as the eTOM for the telecommunications industry, OTA for travel industry, OFX for financial industry, etc. The enterprise should leverage wherever existing work has already been conducted and where shared ontologies already exist. Remember, the intent is to develop a semantic understanding of services within the business today, and not to redesign the business architecture for a future vision (one step at a time).

Figure 4 shows a very rudimentary functional domain model for our fictional company that can be applied as the higher levels of a service ontology. We have simply applied the top two organizational layers of our service taxonomy from earlier. Of course over time this could become much more sophisticated and there could be a service hierarchy established within these two layers. For example, when working with one client with a

> "The enterprise should leverage wherever existing work has already been conducted and where shared ontologies already exist"

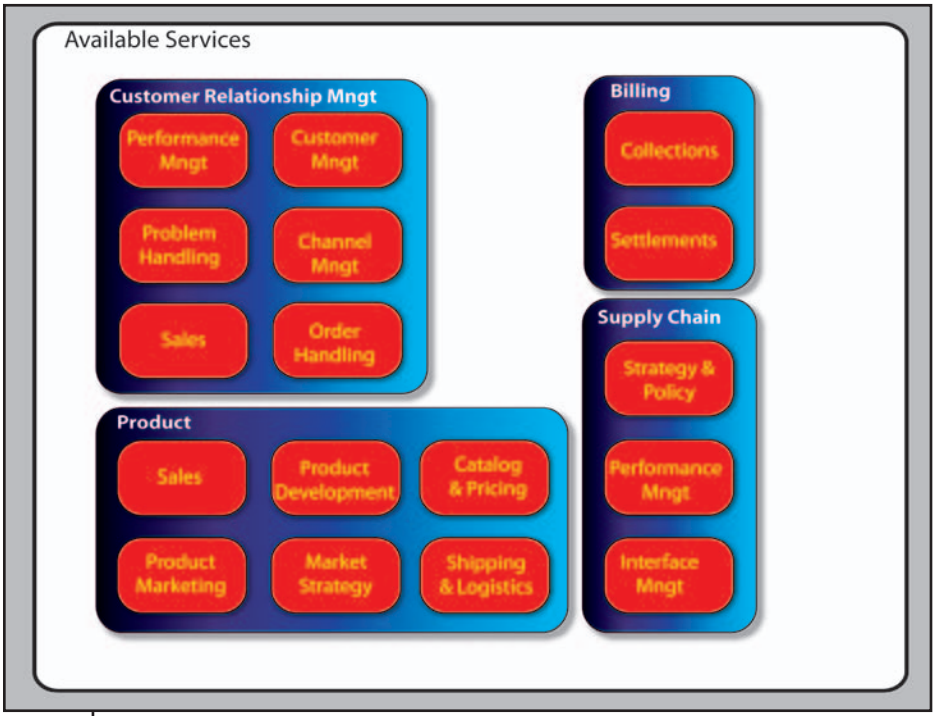


FIGURE 4 **Vertical industry model for introducing a service ontology**

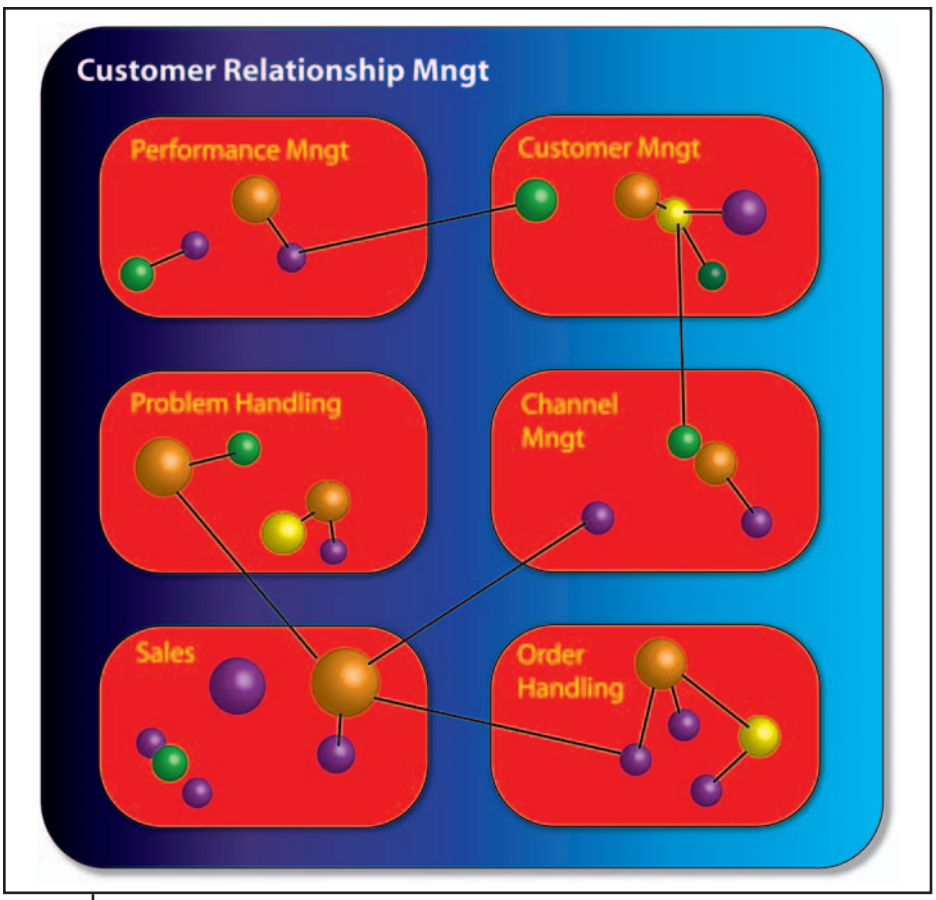


FIGURE 5 **Functionally mapping services to their Business Category**

very large SOA implementation we established a functional domain model with five layers of decomposition.

Ontological analysis should be conducted from the outset and the functional domain models of the business should be established. This will take time and effort and should not stall or inhibit the development of services, but is conducted in parallel (as per the *hybrid approach* to migration). There *must* be a feedback loop established in the governance processes to ensure that any ontology is updated with real understanding from a project, and that ontology is refactored into services when sufficient analysis has been completed (for example taking a "while the hood is up" approach).

Figure 5 shows how the services are now classified according to their functional alignment. We have only zoomed in on the Customer Relationship Management business category, but it is evident that the services are much more understandable now that their vertical and horizontal classifications have been made apparent. It should be noted that the use of a vertical service ontology would predominantly benefit the classification of shared business services rather than other types of services.

Other Useful Classifications

The more useful metadata that can be established regarding the services catalog, the better. Furthermore it is particularly true when attempting to establish that "ever so desirable" business alignment. Other equally useful classifications could be:

- Quality of service
- Versioning services and build metadata
- Scoping services

Management of Classification Mappings

The goal should be that the service ontologies are implemented and managed in software, but this is not crucial to the initial success of the SOA migration, particularly if IT is still implementing either the pilot or early incremental adoption phases. What is far more important at this stage is that the need has been recognized, that analysis has begun, and that it is a *part of the governance process*. Although it is difficult to see and measure, this

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

REAL-WORLD AJAX ONE DAY SEMINAR

www.ajaxseminar.com

March 13, 2006

Marriott

Marriott Marquis Times Square
New York City

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

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

How soon can you be monetizing AJAX?

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

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

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

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

Jesse James Garrett
Father of AJAX

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

Bill Scott
AJAX Evangelist of Yahoo!

David Heinemeier Hansson
Creator of Ruby on Rails

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

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

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

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

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

MEDIA SPONSOR

ajax developer's journal

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

PRODUCED BY SYS-CON EVENTS

> Use taxonomies to define the simple classification types of services and use ontologies to define the semantic understanding of services within the context of the business

work has already helped mitigate great risk and improved the quality of service implementations for now and in the future.

When ontological semantics and mappings are implemented in a server, a further stepwise benefit is achieved over this aforementioned initial recognition benefit; that is, semantics and mappings of the enterprise are centralized. Also, semantic knowledge and transformations are no longer exploded throughout the SOA orchestrations, business process management (BPM), or enterprise application integration (EAI) layers, but are reused and well understood. Even with an SOA (or EAI platform), previously transformation between loosely understood semantics would typically be held in design or data mapping documentation at best rather than centrally managed in a server. Transformations themselves can now become more reusable.

A key element in making successful use of the developed classification models is ensuring that their existence is communicated across the enterprise, or at least to those that should know. This includes the CIO office, development, architects, business analysts, operations planners, and so on. This should not be thought of as just some metadata information to be used exclusively by development. We need this information to be up to date and accessible to all of the business, not just the Web service developers. Similar to the service catalog itself, information should be published not only in a set of tools but in diagrammatic and document form so that all can understand and make use of it. Again, its formalized usage, review, and update should be included as part of the SOA governance.

Supporting Technology

So what do you use to implement your taxonomy, ontologies, and mappings? Well again, one size does not fit all, and that depends on the scoping of the SOA strategy and the size of the enterprise.

Although the following is not an exhaustive list, it is clear that this undertaking it is much more feasible with standards such as Universal Description, Discovery, and Integration (UDDI) to implement the service registry and associated taxonomies, and perhaps Web Service Modeling Ontology (WSMO), Web Ontology Language (OWL), or Semantic Markup for Web Service (OWL-S) to implement the semantic model.

When using WSDL with UDDI we are already creating metadata that represents the service itself. We can create further information taxonomies using the categorization tModel of UDDI. Once a tModel has been established, Listing 1 shows how we might classify a service as part of our horizontal taxonomy from earlier.

Taking the example of using OWL we can build up (document) a reasoning of the implemented services, in our case understanding the services context within the business. Because we are only looking to implement a classification hierarchy to ensure successful SOA migration, OWL Lite is more applicable (of the three sub languages offered). Hierarchies of information can be built up as shown in Listing 2.

It is not, however, the purpose of this article to detail these technology standards (there are plenty of documents detailing their usage), but to reinforce that a clear plan should be established to include classification analysis in the SOA migration as a best practice.

Summary

There are never enough ways that we can abstract our understanding of the service catalog when it is considered holistically. Classifications don't have to be about just aiding run-time binding or populating low-level registry information for Web service developers' benefits only. We should also be focused on establishing models that allow for successful management and ease of identification of services for reuse at requirements and design time.

Use taxonomies to define the simple classification types of services and use ontologies to define the semantic understanding of services within the context of the business. This will help ensure that the necessary link between business understanding and IT is achieved and maintained in actuality for the catalog of services.

About the Author

Martyn is an enterprise architect with over 19 years of experience in an engineering environment. He is currently a principal architect with CSC Consulting's national practice, specializing in enterprise architecture. He has led the successful development and implementation of strategic architecture and roadmap visions for SOAs, enterprise application integration, Web portals, business gateways, and Web services management platforms for large-scale enterprises.

martyn.hill@adelphia.net

Listing 1: Using a tModel to classify a service

```
<businessEntity businessKey=...
    ...
    <categoryBag>
        <!-- Classify using the
          horizontal taxonomy -->
        <keyedReference keyName=
          "BusinessProcess"
        keyValue="21512"
        tModelKey="uuid:ACMECorp:
B8E5BB2"/>
    </categoryBag>
...
</businessEntity>
```

Listing 2: OWL Class Definition

```
<owl:Class rdf:
ID="CatalogAndPricing">
  <rdfs:subClassOf rdf:
    resource="Product"/>
  ...
</owl:Class>
```

Welcome to the Future of Video on the Web!

REGISTER NOW!
www.iTVcon.com

CALL FOR PAPERS NOW OPEN!

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

iTVCON.COM
INTERNET TV CONFERENCE & EXPO 2006

Coming in 2006 to New York City!

"Internet TV is wide open, it is global, and in true 'Web 2.0' spirit it is a direct-to-consumer opportunity!"

For More Information, Call 201-802-3023 or Email itvcon@sys-con.com

List of Topics:

- Advertising Models for Video-on-demand (VOD)
- Internet TV Commercials
- Mastering Adobe Flash Video
- How to Harness Open Media Formats (DVB, etc)
- Multicasting
- Extending Internet TV to Windows CE-based Devices
- Live Polling During Webcasts
- Video Press Releases
- Pay-Per-View
- Screencasting
- Video Search & Search Optimization
- Syndication of Video Assets
- V-Blogs & Videoblogging
- Choosing Your PVR
- Product Placement in Video Content
- UK Perspective: BBC's "Dirac Project"
- Case Study: SuperSun, Hong Kong

Welcome to the Future!

Did you already purchase your ".tv" domain name?
You can't afford not to add Internet TV to your Website in 2006!

2005 was the year of streaming video and the birth of **Internet TV**, the long-awaited convergence of television and the Internet. Now that broadband is available to more than 100 million households worldwide, every corporate Website and every media company must now provide video content to remain competitive, not to mention live and interactive video Webinars and on-demand Webcasts.

20 years ago the advent of desktop publishing tools opened the doors for the creation of some of today's well-known traditional print media companies as well as revolutionized corporate print communications. Today, with maturing digital video production, the advent of fully featured PVRs, and significant advances in streaming video technologies, **Internet TV** is here to stay and grow and will be a critical part of every Website and every business in the years to come.

It will also very rapidly become a huge challenge to network and cable television stations: **Internet TV** is about to change forever the $300BN television industry, too.

The Internet killed most of print media (even though many publishers don't realize it yet), Google killed traditional advertising models, and **Internet TV** will revolutionize television the way we watch it today. You need to be part of this change!

Jeremy Geelan
Conference Chair, iTVCon.com
jeremy@sys-con.com

PRODUCED BY SYS-CON EVENTS

Track 1	Corporate marketing, advertising, product and brand managers
Track 2	Software programmers, developers, Website owners and operators
Track 3	Advertising agencies, advertisers and video content producers
Track 4	Print and online content providers, representatives from traditional media companies, print and online magazine and newspaper publishers, network and cable television business managers

SOA Will Kill the Offshore Model

Can SOA really impact geopolitics?

I know what you're thinking: SOA hype has reached an absurd level and now someone is literally proclaiming that it will change the world – but bear with me for a minute. Anyone who has been around corporate IT for the last five years or so has seen an avalanche of development work sent offshore for two primary reasons: cheaper unit labor cost and the flat-out inability to find qualified American developers. Also, the mainstay system development model whereby business units built up app silos, which served to minimize reuse and increase integration complexity, demanded a way to deal with the cost of its own inefficiency.

WRITTEN BY
PAUL O'CONNOR

Offshoring was perfect – you could justify using twice as many developers as necessary when they cost one third as much. The trend hasn't been just limited to corporate IT either. Offshore workers are handling many other business functions, though these are mostly lower-level functions like call centers and help desks.

On a human level, I think offshoring has been a great thing. It has uplifted third-world economies and given opportunity to people whose talent means more now than their location. Furthermore, it sure seems that countries that benefit from this type of globalization are more stable than those that don't. Unfortunately, the results for business have been more mixed. Poor management and architecture has lead to many project failures along the way.

Corporate IT is now entering a new era. It's hard to find a large- or medium-sized enterprise that is not at some point along the SOA adoption curve. Some have charted a course for agile business process management while others are being dragged along as their platforms and tools make the transition under them. However what makes SOA a tectonic shift in enterprise computing is that it is being driven by the business. For once the move to retool is not being driven by techies or platform vendor salesmen. Business managers have grasped that implementing SOA standards serves to manage complexity, which in turn drives cost down; however to business, SOA is more about competitive advantage through agile management of processes and rules than pure cost containment.

"Agility" in SOA terms can be best described as a measure of the cost (dollar cost and opportunity cost) of implementing changes in IT systems. The old IT release cycle that never kept up is at one end of the spectrum, and the future of near-instantaneous changes affected by business analysts *without* IT staff intervention is at the other. The ability to quickly add or change business processes brings companies closer to their customers, partners, employees, and anyone else with whom they choose to interact. The more agile the enterprise, the closer the bond is – and the greater the competitive advantage.

Remember that we are talking about the future. No business analyst today composes new business processes and puts them into production, but that is exactly where the art is headed. SOA standards have always promised complexity management and cost savings by relegating to the infrastructure functions that are currently developed over and over again in applications. Additionally, now we are seeing that concept taken to the nth degree with the new Service Component Architecture (SCA) standard that promises declarative, on-demand infrastructure for services. Infrastructure pieces will be boiled down to checkboxes and radio buttons on the analyst's process management session as they compose new services and rules. To conceptualize this scenario, think of the analyst enabling a service for external federation via a checkbox and then being assured that it will be governed under the company's trust and access control policies. To be sure, a great deal of effort from IT will have gone into delivering that assurance to the analyst, but all in the infrastructure – none at the level of the services themselves.

> "Offshoring was perfect – you could justify using twice as many developers as necessary when they cost one third as much"

So how does this developing trend affect offshoring? It greatly diminishes the need for application developers, and instead separates corporate IT into two worlds: infrastructure management and business logic management. These are the two areas that have remained largely onshore for various risk management reasons. Instead, it seems clear that we will see a shift to higher-order functions being handled offshore – published as services and incorporated into business processes of trading partners without respect to locale. Also, remember that we are talking about corporate IT. The infrastructure elements themselves will no doubt continue to be developed in large measure outside of Europe and America as they are today.

Everything from financial services to energy trading to macadamia nut fulfillment will be published to service registries and anxiously consumed by business analysts around the world to save money on transactions or to introduce new functions. This will usher in an era where countries can trade on their intellectual capital at the service fulfillment level and usher out an era where they have to send that capital overseas or otherwise sell it piecemeal to foreign firms. Return on investments in education and technology will be much quicker in this climate, globally. Moreover, it will no doubt refocus governments on being reliable service fulfillment platforms. Too utopian you say? Possibly, but it seems clear that SOA is going to serve as a key facilitator in accelerating business globalization, and that the old offshore model will fade away.

> "The ability to quickly add or change business processes brings companies closer to their customers, partners, employees, and anyone else with whom they choose to interact"

About the Author

Paul O'Connor is chief architect and SOA Practice managing principle for e-brilliance LLC (a leading NE SOA consultancy), and is currently doing major SOA architecture and implementations for Fortune 100 clients across the US. Previously he was chief architect for Damascus Road Systems, specializing in security architecture.

poconnor@e-brilliance.com

BUILDING BLOCKS OF SOA GOVERNANCE

Establishing demand and supply centers is a reliable approach for SOA governance

SOA initiatives have gathered momentum in the past year with more enterprises either implementing SOA or considering implementing in the near future. The implementations we studied reveal that one of the critical challenges in SOA is designing an effective governance mechanism. A good understanding of governance concepts is essential to implementing and operating a successful SOA. Reliable governance for SOA leads to a manifold increase in an enterprise's ability to achieve the goal of business agility through SOA.

WRITTEN BY

N. DAYASINDHU

SRINIVAS PADMANABHUNI

SRIRAM ANAND

Defining IT Governance

The IT Governance Institute defines IT governance as "a structure of relationships and processes to control the enterprise in order to achieve the enterprise's goals by adding value while balancing risk versus return over IT and its processes." Another definition by Peter Weill describes IT governance as "specifying the framework for decision rights and accountabilities to encourage desirable behavior in the use of IT." The objective of IT governance is to assist enterprises in leveraging IT to achieve business goals, while governance is essentially the structure, the roles, and the responsibilities that help deliver IT services effectively and efficiently. Successful governance mechanisms that help enterprises meet their business goals typically consist of simple and transparent mechanisms. Half of the managers in the top 50 percent of governance performers could explain governance, while fewer than 30 percent of managers could do so among weaker performers (see second reference in the References section)!

Governance Guidelines for SOA

The IT organizations today are dominated by a central IT function. The IT function has a near-unilateral responsibility for governance. However many business units have established in-house IT functions that work in tandem with the central IT function to cater to specialized needs. This bifurcation of roles and responsibilities between in-house and central IT functions is easy since most applications are "owned" by a business unit that controls budget for design, development, and support for specific applications. This is the first point of departure in the service-oriented architecture (SOA) context where multiple business units "own" and "use" the same set of services. This implies that the aggregation of requirements for services now comes from multiple business units while the budgets for design, development, and support for specific applications have to be apportioned among multiple business units. In such a case the delivery of services will necessarily have to come in from a central IT function rather than from in-house IT functions in business units, and the central IT function will enter into service-level agreements (SLA) with multiple business

functions for provisioning the same services.

The following are typical scenarios that we have come across in large enterprises that are adopting SOA:

- A large bank where the IT function developed a proof of concept of SOA architecture and was contemplating next steps. The challenges for the bank were: How to obtain a buy-in from business functions on moving to the SOA architecture? What will be an ideal governance mechanism in the steady state with SOA?
- A health insurer was in a legacy modernization program across the enterprise with SOA and introduced a new organization between IT function and business functions to own the services.
- In another large bank's migration to SOA, a business division with senior executive commitment pilots the migration helped get budget commitment for the enterprise SOA initiative.

We believe that governing SOA is more centralized than traditional shared services for IT applications, and this requires tweaking existing governance models to provide guidelines to address the challenges posed in the SOA context. The following are typical challenges in designing practical SOA and their solutions.

Challenge of single ownership

Proxy for single ownership that is managed by a new organization layer between central IT and business functions. Establish SOA governance committee with representatives from business units and central IT function.

Challenge of managing multiple owners

Map services to business processes/projects/cost and profit centers to apportion investment and operating costs. Usage-based funding appears elegant but may be arduous to implement enterprise-wide and works best in a high-trust environment. An SOA governance committee/SOA management organization decides on investment and running costs based on previously agreed upon cost-apportionment rules. Prioritization of enhancements and new development is determined yearly/semiannually by the SOA governance committee/SOA management organization.

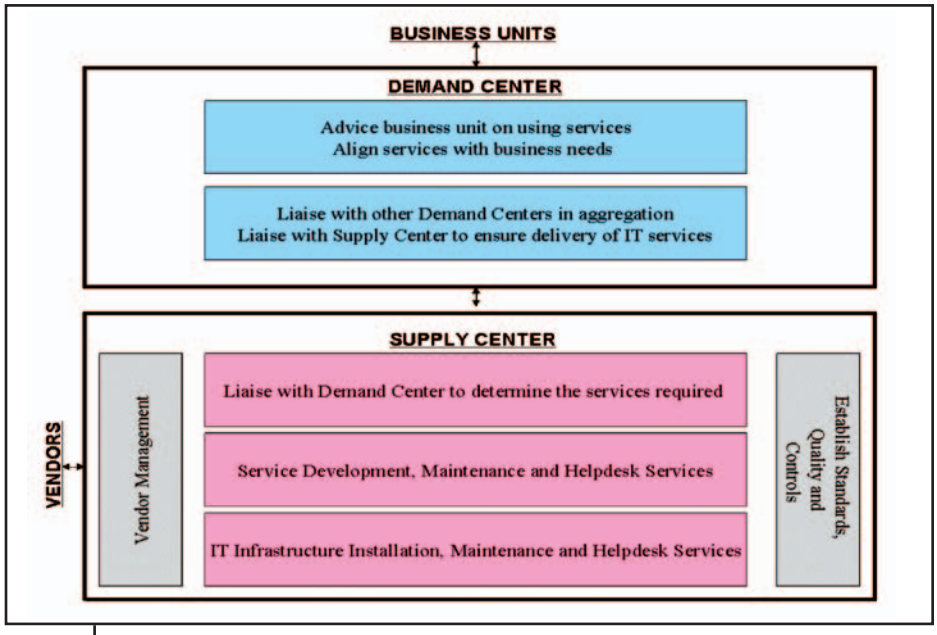


FIGURE 1 | **Roles and responsibilities of demand and supply center based on INDIGO**

Challenge of aligning SOA with enterprise IT architecture

SOA needs to be consistent and aligned with enterprise-wide IT architecture policies with a representation from the enterprise architecture group part of the SOA management organization/SOA governance committee. The same team should set IT policies, deliver and maintain SOA infra/development/maintenance, manage vendors, and ensure quality of service (QoS). A significant part of this alignment would be deployment of appropriate IT infrastructures like policy registries, policy repositories, and policy management infrastructures to enable this alignment.

Challenge for small enterprises that cannot afford costly governance mechanisms

Heads of the business units decide on governance mechanism for ownership and funding at periodic meetings, while day-to-day operations reside with IT managers in the function/unit.

These guidelines determine the SOA governance model an enterprise desires to establish based on a bifurcation of demand (to mitigate ownership challenges) and supply (for provisioning enterprise-wide, shareable, standardized services). The governance guidelines are based on the INDIGO (Infosys Design for IT Organization Governance) research program.

> "The role of the demand center is to advise business units on business-IT alignment in the context of SOA"

Demand and Supply Centers Are the Cornerstones of the Governance Model

One of the key guidelines in INDIGO revolves around bifurcation of responsibilities in services provisioning between demand and supply centers. The rationale for bifurcating the IT function into demand and supply centers is based on the premise that it enhances

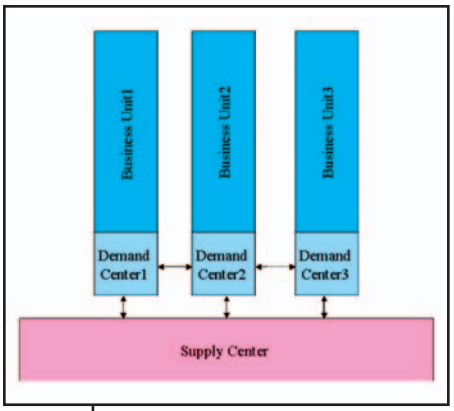


FIGURE 2 | **The "embedded" demand center model**

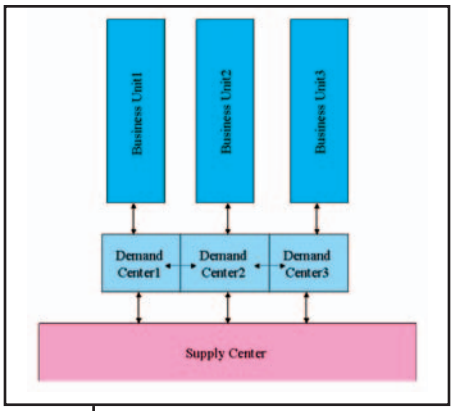


FIGURE 3 | **The "floating" demand center model**

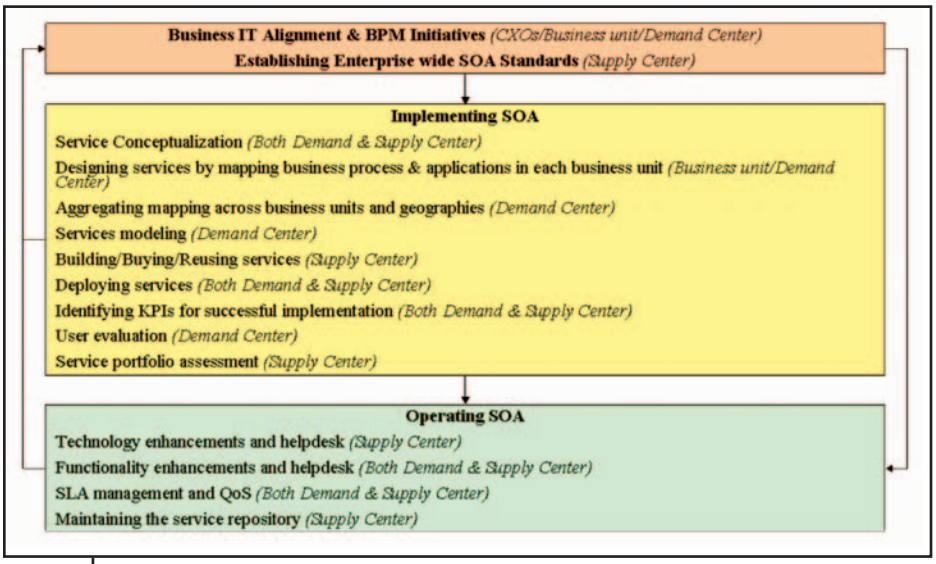


FIGURE 4 | **Roles and responsibilities in implementing and operating SOA**

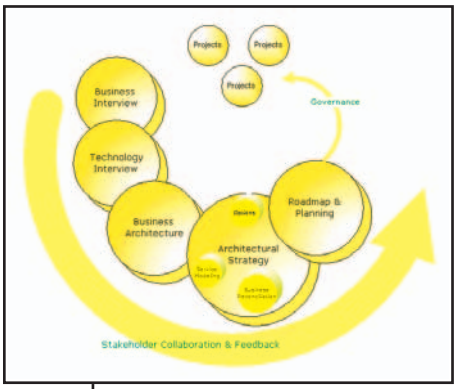


FIGURE 5 | **INSOAP phases**

accountability to the business unit, leverages scale and scope economies for delivering services, and prescribes clear roles and responsibilities, which are illustrated in Figure 1.

The role of the demand center is to advise business units on business-IT alignment in the context of SOA. The analysts in the demand center understand the language of business and bring SOA closer to the business users; additionally, they usually come from the business units (seen as "one of us" by the business users) and have a good appreciation for IT. The demand center focuses on a business case–driven approach to how SOA can increase the effectiveness and efficiency of business processes. Once the demand center has identified the services required by the business users, it passes on the requirements to the supply center that delivers the services.

The aim of the supply center is to deliver best-in-class services that are cost competitive, of high quality, and on time. The supply center has a world-class SOA infrastructure to provide these services to the business users. The SOA infrastructure can be either in-house or, as increasingly is the case, outsourced to best-of-breed IT services vendors. The supply center does not need to be collocated with the demand center, but it can be based in geographies that have the best competitively priced delivery capabilities. The supply center is accountable to business users and this is governed by SLAs.

The supply center is typically the existing central IT function and is headed by the CIO of the enterprise. The supply center is responsible for delivering new SOA-based services, supporting existing systems, and running the SOA infrastructure. The supply center usually needs three divisions that:

- Establish enterprise-wide SOA infrastructure standards and provide associated infrastructure services
- Manage design and delivery teams for providing SOA-based services for business units in collaboration with the respective demand centers
- Perform vendor management, SLA management, quality management, and internal control activities that ensure smooth delivery of services

INDIGO identifies two governance structure models: "embedded" demand center and "floating" demand center models for enterprises adopting SOA, as shown in Figures 2 and 3.

The "embedded" demand center has a clear demarcation of roles, responsibilities, and reporting of both demand and supply centers; however, implementing the structure may be time consuming because most structures today that have a powerful central IT organization with a few large and powerful business units have in-house IT organizations. We believe that the "embedded" demand center model is best suited for those enterprises where business units have good IT systems analysis appreciation and skills (typically found today in industries like hi-tech, CSP, and financial services). The "floating" demand center has a clear demarcation of roles and responsibilities of supply center while the demand center is a separate entity under partial control of both business units

and the supply center. This model may be easier to implement since this is not a major shift from the prevalent traditional centralized IT organization structure. Even if this model has an ease of implementation and the roles and responsibilities of the demand center are clear, there may still be some ambiguities in reporting relationships because there is a dual reporting to both the business unit and the supply center. We believe that the "floating" model is ideal when transitioning from traditional to "embedded" model as a transitional phase.

Whichever model is selected by an enterprise, it is important to understand the roles and responsibilities for the demand and supply centers in the various stages in SOA implementation and operations.

Responsibilities in SOA Implementation and Operations

The objectives of SOA governance are to identify the services that business users require and to focus on delivery excellence in provisioning these services. The roles and responsibilities for the demand and supply centers in implementing SOA in an enterprise entail keeping in mind that the twin objectives of governance is met. Figure 4 lists the roles and responsibilities in SOA implementation and operations. The roles and responsibilities mapping is based on the INDIGO premises that:

- Business units along with the demand center have control of how to leverage IT in their business processes
- Business units have trusted partners in the demand center who understand them and can translate business needs into requirements for services
- Supply center can leverage the economies of scale that are provided by aggregating the services demanded by different business units
- Supply center can exclusively focus on providing cost-competitive, high-quality services on time

Implementing SOA

A strategic decision to implement SOA is made after a buy-in from the CXOs based on the recommendations of the business-IT alignment, business process management (BPM) initiatives, and the supply center establishing enterprise standards for SOA. INSOAP (Infosys Service Oriented Analysis/Adoption Process) is a process to design and realize SOA in order to achieve a better business and IT alignment. Typically, the decision to implement SOA will be in phases, and one such phased implementation that is in line with The Open Group Architecture Framework (TOGAF) is shown in Figure 5 (see the third entry in the References section).

Some of the important phases in implementing SOA based on the INSOAP are:

- As-is and to-be modeling
- Process-to-application mapping and assessment
- Services identification
- Services definition and modeling
- Services realization
- SOA hosting
- Project management
- Governance

The key processes of implementing SOA based on INDIGO are depicted in Figure 6. One of the first steps in implementing SOA is the demand center's mapping the as-is business architecture and process/applications to services mapping. Simultaneously, the supply center maps the as-is technical architecture and application portfolio metadata. Both the demand and supply centers jointly decide on the to-be architecture and SOA solution blueprint for the business units based on a service identification and matching process. The blueprint is the basis for the demand and supply centers definition of the granularity of services, classification and matching of services, and consolidation and rationalization of services. While the demand centers work on the service contracts and the service data model, the supply center works on service realization and hosting. Project management, risk assessment and mitigation, business continuity planning, service governance, and management are jointly performed by the demand and supply centers.

An important facet of implementing SOA using INSOAP is the services identification and matching process as depicted in Figure 7. The demand centers provide use cases associated with the business process. This, along with the applications and databases associated with the business process, is used by the supply center to identify composite and fine-grained services that need to be provisioned. This is the top-down mapping of business process to services. If there are legacy or existing applications in the enterprise, the supply center also performs a bottom-up mining for services from them. The demand and supply centers then match the set

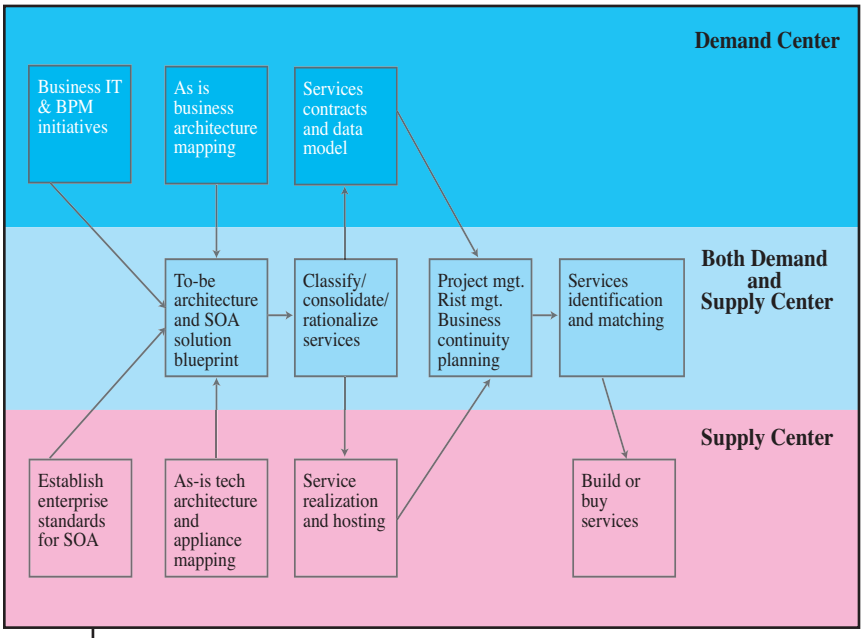


FIGURE 6 **Implementing SOA**

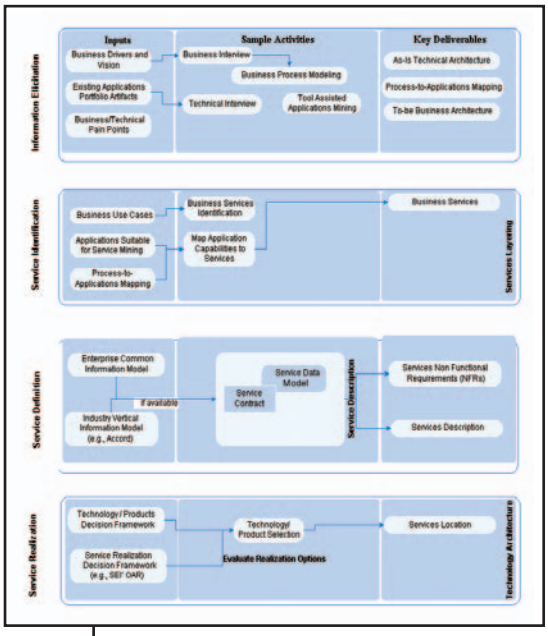


FIGURE 7 **Services identification and matching**

of services obtained from both the top-down mapping and bottom-up mining for services to arrive at a set of services that is required by the business units. This is the basis for the consolidation and rationalization of services. The supply center uses the output of the rationalization exercise to develop and realize the service required by the business units. The service realization decision is taken by the supply center based on a modeling of business drivers, pain points, and cost-benefit analysis with inputs from the demand centers. The evaluation of business impact of services realized and non-functional requirements implications are jointly performed by the demand and supply centers. The output of the rationalization exercise is also the basis for the supply center to take decisions on whether it will buy or build the services and retiring applications. Once the services are provisioned either in house or through third-party providers, governance mechanisms for operating SOA and supporting the business users need to be in place.

Operating SOA

The key processes of operating SOA based on INDIGO are supporting users using services, SLA, and QoS management, as shown in Figure 8. In SOA support, the supply will be responsible for setting up the infrastructure for technical and functional helpdesks. Usually enterprises follow the ITIL (IT Infrastructure Library) guidelines for managing helpdesk and support operations in the traditional IT applications context. We believe that similar guidelines can be followed in the SOA context too.

While the technical helpdesk and support is a responsibility of the supply center, the functional helpdesk and support is the responsibility of the demand centers. Typically enterprises would like three of four levels of support. The first level is a helpdesk for resolving simple issues, the second and third levels involve a more specialized support group that resolves more complex issues, and the fourth level is for SOA infrastructure and making enhancements to services. Both technical and functional helpdesk and support will leverage workflow tools and database systems that help categorize, resolve, and log escalating requests.

SLA and QoS management is critical to ensure that the users of services in the business units are satisfied with the services. The demand center and the supply center jointly decide on the planning process for provisioning new services. They also establish the processes to manage SLAs that include financial aspects, availability/continuity, QoS, incident resolution, and security aspects of the services provided. If a third-party vendor is providing the service, the supply center also establishes processes for vendor management to ensure QoS as per SLAs. The supply center also publishes processes for configuration, change, and release management that are acceptable to the demand center. A significant part of this might involve the infrastructure to manage SOA policies, using a combination of policy repositories, registries, and policy management infrastructures. Traditionally IT organizations have been leveraging guidelines of BS1500 and ITIL for SLA and QoS management.

Identifying Roles and Balancing Expectations

The reorganization of demand and supply centers can be tricky because traditionally senior executives in business units and the central IT function have different perspectives that are often not in sync with one another. While the business unit wants value from IT, the IT function more often than not focuses on reducing cost of services. Enterprises will need to establish a clear business case for the demand and supply centers and their objectives. Specific roles and responsibilities charting and reporting structures need to be prepared and communicated before attempting the reorganization accompanying a shift to SOA. In large enterprises this can take up to 12 months for the transition.

New Roles Envisaged

Some of the important managerial roles we envisage in the demand and supply centers are:

- General Manager, Business Services (in the demand centers) who can straddle both business process and IT domains to spearhead projects on BPM and service conceptualization and liaise with senior management in the business units and supply center
- Manager, Business Services (in the demand centers) who is responsible for mapping processes into services, services modeling, user evaluation, operating the functional helpdesk, and liaising with counterparts in the supply center
- General Manager, Services Architecture (in the supply center) who is involved in setting the enterprise standards for SOA, services conceptualization, service portfolio assessment and rationalization, maintaining services repository, and liaising with counterparts in demand centers
- Services Implementation Manager (in the supply center) who is responsible for making decisions on building/buying/reusing services and implementing them, program management for implementing services, identifying KPIs for successful implementation, and liaising with counterparts in demand centers
- Services Operations Manager (in the supply center) who is responsible for technical helpdesk operations, SLA management and QoS, and liaising with counterparts in demand centers

The Balancing Act

Having a demand center in each business unit can be expensive for smaller enterprises and can sometimes be ineffective in certain business units. A floating demand center would be the preferred model in such cases.

In some cases there is a possibility that the demand centers may behave as the business units do and develop rigidities that may weaken the role of the supply center. These demand centers may want to bypass the service center and shop for services directly from vendors. Not only should the supply center have the capability to convince beyond a doubt that they are the best value service providers for the business units, but it should also have the support of the senior executives in the enterprise in their role. Often the supply center will have questions raised on the transfer price charged for the services delivered and the mechanisms for calculating the transfer price by the demand center. The calculation of the transfer pricing for services should be transparent and the rules for apportionment among different business units be should perceived as fair.

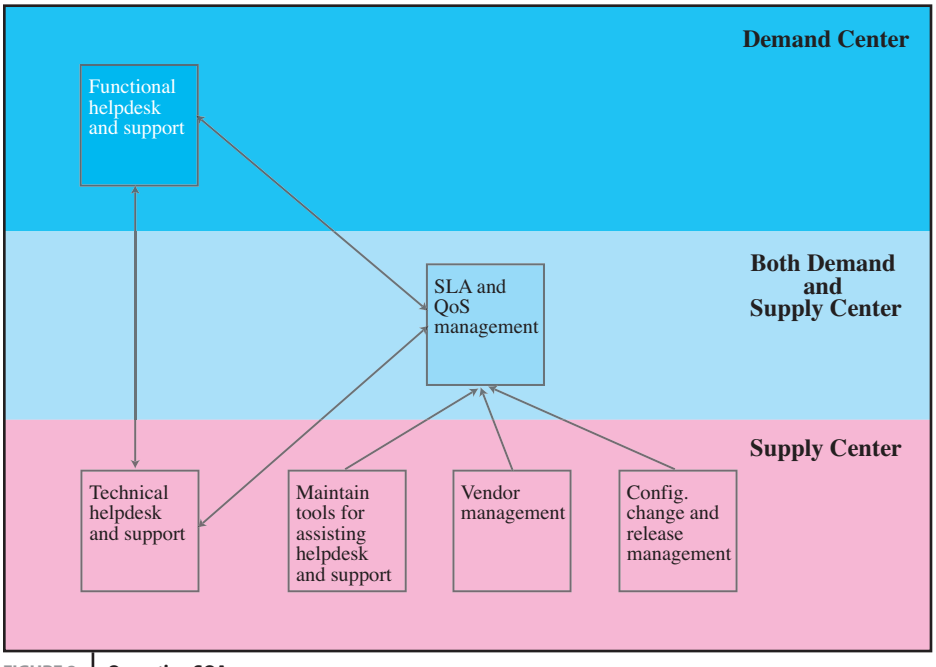


FIGURE 8 | **Operating SOA**

We believe that bifurcating the IT function into demand and supply centers enhances accountability to the business unit, leverages scale and scope economies for delivering services, and prescribes clear roles and responsibilities for governance in the SOA context. The transition to demand and supply centers needs to be managed carefully, keeping in mind the governance guidelines and the roles and responsibilities bifurcation.

References

- CobiT, 3rd Edition, Executive Summary, CobiT Steering Committee and the IT Governance Institute, July 2000.
- Peter Weill, "Don't Just Lead, Govern: How Top-performing Firms Govern IT," MIS Quarterly Executive (3:1), March 2004.
- Abdelkarim Erradi, Sriram Anand, and Naveen Kulkarni, SOAF: "An Architectural Framework for Service Definition and Realization," submitted to International Conference on Software Engineering 2006 Shanghai (China), September 2005.

About the Authors

N. Dayasindhu, PhD, is a senior research associate at the Software Engineering Technology Labs, Infosys Technologies. His research helps IT organizations align better with business functions. He has published in peer-reviewed journals and conferences and has consulted for Fortune 500 enterprises.
dayasindhun@infosys.com

Dr. Srinivas Padmanabhuni is a principal researcher with the Web Services Centre of Excellence in SETLabs, Infosys Technologies, and specializes in Web Services, service-oriented architecture, and grid technologies alongside pursuing interests in Semantic Web, intelligent agents, and enterprise architecture. He has authored several papers in international conferences. Dr. Padmanabhuni holds a PhD degree in computing science from University of Alberta, Edmonton, Canada.
srinivas_p@infosys.com

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

Modeling Web Services Choreography

Dancing with BPMN and new Eclipse tool pi4soa

Choreography is the dark continent of Web services: few onlookers have traveled there, and many question whether there are any riches to be brought home from the trip. In the first place, choreographies bear such a striking resemblance to business processes that the novice might think that the two types of artifacts are indistinguishable.

WRITTEN BY
MICHAEL HAVEY

After all, isn't choreography just a way to describe what a business process does (i.e., it *choreographs* the actions of its participants)? And then there is the dearth of choreography tools; until recently choreography was a topic learned by reading, not by hands-on experimentation. This article takes the trip. It describes how, in the ideal set of design tools, not only are choreographies and processes treated as entirely different artifacts (with different development life cycles), but that special modeling techniques are available to fully accentuate the nuances of choreographies.

Choreography and Process

Contrary to what many Web services and business process people think, it is terminologically incorrect to say that a process *choreographs* its services. *Choreography* describes the global protocol that governs how individual processes interact with one another. Each process offers its own services and uses services of partner processes. It is correct to say that a process *orchestrates* these services, but the view from one process is only the behavior of that process in terms of its partners. Choreography presents the unified global view, depicting all of the processes and their required interactions.

Web Services Choreography Description Language (WS-CDL) is the leading choreography language, and Business Process Execution Language (BPEL) is the dominant process orchestration language. Though both XML-based languages feature a similar flow-oriented design style, only BPEL is meant to have an actual run-time platform: BPEL processes run, and WS-CDL choreographies are formal specifications documenting rules to guide interprocess exchange. There are no traffic cops in this *laissez faire* world, only traffic laws and law-abiding drivers.

Figure 1 shows the development life cycle for both choreographies and processes. In part (a) of the figure, the work to build a choreography begins with the gathering of requirements from representative participants, whereupon a software designer, using a business process modeling tool, draws the choreography in a notation language, preferably Business Process Modeling Notation (BPMN) or UML. The tool generates from the diagram WS-CDL XML code, which in turn is input to a choreography code editor, such as pi4soa (discussed in detail shortly), which enables a software developer to refine the choreography into a form that is suitable for rigorous testing. A good way to test the choreography is to create stub processes (preferably in BPEL) that represent each participant, and have these processes exchange messages with each other. An endpoint monitor watches the message traffic and checks for compliance to the choreography. When testing completes, the WS-CDL choreography and its reference BPEL stubs are ready for release.

Part (b) shows the life cycle for a particular participant process that intends to follow the choreography. The software designer bases the formal process design on the choreography itself as well as requirements specific to the participant organization. As in (a), the modeling tool should support BPMN or UML and be able to export BPEL code (as a bonus, it should also be able to *import* the BPEL stubs provided with the choreography), which can then be fed into a BPEL code-level editor, where the process can be refined and be made test-worthy. The test cycle requires a BPEL platform and scripts to test for both private and public requirements, and the endpoint monitor introduced in (a) can be used to verify choreography compliance.

This article focuses on two parts of the

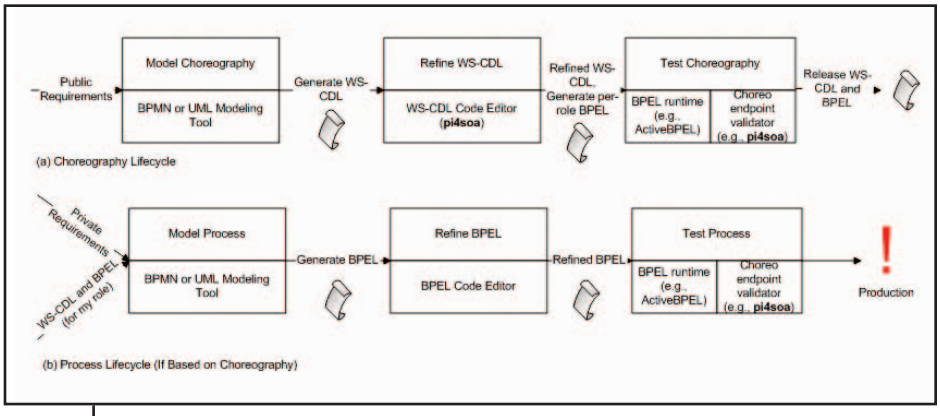


FIGURE 1 | **Choreography and process tools and life cycles**

choreography cycle: modeling and code refinement. In the modeling area, there are plenty of good business process tools supporting UML and BPMN from which to choose, but none of them can generate WS-CDL output directly. Many can export models in a canonical form (e.g., XML metamodel interchange, or XMI), but there are no third-party tools that can generate WS-CDL code from that form. An open-source version of the proposed code editor, to our delight, is now available in alpha form. The tool, known as Pi Calculus for Service-Oriented Architecture (pi4soa, developed by the company Pi 4 Technologies), is an Eclipse plugin that provides a graphical editor to compose WS-CDL choreographies and generate from them compliant BPEL.

Example: Open Energy Market

To demonstrate the tools described above, we will model and then manipulate the WS-CDL source code of a simple energy market choreography, which describes how, in a competitive, deregulated market, retailers work with a single distributor to enroll customers. The energy example is inspired by the Ontario Energy Board Electronic Business Transaction standard (good examples come from the most improbable places), which was published in 2001 before choreography came along, but was influenced by the B2B ideas of the day. The standard uses UML-like collaboration diagrams, rather than choreography, to depict message exchanges between customers, retailers, and distributors.

The rules of the choreography can be stated in English as follows:

- When a customer decides to enroll with a retailer, the retailer sends an enrollment request message to the distributor to sign up the customer.
- If there is a problem with the enrollment, the distributor sends an enrollment reject message to the retailer. Problems include:
  - The customer is already enrolled with that retailer
  - The customer has an enrollment in progress with another retailer
  - There is a problem with the customer's account
- Otherwise, if the customer is not currently enrolled with a retailer, the distributor sends an enrollment accept message to the retailer. The customer now has 10 business days to cancel the enrollment. If the customer decides to cancel, the retailer sends a cancellation message to the distributor; otherwise, at the end of the 10 days, the distributor sends an enrollment complete message to the retailer. The customer is now enrolled with the retailer.
- Otherwise, if the customer is currently enrolled with a different retailer (i.e., the "current" retailer), the distributor notifies both retailers by sending to each a pending switch message. The customer now has 10 business days to cancel the switch. If the customer decides to cancel, the new retailer sends a cancellation message to the distributor, who sends a message to the current retailer indicating that the switch has been cancelled. Otherwise, at the end of the 10 days, the distributor sends to both retailers a message indicating that switch has completed, and the customer is now enrolled with the new retailer.

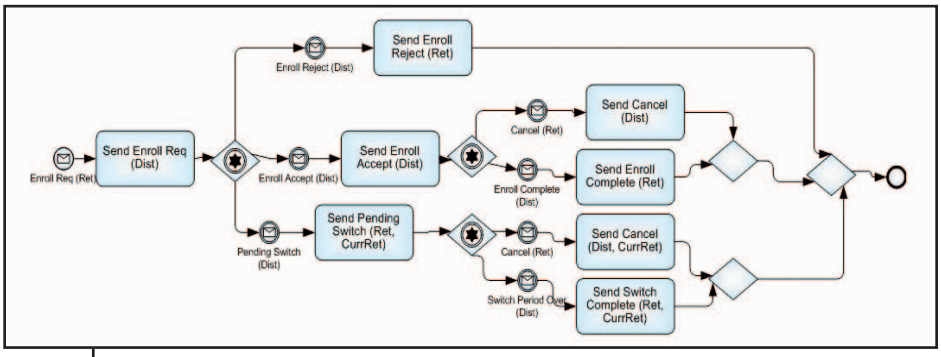


FIGURE 2 **Imaginary hub**

The Choreography Model

In BPMN, two possible ways to model choreography are:

- ***Imaginary Hub:*** Although choreography is fundamentally decentralized, we imagine there is a central hub through which all messages pass, and model the choreography as the process of that hub.
- ***Sum of Parts:*** The process of each participant is drawn in a separate pool. The steps in each process are the public steps – the ones required by the choreography. Message flow depicts the passing of messages from a step in the process of one participant to the step in a process of another.

Figure 2 shows the imaginary hub process, which oversees all messages exchanged among retailer, current retailer, and distributor participants. There are two types of interactions in this process:

- ***Receives*** (circles with enclosed mail envelopes), representing messages sent by the participant indicated in parentheses
- ***Sends*** (rounded boxes), representing messages sent to the participant or participants indicated in parentheses

The diagram also makes use of the exclusive-OR event gateway (delimited by a diamond with an enclosed star and a plain diamond), which waits for exactly one of multiple messages to be received, and executes the logic on the path for that event.

The logic resembles the English description above, identical but for the inclusion of the hub. When an enrollment request is received from the retailer, the hub sends it to the distributor, and then waits for the one of three events to arrive from the distributor: an enrollment reject, an enrollment accept, or a pending switch notification. In each case, the hub forwards the message to the retailer and, in the pending switch case, to the current retailer too. In the accept and switch cases, the hub waits for either a cancel from the retailer (which it routes to the distributor, and, for a switch, to the current retailer) or a notice of completion (i.e., an enrollment complete or switch period over event) from the distributor, which the hub forwards to the retailer (and current retailer for a switch).

As we will discover in the next section, the imaginary hub approach maps nicely to the WS-CDL representation of the choreography.

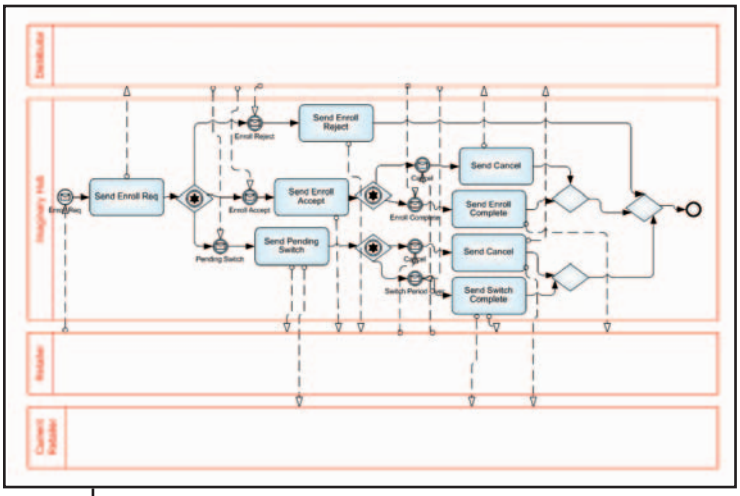


FIGURE 3 | **Imaginary hub with pools**

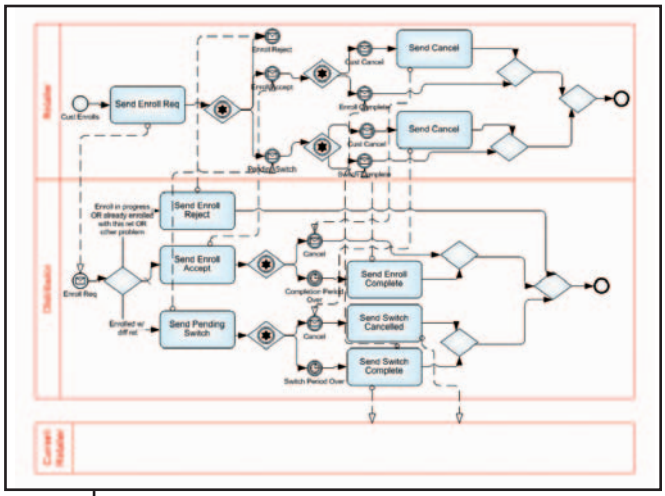


FIGURE 4 | **Sum of parts**

Figure 3 shows an alternative (and inferior) representation of the imaginary hub, in which the participants are represented in separate pools (long narrow rectangles), with message flow (the dotted arrows) between them depicting the interchange of messages. Although the pools are a nice visualization of the participants, the message flow creates an indecipherable clutter that makes the diagram almost impossible to read.

The sum-of-parts approach is shown in Figure 4. As the name suggests, choreography represented this way is, in a sense, the sum of the public processes of the individual participants and the exchange of messages that drives them. This approach is suitable if there is no requirement to evaluate the sum! In Web Services Choreography Interface (WSCI, an earlier choreography specification and a precursor to WS-CDL), for example, choreography is defined as the public behavior of each participant plus the global message exchange view. Figure 4 is a convenient representation of a set of WSCI interfaces. WS-CDL, by contrast, requires a single, global, unified definition, which would be hard to synthesize from the mess of process steps in the retailer and distributor pools in the figure. We are accustomed to watching dancers in a show dance as a group. What if each dancer performed separately, and we were left to combine them in thought as a group?

In UML, activity diagrams or collaboration diagrams could be used to build the imaginary hub and sum-of-parts approaches.

Code-Level Choreography with pi4soa

A software developer using pi4soa, the new WS-CDL code editor, can either create a WS-CDL choreography from scratch or import an existing one for modification. Although the import option fits our life cycle best, no tools exist to generate WS-CDL from a model (such as the one designed in the previous section), and hence the developer is left with the less desirable option of building a fresh WS-CDL while eyeballing the model.

The choreography managed by pi4soa is a WS-CDL file in a Java project in Eclipse. The pi4soa plugin provides a graphical editor that, behind the scenes, generates WS-CDL XML code (which can be viewed in a text editor). Figure 5 shows the pi4soa editor (the large window on the right side) open in the "base types" view, which allows the developer to assemble the main building blocks of the choreography. The "participants" view, shown in Figure 6, allows the developer to build structural participant relationships.

The key information in these views is the following:

- The participants (or roles) involved in the choreography are Distributor, Retailer, and CurrentRetailer, as well as two participants meant to simplify the implementation (for reasons discussed below): Customer (to model a customer's interaction with a retailer) and DistributorBizCal (a subsystem of the distributor to model the management of business calendars for completion and switch periods).
- Each participant (or role) has a behavior and a channel. The behavior is the participant's Web service, and the channel is its inbound communication endpoint. All interactions are asynchronous. To send a message to a retailer, for example, call its Web service by placing a message on its channel.
- There are four relationships. RD represents the interface between Retailer and Distributor, CRD the interface between CurrentRetailer and Distributor, RC the interface between Retailer and Customer, and DInt the interface between Distributor and DistributorBizCal. The relationships allow communication in both directions; RD, for example, lets Retailer call Distributor, or vice versa.
- The message type (or information type) exchanged between participants is an XML document called EnergyMsg, which contains fields (or tokens) such as custID, retailer, txID, currentRetailer, and reason.

In the third view ("choreography flows"), the developer uses these building blocks to construct the control flow of the choreography. The overall structure is shown in Figure 7. The choreography begins with the "interaction" (i.e., message exchange) *enrollReqFromCust*, in which the customer sends an enrollment request message to the retailer. The retailer then forwards that message to the distributor in the interaction *enrollReq*, and the distributor executes a "private" (i.e., placeholder) step

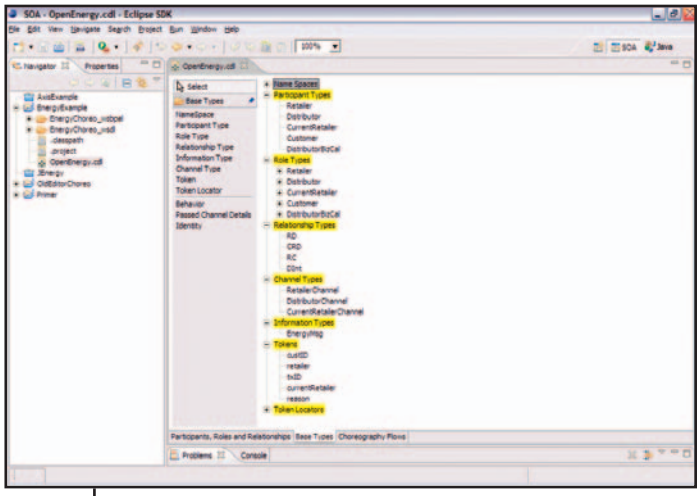


FIGURE 5 | **pi4soa editor in base types view**

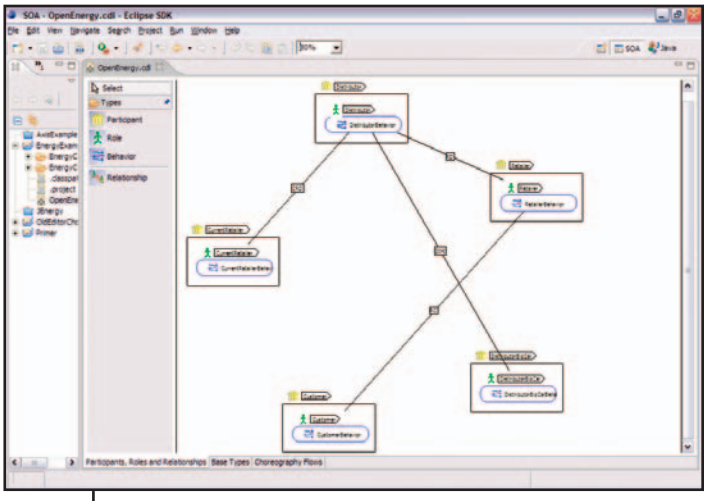


FIGURE 6 | **pi4soa role view**

determineEnrollmentStatus to decide whether to accept the request, reject it, or initiate a switch.

The heart of the choreography is the "choice" structure *enrollmentResult*, which allows exactly one of its paths to run based on the enrollment status determined in the previous private step. The simplest of the three paths, for rejection, is the "sequence" shown in Figure 8, in which the distributor, in a private step, adds the reject reason to the message (the WS-CDL "assign" activity could also have been used for this), and then, in the interaction *enrollRejected*, sends the message to the retailer.

The acceptance sequence in Figure 9 is more complicated. The distributor begins by sending an acceptance message to the retailer in the interaction *enrollAccepted*, and then, in the private step *setCompletionTimer*, sets an alarm to go off at the end of the completion period. Next, one of two events can occur: the completion timer expires, or the customer cancels the enrollment with the retailer. These options are modeled as sequences *periodExpired* and *cancel* in the choice structure *completionPeriod*. The logic of each sequence is straightforward: in *periodExpired*, DistributorBizCal sends an alarm notice to the distributor (*periodExpired*) and the distributor, in turn, sends a completion event (*enrollmentComplete*) to the retailer; in *cancel*, the customer sends a cancel event to the retailer (*cancelFromCust*), which the retailer forwards to the distributor (*cancel*).

The switch path, shown in Figure 10, is similar to the acceptance path. The distributor begins by sending a switch pending message to both the retailer and the current retailer (in the pair of interactions named *switchPending*), having already, in private steps, added the identity of the CurrentRetailer to the message and set the switch business calendar timer. If the timer expires (in the *periodExpired* interaction), the distributor sends completion messages to both retailers (*switchCompleted*); if the customer cancels (*cancelFromCust* from customer to retailer, *cancel* from retailer to distributor), the distributor sends a notification to the current retailer (*switchCancelled*).

From Model to WS-CDL to BPEL

The WS-CDL choreography closely resembles the "imaginary hub" process shown in Figure 2 (described in the previous section). In particular:

- An interaction activity in the WS-CDL choreography is equivalent to a receive event followed by a send activity in the hub's process. In both cases, the effect is to move a message from one participant to another.
- The choice control structure in the WS-CDL is equivalent to the exclusive-OR event gateway in the hub's process. In both cases, the effect is to execute exactly one of several paths.

FIGURE 7 | **pi4soa overall choreography**

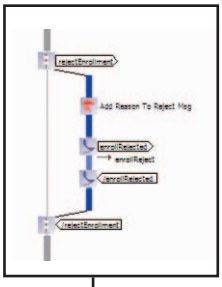


FIGURE 8 | **Reject path**

One glaring difference is the introduction in the WS-CDL choreography of the Customer and Dis-

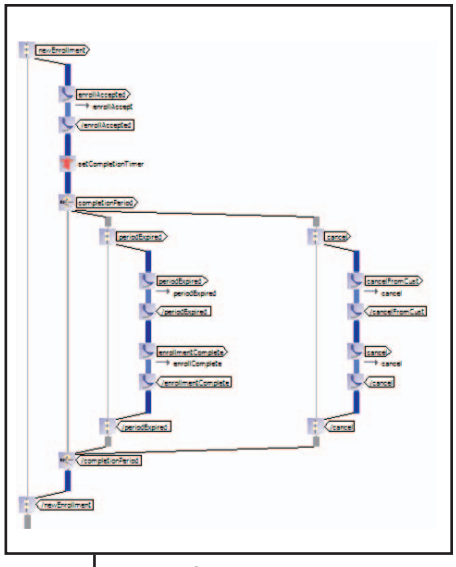


FIGURE 9 | **Accept path**

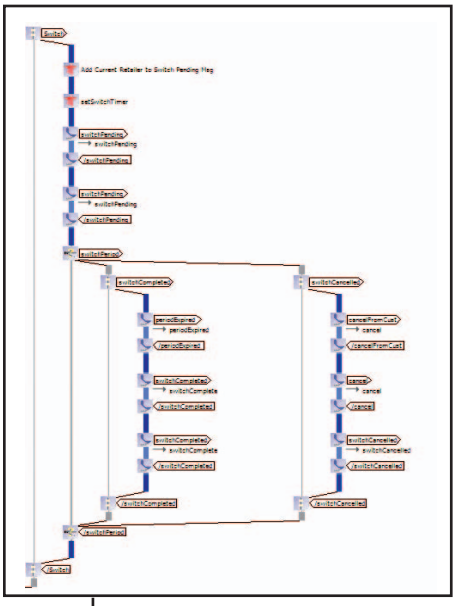


FIGURE 10 **Switch path**

tributorBizCal participants. The purpose is simplified implementation; specifically, to use event choice rather than exception handling to handle the race between period expiry and customer cancellation in the logic for enrollment acceptance and switch. Both outcomes are represented as events, and the first one to occur wins. The onerous alternative is to wait for the cancellation event for a specified duration, and handle the period expiry as a timeout exception.

The event choice approach also makes the individual participant process stubs easier to understand. For the retailer, once its enrollment request has been accepted by the distributor, it waits for either an enrollment completion message from the distributor or a cancellation event from the customer. For the distributor, once it has accepted the enrolment, it waits for either a period expiry notification from its business calendar system (an internal event) or the cancel event from the retailer.

The BPEL code generated by pi4soa from the energy choreography uses "pick" activities to model the event choice. Listing 1 shows, in pseudo code, the retailer's BPEL stub. There are two levels of event choice: the outer pick to determine whether the request was accepted, rejected, or caused a switch (lines 3-17), and the inner picks to manage the periods (lines 8-11 and 14-17).

The development life cycle spanning the modeling of the BPMN imaginary hub, the refinement of the WS-CDL code, and the generation of the stub BPEL process works. Viable participant processes can be derived from properly modeled, implemented, and tested choreographies.

Listing 1: Retailer BPEL Pseudo Code

```
01 receive "enrollRequest" from customer
02 invoke "enrollRequest" on distributor
03 pick
04    onMessage "enrollRejected" from distributor
05       empty -- placeholder
06    onMessage "enrollAccepted" from distributor
07       pick
08          onMessage "enrollCompleted" from distributor
09             empty -- placeholder
10          onMessage "cancel" from customer
11             invoke "cancel" on distributor
12    onMessage "switchPending" from distributor
13       pick
14          onMessage "switchCompleted" from distributor
15             empty -- placeholder
16          onMessage "cancel" from customer
17             invoke "cancel" on distributor
```

References

- W3C Web Services Choreography Working Group, "Web Services Choreography Description Language Version 1.0" (November 2005). www.w3.org/TR/2005/CR-ws-cdl-10-20051109/
- BPMI, "Business Process Modeling Notation 1.0 Specification" (May 2004). www.bpmn.org/Documents/BPMN%20V1-0%20May%203%202004.pdf
- *pi4soa sample apps:* www.pi4tech.com
- *Official pi4soa site:* http://sourceforge.net/projects/pi4soa
- *To run pi4soa, you need several Eclipse components (see pi4soa release notes for details):* www.eclipse.org
- Havey, M. *Essential Business Process Modeling*. O'Reilly, 2005. Discusses BPMN and choreography.
- Ontario Energy Board, "Ontario Electronic Business Transactions Standards Document for Retail Settlement in the Electric Retail Open Access Industry" (December 2001). www.xmlenergy.net/oeb_ebt_wg/docs/standards/EBTStdsDocumentDecember21.doc

About the Author

Michael Havey is an IBM consultant with 10 years of industry experience, mostly with application integration. Michael's book Essential Business Process Modeling was published by O'Reilly in August 2005.

haveym@ca.ibm.com

XML JOURNAL SUPPLEMENT

This Month

Out of Step: NIEM and N-DEx

Neil Kurlander

Since the horrific events of September 11, 2001, the federal government has intensified its efforts to improve communications, collaboration, and information sharing between government and private sector agencies at all levels. The task of creating a seamless system of data and communication between disparate agencies has faced both technological and political obstacles.

Java Programming with Berkeley DB XML

Selim Mimaroglu

Berkeley DB XML (BDB XML) is a popular native XML database. It can be accessed through the shell or within another program. This month I will show you how to use BDB XML in Java. BDB XML has similar APIs for all supported languages such as Java and C++, therefore the ideas presented in this article apply to all supported languages. I have been closely following BDB XML from the very first release, and there have been tremendous improvements in this product.

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

Out of Step: NIEM and N-DEx

Two national data-sharing initiatives face major challenges

WRITTEN BY
Neil Kurlander

Since the horrific events of September 11, 2001, the federal government has intensified its efforts to improve communications, collaboration, and information sharing between government and private sector agencies at all levels. The task of creating a seamless system of data and communication between disparate agencies has faced both technological and political obstacles.

This article will look at two federal data-sharing initiatives: the National Information Exchange Model (NIEM) and the National Data Exchange (N-DEx). Each offers new information-sharing capabilities and each faces major challenges.

Is universal sharing of information between governmental agencies actually doable? NIEM is an ambitious new initiative that is taking a giant step towards making the dream of government-wide data interoperability a reality. NIEM is a joint effort of the Departments of Homeland Security and Justice. Its mission is to create the means to seamlessly exchange information electronically between multiple governmental agencies.

The task of bridging hundreds of formerly stove-piped systems is enormous. Every system has its own way of formatting data and defining the meaning of database terms. For instance, one system may use the term "FirstName" and another the term "FName" to specify a person's first name in a database. In other cases, different systems may use an identical label to represent different data. "CNum" may mean "case number" in one system and "catalog number" in another.

"Currently, there is no national system for automatically sharing investigative data from law enforcement records-management software"

Clearly the development of common standards is central to the success of NIEM. To bridge the differences in business needs and data standards between domains, NIEM will build upon the commonalities of data elements to create Common Core and Universal Core Information Exchange Packages (IEPs). Domain-specific exchange standards will also be developed to address intradomain characteristics.

The National Information Exchange Model – Following Successful Footsteps

Shared Extensible Markup Language (XML) is a technology standard that allows information and services to be encoded with meaningful structure and semantics that facilitate information exchange. The Department of Justice's (DOJ) Infrastructure and Standards Working Group chose XML as the open standard for data exchanges in early 2002.

The Global Justice Information Sharing Initiative (Global) was then founded to create a Justice-specific implementation of XML. Global, a consortium of organizations that are members of the justice community, released The Global Justice XML Data Model (GJXDM) in February 2004.

One year later in February 2005, Steve Cooper, the CIO of the Department of Homeland Security (DHS), and Vance Hitch, the CIO of DOJ, jointly announced that both agencies had selected the GJXDM as the standard for data sharing and interoperability by their agencies and that a new federal initiative, the National Information Model Exchange Model, would be created.

NIEM will be designed to bridge the divides between Justice, Health, Transportation, Intelligence, and other domains. It will build upon the successful implementation of the Global Justice XML Data Model (GJXML), which has already provided the means to tie together data between all federal, state, local, and tribal agencies in the justice community.

One major objective the NIEM is to prevent the creation of XML information silos. While GJXDM has become the standard for interoperability in the justice domain, other domains such as homeland security, emergency management, health and human services, and transportation each have unique business rules and information-sharing standards that are domain-specific. To prevent domain silos, NIEM will provide the framework, architecture, security, and metadata controls necessary to assure that interoperability between domains will not be compromised. It will also develop domain-data dictionaries and schemas to meet the business needs of each community.

NIEM's success is dependent upon continued political support and funding. It also requires the acceptance of a new sharing paradigm among federal agencies and among federal, state, and local entities. Its foundation is based upon years of work by a large number of organizations that have established credentials and that support the program. It is built upon the successful implementation of the GJXDM and its future offers a major step forward in cross-domain electronic data sharing. For all of these reasons the National Information Exchange Model has a high probability of success. The same cannot be said for the National Data Exchange.

The National Data Exchange – Trying to Find Its Footing

The Department of Justice created the Law Enforcement Information Sharing Program (LEISP) in response to a presidential order from August of 2004 requiring federal law enforcement and domestic security agencies to cooperate more closely with state and local police. In response, the Federal Bureau of Investigation (FBI) developed its National Information Sharing Strategy (NISS). The National Data Exchange (N-DEx) and Regional Data Exchanges (R-DEx) are two NISS initiatives by the FBI to improve the sharing of information among law enforcement agencies.

"To many in the law enforcement community, the sharing of regional criminal data is of more value than a national system"

Information concerning criminal suspects including, their method of operation, the vehicles they use, their accomplices, phone numbers, addresses, weapons used, and many additional factors are shared locally and in some cases regionally, but currently there is no national system for automatically sharing investigative data from law enforcement records-management software.

N-DEx Will Be a Criminal Investigative Sharing System

N-DEx has been in pilot testing since June 2004 with the Uniform Crime Reporting (UCR) State Repository, West Virginia State Police, the Cabell County, West Virginia, Sheriff's Office, and police departments in Marietta, Georgia, and Alexandria, Virginia. Based on the lessons learned from the pilot sites and input from focus groups and national law enforcement organizations, the Bureau is currently completing its Concept of Operations Plan, which will guide the implementation process.

N-DEx will correlate data from all major FBI databases and provide the ability to execute nationwide inquiries from a single access point. It offers for the first time a national system that can be used to identify crime trends, link cases, and solve crimes.

In 2005 the FBI implemented two Regional Data Exchanges (R-DEx), the first in St. Louis and the second in Seattle. R-DEx will give state, local, and tribal law enforcement appropriate access to federal investigative and intelligence information on a regional basis. The FBI, US Marshals, and the Bureau of Prisons are currently sharing investigative reports with the local participants. The DEA and ATF are also expected to add information from their databases to the system shortly. Twelve to 18 additional R-DEx projects are currently being planned for implementation over the next several years.

To many in the law enforcement community, the sharing of regional criminal data is of more value than a national system. This is due to the fact that the overwhelming majority of criminal investigations are local or regional in scope. Programs such JNET, ARGUS, and FINDERS have demonstrated the value of such regional approaches.

N-DEx Will Be an Incident-Based Collection Instrument Used for Crime Reporting

In addition to serving as a criminal investigation data-sharing tool, N-DEx has two additional major objectives: the electronic reporting of crimes and reporting of crime statistics to the FBI.

State and local agencies have been reporting monthly

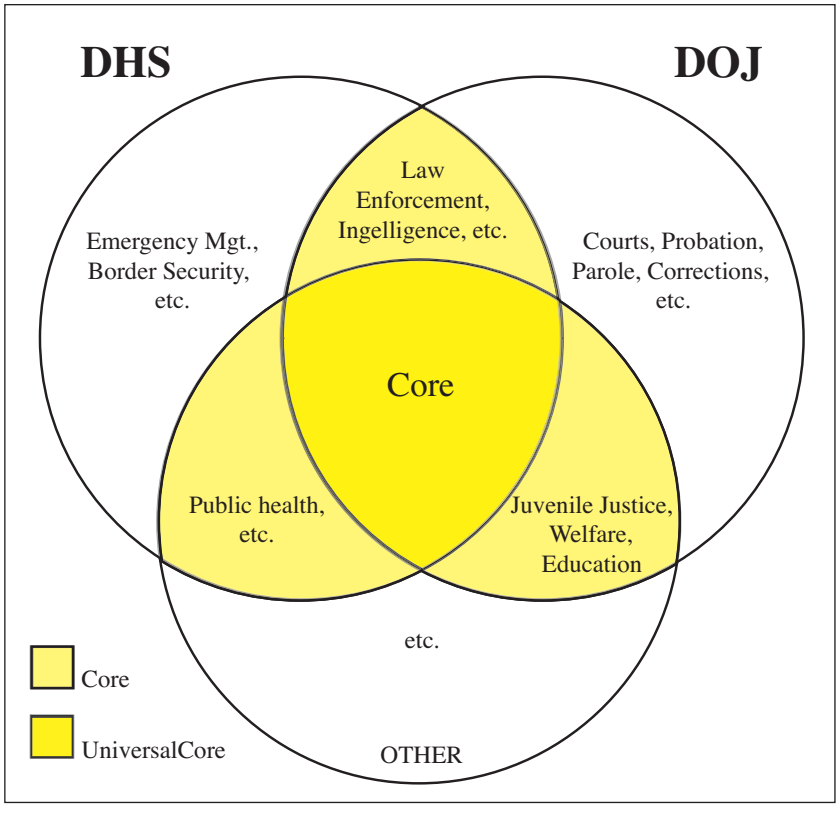


Figure 1 • NIEM's domain data core concept

counts of key offenses and arrests to the FBI through the Uniform Crime Reporting (UCR) Program since 1929. This program was greatly expanded in 1992 through the National Incident-Based Reporting Systems (NIBRS), which collects more details on more categories of crime such as hate crime, sexual assault among young victims, and kidnapping of juveniles.

Both programs are still used for the collection of crime statistics. The information that is reported to the FBI under these programs is collected from data that is contained within police reports. Most computerized law enforcement records-management systems produce UCR or NIBRS reports electronically. The reports are then sent via paper or electronically to the FBI for publication.

As a national program, NIBRS is widely regarded as a failed effort. One measure of its failure is that by 2004, 14 years after its introduction, only 23 states were fully or partially participating in submitting data. The proposal to include crime reporting as part of the N-DEx program has meet with confusion and concern by local law enforcement agencies. They want to know if N-DEx will replace UCR and NIBRS, and who is going to pay for the cost for changing their software to provide the new functionality.

> "N-DEx, if properly implemented, would be a major step forward in improving information sharing among all levels of law enforcement"

The N-DEx program has used a consensus-building process called the National Consensus Process (NCP) to help define the scope of the program. Law enforcement and other criminal justice entities representing both large and small agencies have participated in the consensus process as members of focus groups. Several focus group meetings have been held to gather input and build consensus with regard to the definition of the N-DEx program. However, while individual focus groups are important in defining the program's objectives, the input and approval of the national law enforcement associations must be sought before the political support necessary to carry this project forward is obtained.

The failure to actively engage these organizations earlier in the process has caused the project to be viewed with concern by these powerful national associations. To its credit, the FBI's CJIS staff has recently meet with both national law enforcement and industry groups to allow for their input prior to finalizing the concept of an operations document.

This document must set forth an unambiguous message to the law enforcement community about what N-DEx will and will not include if the Bureau wishes to prevent further confusion. Questions concerning the future of the Uniform Crime Reporting System (UCR) and the National Incident-Based Reporting System (NIBRS) need to be answered, and cost projections for the program's implementation need to be provided. Lack of clarity on the future and interoperability among N-DEx, R-DEx, and NIEM should also be addressed.

A joint Position Statement dated August 15, 2005 was released by the International Association of Chiefs of Police, the National Sheriffs Association, the Major Cities Chiefs Association, and the Major County Sheriff's Association concerning the future of N-DEx.

While the associations indicated that they support the ability of law enforcement to share information between all levels of government, they stressed that the project's success is dependent upon the voluntary participation of the agencies. To achieve that support, the statement indicated that the N-DEx Program needed to ensure that law enforcement practitioners at every level participate in the system design, testing, and implementation.

The statement set forth three requirements for success:

- Development of a Statement of Requirements designed with local law enforcement input and utilization in mind. It recommends that the requirements should be validated through representatives of their associations and coordinated by the FBI CJIS Advisory Board (APB).
- After an agreed upon Statement of Requirements is completed, a Funding Projections document that reflects the costs for each phase of the project to the agencies needs to be prepared.
- Based upon the Statement of Requirements and Funding Projections, the FBI and the DOJ need to formalize a process that will result in a consistent message about the "Project's mission, goals, strategy, and status." The statement also indicated that the roles of each agency need to be documented to facilitate, "by-in" by all levels of law enforcement.

It is clear that the intent of the Position Statement is to warn the Bureau and the DOJ that without the support of local law enforcement, N-DEx will not be successful.

N-DEx, if properly implemented, would be a major step forward in improving information sharing among all levels of law enforcement. Its use as a tool for replacing and upgrading UCR and NIBRS is problematic and raises many issues concerning cost and participation. The process for designing and marketing it to the law enforcement community has been uneven, and its value undersold. The Bureau will need to win the support of the national law enforcement agencies and through them, the law enforcement community. Gaining their support will require the FBI to work in close collaboration with law enforcement in the design and implementation of the program. It also will need to work with the public safety software industry to build a cost-effective solution.

Author Bio

Neil Kurlander is the vice president of Public Sector Solutions for Asynchrony Solutions (www.asolutions.com). His dual careers in public safety and technology span four decades. His advocacy for the use of technology by law enforcement resulted in his appointment as chair of technology committees on the national, state, and local levels. Neil is a life member of the International Association of Chiefs of Police and currently serves on the association's Communication and Technology Committee. He also chairs the Law Enforcement Information Technology Standards Advisory Committee of the Integrated Justice Information Systems Institute.

neil.kurlander@asolutions.com

ENGAGE AND EXPLORE...

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

CALL FOR PAPERS NOW OPEN!

SOA WebServices Edge 10th International conference+expo 06

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

Suggested topics...

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

CALL FOR PAPERS NOW OPEN!

2006 ENTERPRISE> OPENSOURCE CONFERENCE+EXPO

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

Suggested topics...

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

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

Sponsored by

WebServices JOURNAL · XML JOURNAL · .NET DEVELOPER'S JOURNAL · eclipse developer's journal · WebSphere JOURNAL · information STORAGE+SECURITY journal · wldj · JDJ · LINUXWORLD MAGAZINE · MX developer's journal · asp.netPRO · SD Times · CoDe · Software Test & Performance

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

Attention Exhibitors:

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

For Exhibit and Sponsorship Information – Call 201 802-3023

Produced by SYS-CON EVENTS

WRITTEN BY **SELIM MIMAROGLU**

Java Programming with Berkeley DB XML

Explained through examples

Berkeley DB XML (BDB XML) is a popular native XML database. It can be accessed through the shell or within another program. This month I will show you how to use BDB XML in Java. BDB XML has similar APIs for all supported languages such as Java and C++, therefore the ideas presented in this article apply to all supported languages. I have been closely following BDB XML from the very first release, and there have been tremendous improvements in this product.

Last month in this column I wrote about using BDB XML through the command shell (*WSJ* Vol. 5, iss. 12). I also mentioned some of the basic BDB XML concepts, which I think may be helpful in constructing Java programs. If you aren't familiar with BDB XML concepts, I recommend reading last month's article or checking the online BDB XML documentation.

Installation

BDB XML is distributed by SleepyCat Software, and has an installer for the Windows operating systems. You may choose to build it from the source files, which exist for almost every popular operating system. For convenient coding I used the Eclipse Integrated Development Environment (IDE) from www.eclipse.org. Eclipse is a very nice IDE that makes programming tasks easy and fun. It's available free of charge. In order to use the Java API provided with the BDB XML, we have to include two .jar files in the *Classpath*. These files are *db.jar* and *dbxml.jar*, and they should be located under the .jar directory of your installation path.

Environment

In order to start making something useful with our Java programs, we have to take a moment to decide the properties of the environment that we will use throughout the lifetime of the code. I will explain some of the widely used environment properties. It's true that an embedded database such as BDB XML puts more pressure on the shoulders of the programmers. You should only enable a feature if there is a need for it. If you don't feel very comfortable with these database environment features, refer to a database book, or check them online.

> "You should only enable a feature if there is a need for it"

An important feature of the Environment is that once it has been created, it's not possible to change many of its attributes. The following is a summary of commands:

- ***EnvironmentConfig.setAllowCreate(true)*** – This command enables creation of the environment.
- ***EnvironmentConfig.setInitializeLocking(true)*** – If there are multiple processes or multiple threads, this should be turned on. It locks the database sources for preventing anomalies. Data locking is an important database concept. It's definitely needed in a multiple transaction environment. However, if the transactions are serial, i.e., one starts after the other finishes, you might not need it. In a multiple process or in a multiple thread environment, there are competing transactions. In some situations deadlock may occur because of locking, but this feature comes with a deadlock detector for resolving such resource conflicts.
- ***EnvironmentConfig.setInitializeLogging(true)*** – This command initializes the logging. Logging, which is essential for recovery, is basically keeping track of read and write operations in the database. When there is a power outage you may need to recover using the log files, and in some cases log files are used for replication purposes.
- ***EnvironmentConfig.setInitializeCache(true)*** – Initializes the shared memory. It must be set to true when using multithreaded applications.
- ***EnvironmentConfig.setTransactional(true)*** – Enables transactional support that is essential for most of the applications. A transaction is a set of read and write operations. Conceptually, a transaction looks something like this:

  Start the transaction:

AUTHOR BIO

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

- Read customer credit card number
- Contact Visa, verify the number
- Charge the credit card account
- Ship the item
- Commit transaction

The transaction above is an atomic operation; either all of the operations succeed or none of them does. Imagine that there is a power outage after the third step ("Charge the credit card account"); the credit card will be charged, but the item will not be shipped. This is not good news for the customer. In a transactional database such as Berkeley DB XML, if there is a power outage after the third step, when the system is back on power, it will roll back all of the operations done in the first, second, and third steps. The credit card charge will be refunded because the fourth step ("Ship the item") was not completed, so a customer won't be charged for something he didn't receive.

- ***EnvironmentConfig.setRunRecovery (true)*** – This will turn on the normal recovery feature, which makes sure that the database files are consistent with the log files. For example, after a power failure it's possible that the database files are behind the logs. When the recovery is on, BDB XML will recover the database files from the logs. This feature depends on the setInitializeLogging, which in fact must be turned on in advance. Listing 1 shows a Java snippet for configuring the environment.

XmlManager

This is the high-level class for managing XML collections. Queries are sent to the database using the XmlManager object. XmlManager can also create input streams and collections.

XmlManager has three different constructors. One of them, shown below, takes *Environment* and *XmlManagerConfig* objects as its two arguments:

```
XmlManager(Environment dbenv,
XmlManagerConfig config)
```

XmlManagerConfig, as the name suggests, is the configuration object for the XmlManager. Let's see what kind of configurations can we make using this object:

- ***XmlManagerConfig.setAllowAutoOpen (true)*** – If a query refers to an unopened container, it will be opened automatically.
- ***XmlManagerConfig.setAdoptEnvironment(true)*** – XmlManager will automatically close the Environment when its close() method is called. There will be no need to use the close() method of the Environment object.
- ***XmlManagerConfig.setAllowExternalAccess(true)*** – Allows use of external resources such as DTD entities. It's important to enable this feature, because when using external general entities, the result or part of the result may be outside of the database. In this situation BDB XML will be able to access the external resources.

Adding a Document

After instantiating an XmlManager object, creating a container and then adding an XML document is easy. The Java snippet in Listing 2 shows how to do this. Listing 2 demonstrates creating an XML container called "xbench.dbxml." Following this, an input stream is created from the file "C:/dictionary.xml" and it is added to the container by the container's putDocument() method. The complete Java code is provided in Listing 4.

> "In order to start making something useful with our Java programs, we have to take a moment to decide the properties of the environment that we will use throughout the lifetime of the code"

Querying Data

After setting up the necessary foundations, now we are ready to query the database. As a programmer you can fine-tune the query evaluations using *Lazy,* and *Eager* methods. The documentation states that BDB XML supports the July 2004 W3C XQuery specification. XPath is a sub language of XQuery, therefore it's automatically supported too. Results are provided in the XmlResults object, which requires looping over it as Listing 3 shows.

As I mentioned earlier, I have been following BDB XML for a long time. Unfortunately, the API has not been backward compatible with earlier versions. Code that I wrote earlier for version 1.2 doesn't compile on version 2.2 of Berkeley DB XML because of API changes. This is very inconvenient, and I hope in future versions SleepyCat Software publishes backwardly compatible APIs.

smimarog@cs.umb.edu

Listing 1 (Java snippet shows how to configure the Environment)

```
//Environment configuration goes into this Object
EnvironmentConfig envConf = new EnvironmentConfig();

//If the environment does not exit, create it.
envConf.setAllowCreate(true);

//Turn on shared memory region
envConf.setInitializeCache(true);

//Turn on locking
envConf.setInitializeLocking(true);

//Turn on logging
envConf.setInitializeLogging(true);

//Enable transactions
envConf.setTransactional(true);

//create the Environment Object
Environment environment = new Environment(envHome,
envConf);
```

Listing 2 (Creating a container and adding a document)

```
//open the environment to work on it
environment = new Environment(envHome, envConf);
```

```
XmlManagerConfig managerConfig = new
XmlManagerConfig();
managerConfig.setAdoptEnvironment(true);
managerConfig.setAllowAutoOpen(true);
managerConfig.setAllowExternalAccess(true);
xmlManager = new XmlManager(environment, managerCon-
fig);


//let's create a container named xbench.dbxml
xmlManager.createContainer("xbench.dbxml");
xContainer =  xmlManager.openContainer("xbench.dbxml");

//Get the input stream.
String fileName = "C:/dictionary10.xml";

//Give it a unique name
String docName = "dictionary10";

//Need an update context for the put.
XmlUpdateContext theContext = xmlManager.createUpdate-
Context();

XmlInputStream xStream = xmlManager.createLocalFileInpu
tStream(fileName);

xContainer.putDocument(new XmlTransaction(), docName,
xStream, theContext);
```

Listing 3 (Querying data)

```
import java.io.File;
import java.io.FileNotFoundException;

import com.sleepycat.db.DatabaseException;
import com.sleepycat.db.Environment;
import com.sleepycat.db.EnvironmentConfig;
import com.sleepycat.dbxml.*;

/**
 * @author Selim Mimaroglu
 *
 * written for Berkeley DB XML v2.2
 *
 * This class queries the database.
 *
 * This code is presented as is. If you decide to use
this code, you do so at your own risk;
 * the author takes no responsibility for loss or harm
of any kind.
 *
 */
public class Query_XML {

 public static void main(String[] args) {
  XmlManager xmlManager = null;
  Environment environment = null;
  EnvironmentConfig envConf = null;
  //the database environment is stored under this
directory
  File envHome = new File("D:/bdb_xml_testenv");
  XmlContainer xmlContainer = null;

  try {
   //Configure the database environment
   //Don't forget to use the original configuration
created
   //when the environment was initialized
   envConf = new EnvironmentConfig();
   envConf.setAllowCreate(true);
   envConf.setInitializeCache(true);
   envConf.setInitializeLocking(true);
   envConf.setInitializeLogging(true);
   envConf.setTransactional(true);

   environment = new Environment(envHome, envConf);

   //Create XmlManager and XmlContainer
   //Set up the XmlManager configuration

   XmlManagerConfig managerConfig = new
XmlManagerConfig();
   managerConfig.setAdoptEnvironment(true);
   managerConfig.setAllowAutoOpen(true);
   managerConfig.setAllowExternalAccess(true);
   xmlManager = new XmlManager(environment, managerCon-
fig);
   xmlContainer = xmlManager.openContainer("xbench.
dbxml");
   XmlQueryContext context = xmlManager.createQueryCon-
text();

   String query = "collection('xbench.dbxml')/diction-
ary/e/hwg/hw";

   //Perform the query.
   XmlResults results = xmlManager.query(query, con-
text);

   //Print the results
   String message = new String("Found " + results.
size() + " results for the query: " + query );
   System.out.println(message);

   //Loop over the result set
   while (results.hasNext()) {
    XmlValue value = results.next();
    System.out.println(value.asString());
   }
  } catch (XmlException e) {
   e.printStackTrace();
  } catch (FileNotFoundException e) {
   e.printStackTrace();
  } catch (DatabaseException e) {
   e.printStackTrace();
  } finally {
   if(xmlContainer != null ) {
    try {
     xmlContainer.close();
    } catch (XmlException e1) {
     e1.printStackTrace();
    }
   }
  }

 }
}
```

Listing 4 (Full code, some of it is provided in Listings 1 and 2)

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

/**
 * @author Selim Mimaroglu
 *
 * written for Berkeley DB XML v2.2
 *
 * Some portions of the code are inspired from
"Getting Started with Berkeley
 * DB XML for Java"
 *
 * This class creates the following Database Objects:
 * - Environment
 * - XmlManager
 * - Container
 *
* This code is presented as is. If you decide to use
this code, you do so at your own risk;
 * the author takes no responsibility for loss or harm
of any kind.
 *
 */
public class Create {

 /**
  * This the Environment variable
```

Visit the *New*

www.SYS-CON.com

Website Today!

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

24/7

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

SYS-CON.TV
Watch video of breaking news, interviews with industry leaders, and how-to tutorials

BLOG-N-PLAY!
Read web logs from the movers and shakers or create your own blog to be read by millions

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

EDUCATION
The world's leading online i-Technology university

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

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

INTERNATIONAL SITES
Get all the news and information happening in other countries worldwide

JUMP TO THE LEADING i-TECHNOLOGY WEBSITES:

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

```
  * We have to create an environment, once and only
once
  */
 private Environment environment;
 /**
  * XmlManager manages containers, creates input
streams, runs queries
  * and more
  */
 private XmlManager xmlManager;

 /**
  * We are going to create an environment in  this
location
  */
 private File envHome;

 /**
  * After creating an environment, use the same con-
figuration to
  * open it
  */
 private EnvironmentConfig envConf;

 /**
  * Default Constructor Initializes the variables
  *
  */
 public Create() {
  environment = null;
  xmlManager = null;
  //change this for your needs
  envHome = new File("D:/bdb_xml_testenv");
  envConf = null;
 }

 /**
  * This method creates an environment.
  * You must create an environment with some proper-
ties (like below)
  *  in order to work with the Berkeley DB XML
  */
 public void createEnvironment() {

  try {
   //Environment configuration goes into this Object
   envConf = new EnvironmentConfig();

   //If the environment does not exit, create it.
   envConf.setAllowCreate(true);

   //Turn on the shared memory region
   envConf.setInitializeCache(true);

   //Turn on locking
   envConf.setInitializeLocking(true);

   //Turn on logging
   envConf.setInitializeLogging(true);

   //Enable transactions
   envConf.setTransactional(true);

   //initialize the environment
   environment = new Environment(envHome, envConf);

  } catch (Exception e) {
   //For now this just prints out the Exception mes-
sage
   //Include Exception handling action here
   System.err.println(e.getMessage());
  } finally {
   try {
    //Don't forget closing the environment after you
are done
    if (environment != null) {
     environment.close();
    }
   } catch (DatabaseException e) {
    System.err.println(e.getMessage());
   }

  }

 }

 /**
  * This method creates an XmlManager instance with
the requested
  * specifications
  *
  */
 public void createXmlManager() {

  // if the environment is not created, create it
first
  if(environment == null) {
   this.createEnvironment();
  }

  XmlContainer xContainer = null;

  try {
   //open the environment to work on it
   environment = new Environment(envHome, envConf);
   XmlManagerConfig managerConfig = new
XmlManagerConfig();
   managerConfig.setAdoptEnvironment(true);
   managerConfig.setAllowAutoOpen(true);
   managerConfig.setAllowExternalAccess(true);
   xmlManager = new XmlManager(environment, managerCon-
fig);


   //let's create a container named xbench.dbxml
   xmlManager.createContainer("xbench.dbxml");
   xContainer =  xmlManager.openContainer("xbench.
dbxml");

   //Get the input stream.
   String fileName = "C:/dictionary10.xml";

   //Give it a unique name
   String docName = "dictionary10";

   //Need an update context for the put.
   XmlUpdateContext theContext = xmlManager.createUp-
dateContext();

   XmlInputStream xStream = xmlManager.createLocalFileI
nputStream(fileName);
   xContainer.putDocument(new XmlTransaction(), doc-
Name, xStream, theContext);

  } catch (Exception e) {
   System.err.println(e.getMessage());
  } finally {

   try {
    //close the container when done
    if(xContainer != null) {
     xContainer.close();
    }
    if (xmlManager != null) {
     xmlManager.close();
    }
   } catch (Exception e) {
    System.err.println(e.getMessage());
   }
  }



 }


 public static void main(String[] args) {
  Create create = new Create();
  create.createEnvironment();
  create.createXmlManager();
 }
}
```

Ah, remember when EDI was young and full of promise?

The future of EDI is B2B over IP

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

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

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

xenos

terminalONE™

Secure, intelligent ebusiness transactions

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

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

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

Ensure Interoperability.

Validate functionality.

Eliminate security vulnerabilities.

Test performance & scalability.

Confirm compliance.

Collaborate thru reuse.

Ensure Secure, Reliable Compliant Web Services

PARASOFT®

SOA Test™

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

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

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

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

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

PARASOFT®

We make software work.™

Automated Software Error Prevention™