SYS-CON

NOW INCORPORATING XML JOURNAL

SOA Web Services JOURNAL

December 2005 Volume 5 Issue 12

Web Services – Soup to Nuts
Simplified Web services development using visual tools
pg. 20

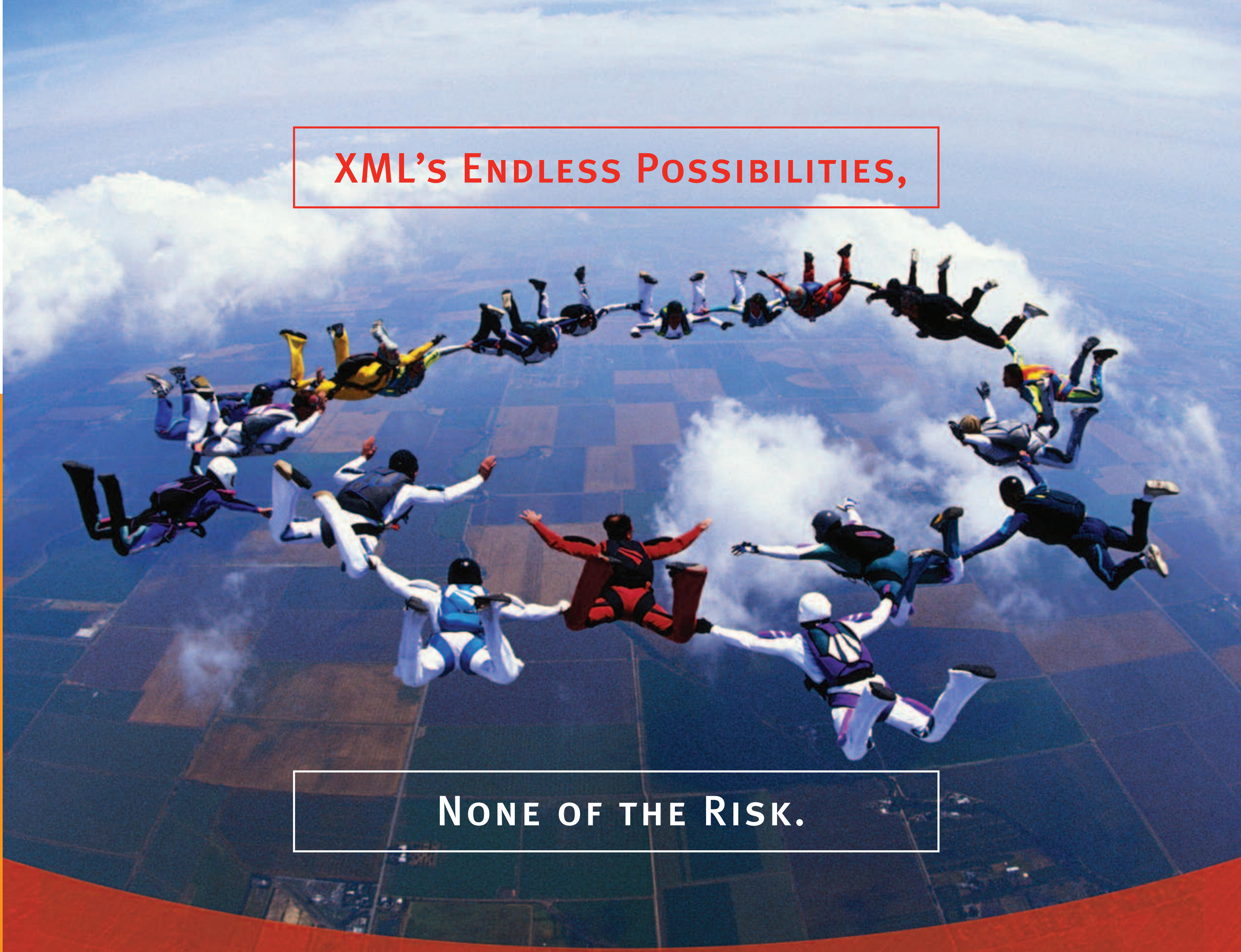
Networks for Service-Oriented Architecture
The impact of Web services on wide-area networks
pg. 30

Bringing SOA to Life: The Art and Science of Service Discovery and Design
Practical guidelines and experiences from real-world SOA projects
pg. 42

WEB 2.0

The Global SOA

PG. 10

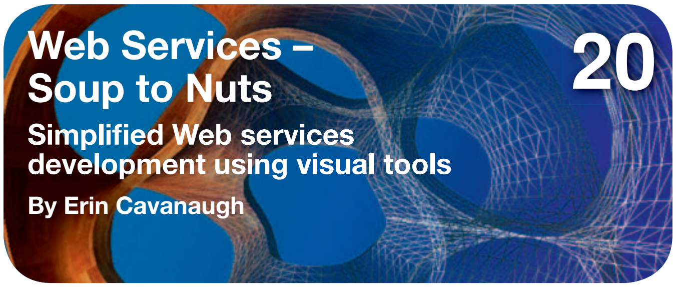

XML's Endless Possibilities,
None of the Risk.

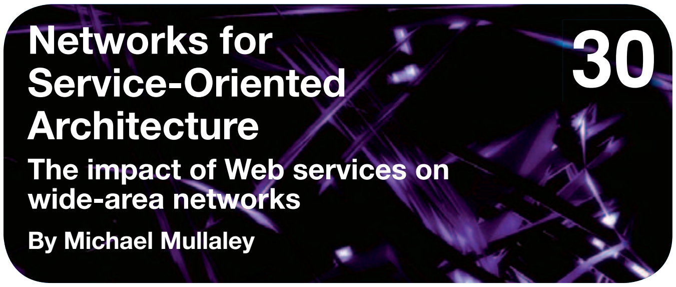
SOAWebServices JOURNAL

December 2005 Volume 5 Issue 12

XML JOURNAL

Inside WSJ





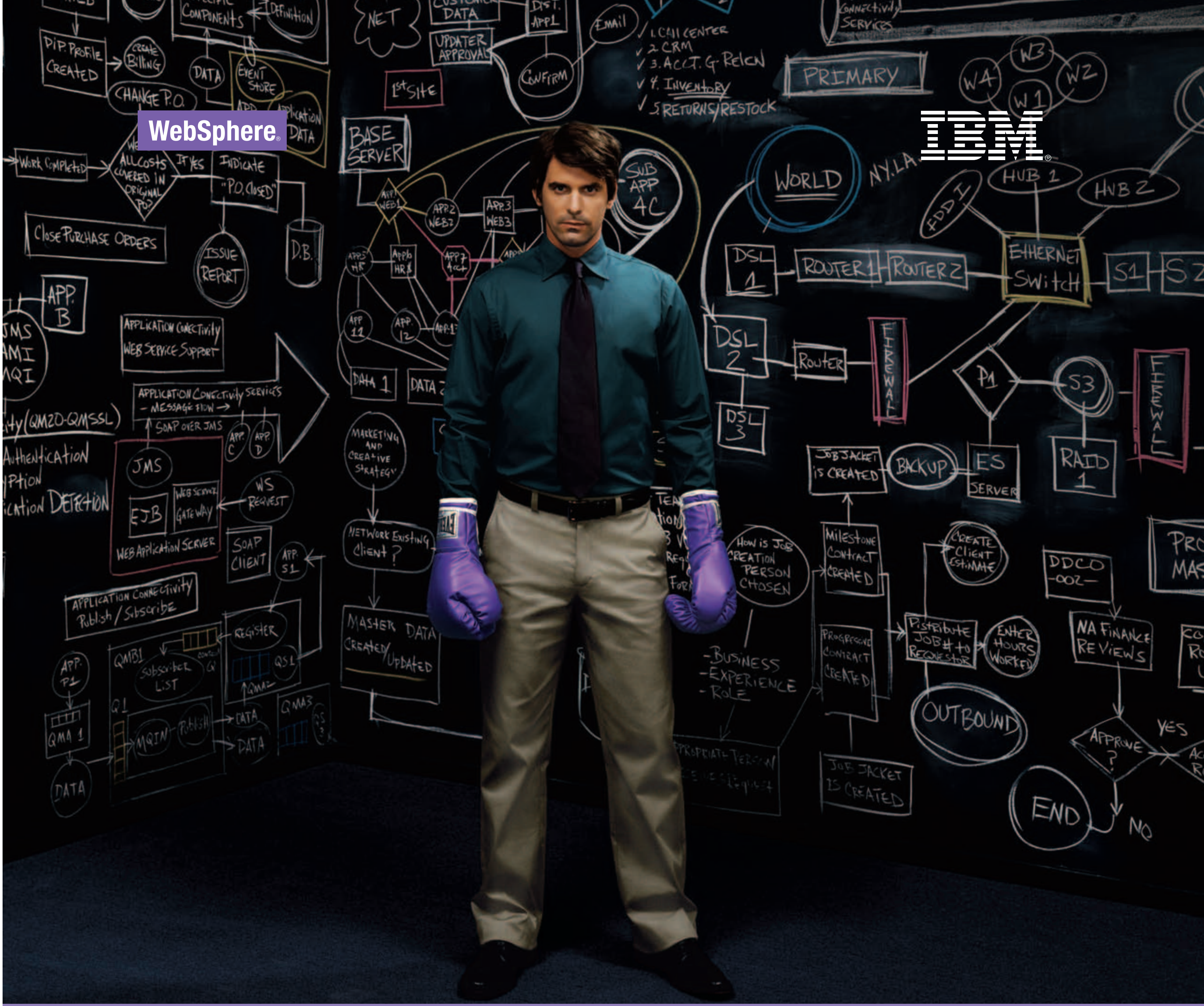
XML JOURNAL SUPPLEMENT

This Month

Flexible Identity Federation Centralized Policy Enforcement Points

XML gateways to the rescue

XML-Based Interop, Close up



IBM WEBSPHERE PRESENTS

YOU VS REWRITE, REVISE RE-EVERYTHING

INFLEXIBILITY: MEET SOA

PLUS: TANGIBLE BUSINESS BENEFITS ★ BEST PRACTICES FOR BEST RESULTS
OVER 10 YEARS OF WORLD-CLASS INTEGRATION EXPERTISE

FEATURING:

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

IBM MIDDLEWARE. POWERFUL. PROVEN.

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

1
2
3
4

SOA Web Services JOURNAL

INTERNATIONAL ADVISORY BOARD
Andrew Astor, David Chappell, Graham Glass, Tyson Hartman,
Paul Lipton, Anne Thomas Manes, Norbert Mikula,
George Paolini, James Phillips, Simon Phipps,
Mark Potts, Martin Wolf

TECHNICAL ADVISORY BOARD
JP Morgenthal, Andy Roberts,
Michael A. Sick, Simeon Simeonov

EDITORIAL
EDITOR-IN-CHIEF
Sean Rhody sean@sys-con.com
XML EDITOR
Hitesh Seth
INDUSTRY EDITOR
Norbert Mikula norbert@sys-con.com
PRODUCT REVIEW EDITOR
Brian Barbash bbarbash@sys-con.com
.NET EDITOR
Dave Rader davidr@fusiontech.com
SECURITY EDITOR
Michael Mosher wsjsecurity@sys-con.com
RESEARCH EDITOR
Bahadir Karuv, Ph.D Bahadir@sys-con.com
TECHNICAL EDITORS
Andrew Astor andy@enterprisedb.com
David Chappell chappell@sonicsoftware.com
Anne Thomas Manes anne@manes.net
Mike Sick msick@sys-con.com
Michael Wacey mwacey@csc.com

INTERNATIONAL TECHNICAL EDITOR
Ajit Sagar ajitsagar@sys-con.com

EXECUTIVE EDITOR
Seta Papazian seta@sys-con.com

EDITOR
Nancy Valentine nancy@sys-con.com

ONLINE EDITOR
Roger Strukhoff roger@sys-con.com

PRODUCTION
PRODUCTION CONSULTANT
Jim Morgan jim@sys-con.com
LEAD DESIGNER
Andrea Boden andrea@sys-con.com
ART DIRECTOR
Alex Botero alex@sys-con.com
ASSOCIATE ART DIRECTORS
Abraham Addo abraham@sys-con.com
Louis F. Cuffari louis@sys-con.com
Tami Beatty tami @sys-con.com

CONTRIBUTORS TO THIS ISSUE
Sriram Anand, Erin Cavanaugh, Abhishek Malay Chatterjee,
Manas Deb, Dan Foody, Johannes Helbig, Dion Hinchcliffe,
Adam Kolawa, Manfred Kroll, Vikas Kumar, Francois Lascelles,
Paul Maurer, Selim Mimaroglu, Michael Mullaley, Vivek Raut,
Sean Rhody, Alex Rosen, Ajit Sagar, Alexander Scherdin,
Vineet Singh

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

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

Holiday Wishes

It's December, and you know what that means: holiday wishes and New Year's predictions are due. That's right, once again we'll gaze into the WSJ mystic crystal ball (okay, so it's a Christmas ornament – we're on a budget here) and come up with our prognostications and pleadings. So, without further ado, here's the list.

WRITTEN BY
SEAN RHODY

This will be the year we finally get some traction on Business Process Management – you know, that thing everyone is talking about, but no one is doing yet. To do this, one or more of the solution providers is going to have to bite the bullet and do a decent job of implementing BPEL, along with WS-Transaction, WS-Choreography, and WS-Orchestration. It's time to stop hashing it all out and implement it – and make it interoperable.

One million more WS-* standards will be released. Most of them will overlap with each other. Few if any will be implemented. None will achieve the widespread acceptance of the basics of Web services – XML, SOAP, UDDI, and WSDL. While I'm exaggerating the number of specifications and standards that will be released, the reality is that there are already too many standards. Here's my holiday wish in this area: stop now. Don't create yet another standard, we have plenty. Work to extend the standards we have, as they cover most of what we want to do. I'm not discounting the significance of things like WS-Reliable Messaging, but really guys, come on. No one can keep up with more than a half dozen or so standards when it comes to Web services – we need one for discovery (UDDI), binding (WSDL), transaction management (legion), business process (also legion), security (legion as well), and maybe a handful of others. The rest should be rolled into these overarching categories. Why? Because then it would force vendors to actually implement some of this, instead of just creating new standards every couple of weeks. That would be good. I've said it before, but I'll say it again: we don't need more standards. So stop already, and implement – let the folks in IT catch up.

Service-oriented architecture will continue to be the buzzword for the year. SOA will be adopted by more organizations, including a number of those who should not go there. Service orientation offers the possibility of real advances in IT – advances that can be coupled strongly to business value. It also requires organizational change and modification to IT governance that many folks have yet to really tackle. It's a lot easier to change the software than it is to change the people who run the software – even when they're willing to change. It should be fairly intuitive, but at the risk of stating the obvious– you cannot buy an SOA. You can buy technology components that make SOA easier to implement, but you cannot buy the whole thing – it requires people, design, business involvement, governance, change management, and especially a measurable ROI. If the ROI is not there, don't do it.

Now, lets move on to my wish list. I wish Web services had a real, multivendor, language-neutral, XML-based user interface approach. No, sorry, AJAX is not it – it's not language neutral, it's not really about building user interfaces, it's more about asynchronous messaging, which while necessary, is only a piece of the puzzle. We really do need this – a way to design user interfaces for services, eliminating the bad parts of HTML while maintaining the download-only nature of the browser for ease of application deployment.

I'd also like to see some services. By this I mean I'd like to see OASIS or the W3C or some similar organization (RosettaNet would be a good candidate) start to define AND implement a set of business services, using Web services as the technology. Creating composite applications and really leveraging the power of an SOA would be a lot easier if standard services were defined. Notice I called for standard services, not a services standard – please, not another standard!

Last, I'd like a Jaguar – but I'll settle for you all having a safe and happy holiday season.

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

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

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 Publications, Inc., reserves the right to revise,
republish and authorize its readers to use the articles
submitted for publication.

SYS-CON MEDIA

Using Services

WRITTEN BY

AJIT SAGAR

It never ceases to amaze me how ambiguity in the definition of simple terms can lead to design choices that have a huge impact on the success of projects. Recently I had a long discussion with a colleague at a client site, where we are in the process of assessing the artifacts that have been created for a Web services–based service-oriented architecture. While we are talking about terms and definitions, let us be clear about the fact that there is a paradigm called service-oriented architecture, and there is a platform on which it can be realized called Web services. Often the two are confused. They are definitely not the same. One is a concept, the other is a technology platform.

Anyway, back to some basic definitions. As you get into defining Web services for service orientation, there needs to be a clear distinction between the definition of the service and its usage. If we get down to the ground-level basics, one is a noun and the other is a verb. A service offers a certain kind of business functionality – it is the *what*, and it is a noun. The usage of the service is most commonly expressed through a set of scenarios, each of which is represented in UML as an instance of a use case. The use case is the sequence of activities that constitutes an instance of the usage of the service, and it is defined as a verb. For example, CreditCheck is a service. On the other hand, VerifyCredit would be a use case that leverages the CreditCheck service to provide certain value to the calling application or component.

A service is specified by a contract. For example, a Web service is typically specified by its WSDL definition. The service specification does not include scenarios for its usage. The contract includes an SLA (service level agreement) that ensures that the consumer of the service clearly understands how to call the service, what information to pass to it, what information to expect back, the protocol used to call it (in the case of Web services, this is obviously SOAP), how the service behaves under different constraints, etc.

As I mentioned earlier, sometimes these terms are confused. Recently I came across an instance in which the services were defined as a use case. While this may seem to be a trivial deviation, in the grand scheme of things it can have substantial problems as the design of the system matures. For example, if the designer of the service is defining it as a use case, he or she needs to have a very comprehensive view of all of the moving parts of the system. The person who designs the service usually does not have this view. Typically the service designer has a view of how the service is to be called, not of who uses it, and when and how.

The problem has its roots back in the great business-IT divide. The use case is derived from a business requirement that is specified by business. When the use case is defined, the service it calls it treated as a black box. The person defining the use case should not be concerned about how the service is implemented, and this is possible only if the service has a well-defined contract. Now, if the person defining the use case is also defining the contract of the service that is used in the use case, he or she will have a view of the service within the narrow scope and context of the use case. This means that the service might not be applicable to callers, and therefore, it will not qualify as a reusable service. The result will be an unmanageable volume of granular services with ill-defined contracts. A system built on this base is doomed to malfunction.

So as you venture out into service-oriented projects, make sure that you have clear distinction among what the system is composed of, what is exposed as a service, and the different use cases for the service.

■ 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

Application: Operations Monitor

The Global SOA

The subject of Web 2.0 has become profoundly important over the last year. Web 2.0 describes the next generation of the Web as an application platform where most of a user's software experience resides. The subject is somewhat controversial, but it's becoming ever more apparent as the successor to monolithic system architecture, prepackaged software, and traditional Web applications.

WRITTEN BY
DION HINCHCLIFFE

Software as a Service (SAAS) and Web as Platform are only two of the larger mantras of Web 2.0 that most of the major software vendors have begun to embrace recently. Yet not only is Web 2.0 still very misunderstood, it's actually part of an even larger way of thinking about software in a fully service-oriented manner. This includes building composite applications, remixing data, building ad hoc supply chains, harnessing user involvement, aggregating knowledge, and more. Web 2.0 is becoming embodied in best practice sets such as service-oriented architecture (SOA).

The term Web 2.0 was originally coined by O'Reilly's Dale Dougherty to describe the forces behind the huge post-dot-com success of Internet companies like Google, eBay, Amazon, and iTunes, as well as noncommercial, emergent Web phenoms such as Wikipedia and BitTorrent. Web 2.0 describes Web experiences that fundamentally engage users by: 1) allowing them to participate in sharing information and enriching data freely, 2) readily offering their core functionality as open services to be composited or "mashed up" into new services and sites, and 3) placing the Web at the center of the software experience both in terms of data location as well as where the software is.

Applications in which the Web app is primarily an online catalog are changing the most. Instead of being just a way to browse for products or information, the Web 2.0 app is itself the tip of an iceberg that integrates services and data from multiple sources and then makes the results available to users and other Web 2.0 apps. At the end of the day, the integration achieved by one Web 2.0 app will likely get rolled up into someone else's Web 2.0 app.

Now evangelized by Tim O'Reilly and others as *Web 2.0*, the concepts themselves are not really new, but they are beginning to dominate the IT industry's collective consciousness. The powerful force of architecture of participation, which is the combined network effects of pervasive two-way participation (blogging, wikis, and media sharing), is having a huge effect and is creating a single, communal service architecture on the Web. In the end, users want access to information anywhere, from multiple sources, without synchronization, delay, or maintenance (software upgrades, data backups, etc.). Users want to be able to share knowledge and collaborate with peers. To do this they need to be using the same underlying

> "Web 2.0 is about the entire Web being a reusable, shareable, public SOA"

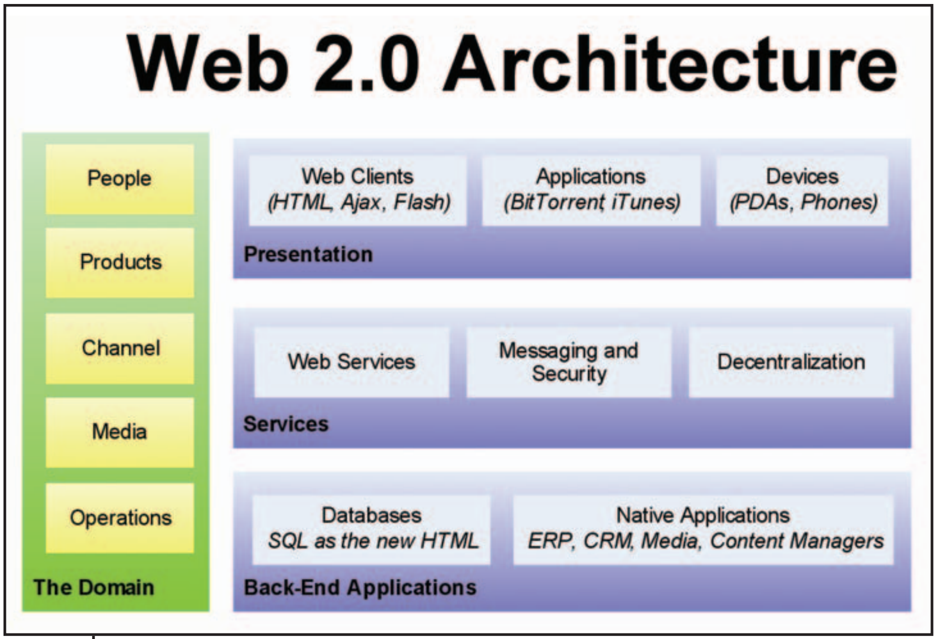


FIGURE 1 One conceptual view of Web 2.0 architecture

set of technologies and paradigms, and this is what Web 2.0 promotes.

A Closer Look at Web 2.0

While large traditional software organizations such as Microsoft, IBM, and Oracle are struggling mightily to deliver traditional prepackaged software on long-term, aperiodic cycles, a newer generation of companies (Google, Yahoo!, Amazon, eBay, and many other smaller companies like 37signals and del.icio.us) are delivering capable software entirely online. This new agile way of providing functionality as a service over the Web provides a nimble, continuous experience with no software upgrades, and no synchronization of data or programs among work, home, or mobile locations. Equally important is that it provides a way to build new capabilities on top of existing functionality that becomes larger than the pieces.

Tim O'Reilly, one of the leading evangelists of the Web 2.0 approach, generally provides seven classic characteristics of Web 2.0 software. These are described in the subsections below.

Web as Platform

Software and services are now the same thing, and the Web has become a computing platform in its own right. The Web is where most software is moving for cost, convenience, agility, and increased overall value.

Harnessing Collective Intelligence

The network effects of massive amounts of users make the collaborative Web a much more potent force than stand-alone software. Wikipedia says network effects "cause a good or service to have a value to a potential customer dependent on the number of customers already owning that good or using that service." Put another way, online collaborative entities such as Wikipedia are a network effect of the combined contributions of their users. This is a classic example of the high-value emergent properties of Web 2.0 forces.

Data Is the Next Intel Inside

The core functionality of many modern information systems is *not* software; more accurately, it's the *valuable data* within it. Look at Google's search database, Amazon's products and associated reviews, or eBay's auctions. While the services these sites provide are also important and integral, the data they possess are just as important, perhaps even more so.

End of the Software Release Cycle

When software is on the Web, upgrading becomes a different experience. Discrete changes become less obvious while continuous improvement becomes the norm. Because services are always available 24 hours a day to anyone connected to the global Internet, upgrades and improvements to service are instantly available and encouraged to be as nondisruptive as possible.

Lightweight Programming Models

When the clients of Web software are numerous and diverse, complex standards can get in the way, reduce interoperability, and stifle connectivity. Web 2.0 realizes that demand for services will route around unnecessary impedance and leverage the easiest methods that work well. This has led to simpler services such as REST and RSS instead of SOAP and WS-* standards. Remixing and compositing of services is also much easier with clean, clear, simple models, and this has also promoted loose coupling and suppler services, especially in the large. Dynamic programming languages that support rapid change are becoming more popular too.

Software Above the Level of a Single Device

PCs are an increasingly smaller aspect of the Web. With so many different devices such as mobile phones, PDAs, and even digital video recorders and personal media servers becoming connected to the Web and both providing and consuming functionality and content, the Software as a Service landscape of the Web now includes these in the picture.

Rich User Experiences

The Web has ceased to be about static Web pages. They still exist, but they are much less important. More central to the Web are rich user experiences that immerse the user in the functionality of the services available on the Web without getting in the way. The AJAX browser application model is famously a Web 2.0 technique that uses the raw ingredients of modern browsers to provide the full interactive experience of native applications to the user while leveraging XML Web services on the back end to provide access to data and services.

There are many interlocking, reinforcing details that are vital to appreciating the best practices in the Web 2.0 toolset, however, the intent of this article isn't to explain every nuance of Web 2.0. Instead the article should convey a general mental model of it and describe Web 2.0's striking similarity to the SOA model. The point is that Web 2.0 describes the Web as a galactic collection of high-value Web services to be used, reused, and leveraged to meet users' needs. The Web itself provides the universal fabric upon which all of this rests and this includes the standards, the users, and the data. The premise of this article is that Web 2.0 actually describes the Web as the convergence of software services into a global service-oriented architecture.

Web 2.0: The Web as the Global SOA

As the rich user experiences portion of Web 2.0 notes, Web pages as the face of the Web are becoming less important in general. Rather, the foundation for the Global SOA has Web services as the underpinning. Open Web services connect the silos of the data and functionality that often fragmented the previous generation of the Web by providing clearly formatted, machine-digestible information instead of the less useful visual HTML layout. This enables information consumers anywhere to take data from a service, present it, remix it, and syndicate it just as they need. Note that the exact mechanism of data sharing in Web 2.0, whether it's XML/HTTP, REST, RSS, SOAP, JSON, etc., doesn't matter as long as it's easily consumable.

When it comes to SOA and services, the principles of service-oriented architecture strongly encourage the liberation of the underlying functionality of software by providing ready, open access via standards. Web 2.0 also strongly promotes open services, as observed above. Both SOA and Web 2.0 embrace Web services in all their forms, though SOA usually has a more complex, hard-wired service model, while Web 2.0 encourages simpler, malleable forms with clear overlap in the middle.

Web 2.0 describes "data as the next Intel inside" and paints a vision of vast interactive access to back-end databases through an open service model. SOA encourages this as well.

Admittedly, most SOAs are still conceptually trapped inside an organization's firewall or VPN. Also, Web 2.0 envisions the global Web as the global stage upon which to act out grand visions of constructing vast supply chains of data and mashed-up functionality. However this is only a matter of scale and is not a genuine difference at all.

	Web 2.0	SOA
Service Model	Web services	Web services
Preferred Service Standards	HTTP, XML, RSS, REST,	WSDL, UDDI, SOAP, BPEL, WS-*
Composition Mechanisms	Web server aggregation (remixing, mash-ups), native application access, user directed remixing	Orchestration, coordination, service wrapping
Data-Centric	Yes	Yes
Reusability	Yes, very	Yes, somewhat
User Interface	Yes, explicit with AJAX and emphasis on RIAs	No, implicit
Pliability and Malleability	Very, via simple data formats and lightweight programming models	Yes, more formal, via composition and orchestration, also via principles of service-orientation
Scalability	Yes, via radical decentralization and cost-effective scalability	Not explicit
Business Models	The Long Tail, network effects, harnessing collective intelligence, achieving control of unique, hard–to–re-create data sources, customer self-service	BPM, BPR, Asset Integration, Data Fusion, Legacy Asset Life Extension, Business Activity Monitoring, Business Intelligence
Design Patterns	AJAX, syndication, multi-device software	Service Layer, Service Bus, Unit of Work, Idempotent Message, Reservation, Point-to-Point Channel, Publish-Subscribe Channel, Content Routing
Architectural Principles	Enhancement by extension, Autonomy, Radical Trust, Participation, Loose Coupling, Reusability, Permalinks, Remixing/Mashing, Personalization	Autonomy, Statelessness, Service Contracts, Interface First Design, Abstraction, Common Vocabulary, Loose Coupling, Consumability. Discoverability, Composability, Static Typing
Core Competencies	• Software as a Service • Control over data sources • Trusting users as co-developers • Harnessing collective intelligence • Leveraging The Long Tail • Software above the level of a single device • Lightweight UIs, development, and business models	• Functional encapsulation • Data as an asset • Enterprise stovepipes • Accessibility • System and data integration • Cost-effectiveness • Business agility • B2B self-service • Open standards • Ontologies • Transparency of operation • Consumption-guided business processes

TABLE 1 **Comparison of Web 2.0 and SOA concepts**

Do the connections between Web 2.0 and SOA go deeper, to a more fundamental level than sharing Web services, encouraging composition and reuse, and leveraging cross-cutting views of information to meet users' needs? Are Web 2.0 and SOA actually different at their core, and if not, how else do they relate?

In my opinion, there are at least two major conceptual connections between Web 2.0 and SOA. The first is that Web 2.0 can indeed be conceptualized as a Global SOA that is already hosting millions of services, thousands of composite applications, and millions of users, today. Second, many traditional brick-and-mortar businesses that are currently implementing SOA as their architectural model will need to connect their Web-facing apps to their internal SOAs, thereby aggregating them onto the Web for further use and composition by their business partners and customers.

Thus Web 2.0 is about the entire Web being a reusable, shareable, public SOA. It's also about organizations adding their own, extant SOAs to the Web. The majority of today's organizations that are withholding their SOA assets from the Global SOA will miss out on the benefits of leveraging large-scale network effects, customer self-service, harnessing users as co-developers via mash-ups, access to The Long Tail, and much more.

Note that the industry analyst firm Gartner

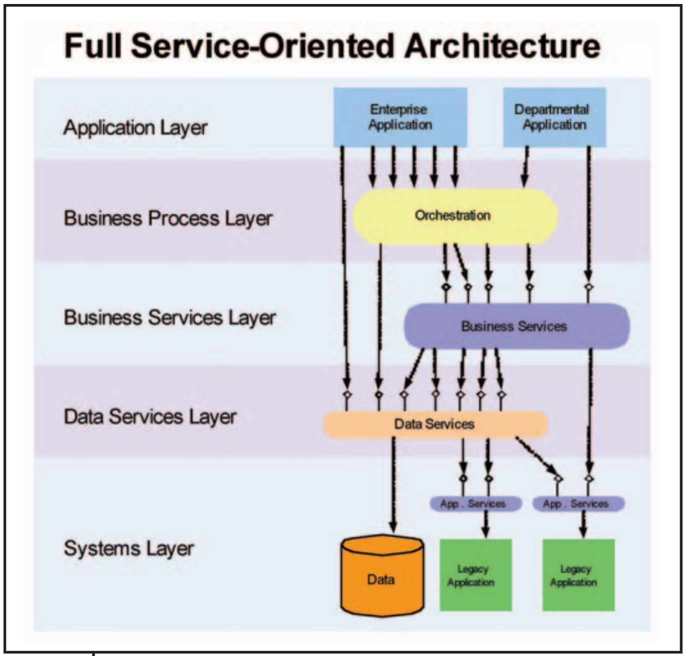


FIGURE 1 **Classic view of a robust SOA**

recently reported that 80 percent of all software development would be based on SOA by 2008. By 2006, Gartner believes that 60 percent of the $527 billion IT professional services industry will be based on exploiting Web services and technology. This is serious convergence of focus. If it's true, this means that more than half of all software development, SOA and otherwise, will revolve around the Web technology, inside or outside organization boundaries. This means Web 2.0 and SOA will be so deeply intertwined that they will inevitably begin to converge into an almost indistinguishable set of software development practices.

However the real question is: Do the deep connections between the two provide compelling advantages as IT departments and Internet businesses launch Web 2.0 applications, open services, and SOAs? One possible problem is that many people outside of the IT industry haven't heard of SOA. Even then, there are vociferous arguments about what SOA really is, just as there are endless debates about what exactly Web 2.0 is. However in the end, there are too many best practices that need to cross-pollinate and SOA's IT-bound sphere of influence isn't really a factor because people building Web 2.0 will mostly have the same skill sets and background. Only Web 2.0 architects will have to understand these techniques and their connections. Web 2.0 and SOA users themselves will generally be blissfully unaware of Web 2.0 or SOA as their underlying organizing principles.

All of this is not to say that Web 2.0 and SOA don't also have significantly divergent elements too. Web 2.0 emphasizes a social aspect that SOA is completely missing, and probably to its lasting detriment. SOA has much more central configuration control, management, and governance, while Web 2.0 is freewheeling, decentralized, grassroots, and has virtually no command and control structure. Web 2.0 also talks about presentation, and the front end is displayed to the user. SOA is largely silent on the issue of presentation, though it certainly admits its existence. SOA tends to be generic and faceless, whereas Web 2.0 shines brightly on human/service interaction. They seem to need each other to be whole. Finally, Web 2.0 is almost too informal and practically calls out for discipline, while SOA is mute and autistic in comparison, a technical virtuosity that wants to be social but that doesn't know how. UDDI? Not likely.

All of this encourages us to view one through the other to check basic principles. For example, SOA has best practices for building business processes into vast supply chains (and so does Web 2.0). SOA is also a mature view of software that eschews a technical view of information and data. Furthermore, it identifies a motive force for business processes via something called orchestration. This is a concept that Web 2.0 does not have explicitly and could certainly use, though its users provide it in some degree. SOA principles also encourage creation of a common vocabulary across establishing lexicons in language that is native to the domain.

SOA also gets very close to addressing a big issue in software development today: that too many IT systems tend to have technology myopia and ignore their most important elements – the people who use them and the way that they work. Web 2.0 gets this part even more right in all of the significant ways. Web 2.0 embraces people, collaboration, architectures of participation, social mechanisms, folksonomies, real-time feedback, etc. All of these are things that SOA – in its more traditional, corporate clothes – does not address, at least not explicitly. The complementary, highly overlapping nature between the two seems clear.

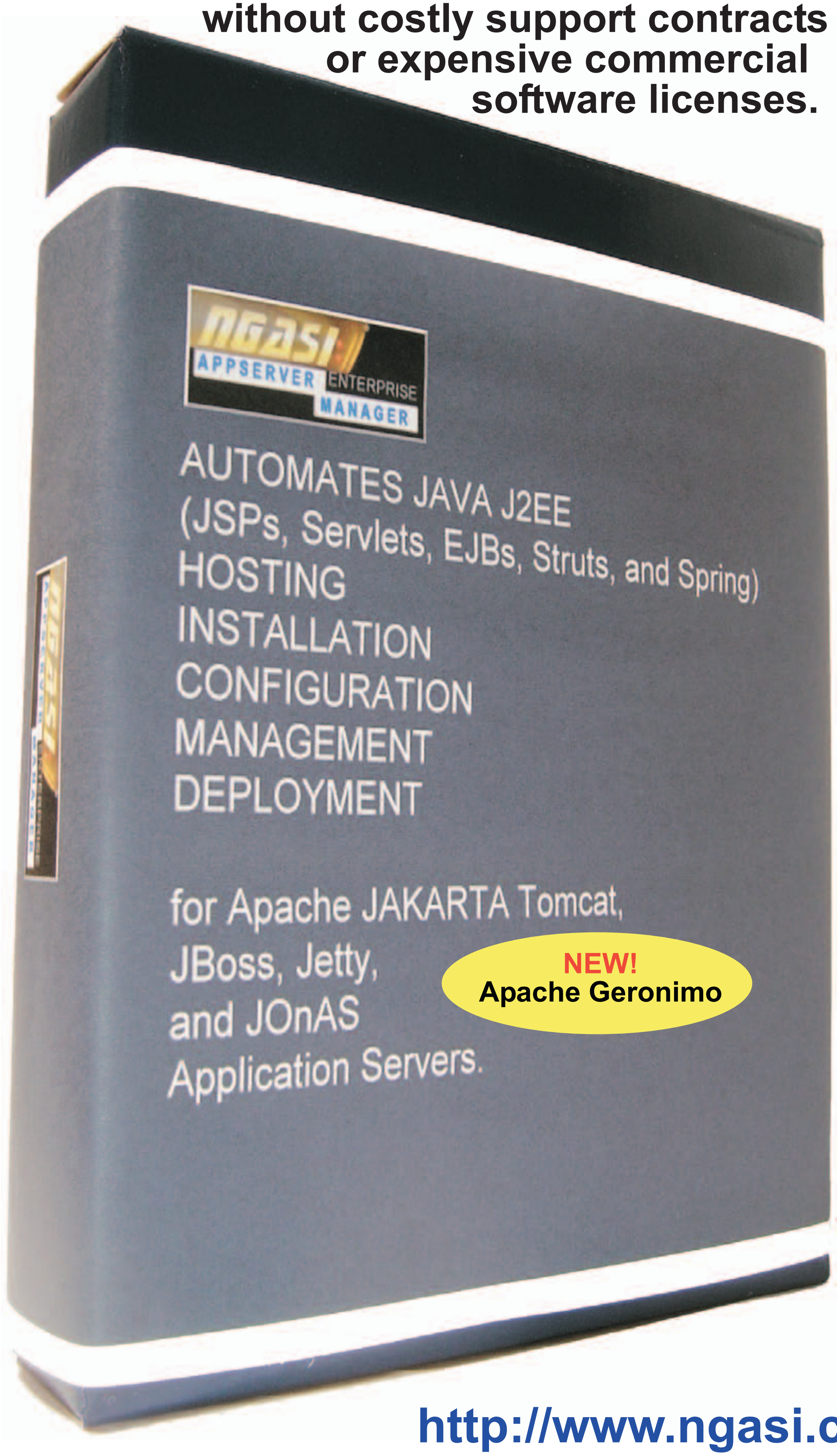
There are other synergies as well between these two powerful software approaches. One checks and finishes the other. SOA is both a heavyweight version of Web 2.0 and a key archetype for it as well (though admittedly, one that lacks a few important ingredients). What is compelling is that Web 2.0 actually has powerful mechanisms that complete a service landscape. For example, Web 2.0 offers services a face and includes numerous best practices for presentation and technical innovation such as radical decentralization for stability and scalability. In the service landscapes that are common today, there is a lot of SOA without the "A": services with nothing tying them together. Web 2.0 can be the "something" that integrates and visualizes these resources. Web 2.0 also identifies important techniques to immerse users into social processes that can make data and services exposed by SOA vastly more valuable.

Web 2.0 is really the Global SOA, available to the whole world today. Don't forget that it will also be connected to your local SOA in ways you will need and are barely beginning to suspect. Be prepared to leverage Web 2.0 and SOA, and reap the enormous benefits of these emerging mindsets and toolkits.

About the Author

Dion Hinchcliffe is cofounder and chief technology officer for the enterprise architecture firm Sphere of Influence Inc. (www.sphereofinfluence.com), founded in 2001 in McLean, Virginia. A veteran of software development, Dion works with leading-edge technologies to accelerate project schedules and raise the bar for software quality. He is highly experienced with enterprise technologies and he designs, consults, and writes prolifically. Dion actively consults with enterprise IT clients in the federal government and Fortune 1000. He also speaks and publishes about Web 2.0 and SOA on a regular basis.

■ ■ ■ dhinchcliffe@sphereofinfluence.com


NGASI
APPSERVER ENTERPRISE MANAGER
AUTOMATES JAVA J2EE
(JSPs, Servlets, EJBs, Struts, and Spring)
HOSTING
INSTALLATION
CONFIGURATION
MANAGEMENT
DEPLOYMENT
for Apache JAKARTA Tomcat,
JBoss, Jetty,
and JOnAS
Application Servers.
NEW!
Apache Geronimo

Implementing a Successful SOA Pilot Program

Best practices for selecting and deploying SOA projects

■ Day by day, company by company, IT organization by IT organization, today's enterprise is busy architecting for business-solution agility and the alignment of key assets around the emerging service-oriented architecture (SOA) umbrella. The ability to embrace SOA leads to the ability to rapidly capitalize on future IT investments and leverage existing technologies both inside and outside of your organization. Many organizations will struggle as they seek to identify, implement, and build on their first SOA forays into the new environment. This is unfortunate, because a number of best practices and lessons learned can be applied from the past and the growing SOA experience from technology providers and practitioners. As organizations work to extend IT capabilities utilizing SOA, many choose to implement a key first step: a pilot program, which enables the greater understanding and methodological deployment of Web services and SOA around a structured initiative.

WRITTEN BY

DAN FOODY

ALEX ROSEN

The Pilot Program offers an expandable approach to SOA, and is a strong foundation for the larger blueprint of a business-driven architecture (BDA). You will find within the context of the following passages a concise understanding of the SOA reality and its application. Each organization will engage SOA differently and extract the greatest benefits based on its unique needs and its environment. Information and tools are the enablers that allow you to determine the best path. In the end, the SOA corridor to success or failure will be paved by the development of an overarching strategy and its deployment across the corporate domains of people, process, technology, and organizational requirements.

Agility

IT agility is one of the main concerns for today's complex enterprises. Even the most well planned IT infrastructures face significant challenges introduced by mergers, acquisitions, and rapid growth of the enterprise. To increase efficiency, companies must phase out functional silos that incorporate overlapping and duplicate processes and add complexity to their IT infrastructure.

SOA provides the framework to re-architect IT infrastructure, eliminate redundancy, and accelerate project delivery via consolidation and reuse of services (often referred to as Web services). SOAs can adapt quickly to changing business needs, deliver new IT projects faster and at lower costs, and reduce ongoing IT administrative and infrastructure expenditures.

In an SOA, applications are not built as stand-alone, monolithic silos. Instead, business-relevant services (e.g., customer, ordering, inventory, etc.) are either built new or layered on top of existing applications. Then, project-specific application logic is built on top of the services in a separate architectural tier. This approach is what leads to the following benefits of SOA:

- By building services, they can be leveraged across multiple projects – thus reducing redundancy and avoiding the need to rebuild functionality in new projects. This helps deliver projects faster and lowers ongoing costs.
- By decoupling project-specific application logic from the services, it enables flexibility because this logic can be easily changed, updated, or even replaced to address new business needs – thus combining the underlying services in new ways without having to change or rewrite them. This leads to business agility as well as lower cost over time.

The benefits are compelling, but SOA also changes the dynamics of IT by introducing interdependencies across projects, applications, and even IT teams. Historically, allowing project teams to be autonomous drove accountability to the project teams and managed risk by limiting the external interactions. In contrast, the benefits of SOA are in many ways derived from external interactions (using services built by other project teams). These two worlds must be brought together without falling into the traps that exist at either extreme: a brittle, unreliable, and difficult-to-maintain spaghetti of application interconnections that can occur when project teams are left to build and use services without the right controls in place; or, at the other extreme, a central organization whose job is to connect applications together, which becomes a bottleneck for project teams, reduces accountability, and increases the risks to project success.

Because of the changes required and the

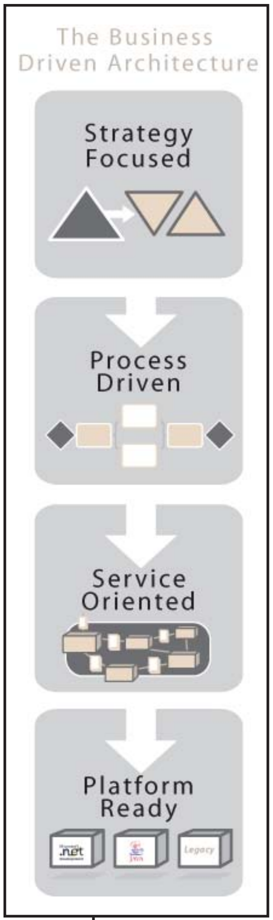


FIGURE 1 **Business-driven architecture**

unique pitfalls of SOA, before moving to this form of architecture, careful preplanning is essential. Your move to SOA is migratory and leverages existing applications that deliver value while building new applications and services. This generates a complex heterogeneous environment that avoids the "rip and replace" approach and focuses on building new services only when a new opportunity arises or when existing solutions do not deliver the necessary results.

Critical Pre-SOA Initiative Questions

1. How do we phase in and successfully manage the move to SOA within the current IT environment?
2. How do we leverage the existing IT infrastructure and investments in the SOA?
3. What procedures need to be established or modified?
4. Where will policy and security reside?
5. How do we explain the benefits of SOA and provide financial justification to the business stakeholders?
6. How does the move from silos to SOA impact our ability to detect and resolve problems?
7. Who owns shared applications or services and how do we allocate costs and budgets for these services?
8. What do we need to do to ensure our SOA scales from pilot to production?

With thorough, up-front planning, a transition to SOA can be achieved efficiently and provide tangible benefits to the enterprise. The following passages will offer guidance in answering the questions listed above.

What Are the Overall Goals of the SOA Initiative?

It is important that you clearly understand and are able to articulate your organization's unique goals that will drive the move to an enterprise-wide SOA. Every enterprise SOA will be driven by unique business and technical goals – as well as by short-term and long-term goals. Understanding these up front will help you choose the right projects and help ensure the success of early implementations.

Critical Questions to Answer

1. What are your primary reasons for using SOA? (To reduce costs? Achieve better flexibility? Enable faster delivery? Improve customer satisfaction?)
2. What is driving your near-term use of SOA? (Connecting your core applications? Integrating with your partners? Providing a single view for customers and/or users? Getting real-time business metrics? Regulatory compliance?)
3. What is the long-term potential for SOA in your organization? (Faster product introductions? Flexible outsourcing? Business process flexibility? Stricter governance? Other?)

Possible SOA Goals

Operational Efficiencies

- Reducing infrastructure and management costs
- Improving end-to-end process visibility, security, and/or control
- Streamlining root cause analysis and automating triage

Business Efficiencies

- Accelerating time to market of critical business applications and processes
- Enabling easier compliance with policies (security, business, regulatory) from all functional areas
- Maintaining business continuity across the enterprise through real-time response to security, performance, or availability breaches
- Seeing business metrics in real time

Development Efficiencies

- Increasing service reuse and reducing development costs
- Reducing time to deployment of new services

Integration Efficiencies

- Reducing integration cost and complexity through the use of industry standards
- Seamlessly interoperating with, and leveraging the capabilities of, key third-party infrastructure solutions such as identity management systems, directory services, and enterprise management systems

By documenting precise goals for an enterprise SOA, teams involved in the initiative can more effectively choose the right projects, manage the projects (e.g., avoid "scope creep"), make sure the overall results are not derailed, and manage the expectations of surrounding stakeholders.

Starting with a Pilot

Rome wasn't built in a day, and neither is SOA. It is best built incrementally, project by project. By starting with an explicit SOA pilot project, your organization can learn from its successes (and failures) to increase the success of its overall SOA initiative. Before commencing your effort, there are key issues you should examine when choosing and launching a successful SOA pilot project.

Key Steps to a Successful SOA Pilot

Step 1: Identify the Goals in an Initial SOA Pilot

The pilot will provide valuable insight into SOA infrastructure that will be extremely useful as you expand to building an enterprise-wide SOA. This is a great time to experiment and learn from trial and error. Goals for this initial project should be clearly articulated, and may include:

- Developing a service that can be reused by multiple departments
- Consolidating duplicate applications into one server
- Providing a showcase of successful SOA implementation to clearly illustrate the benefits of reuse and consolidation
- Learning valuable lessons that can be leveraged in the larger SOA initiative

- Understanding the tasks involved in getting services into production as well as the day-to-day tasks required to manage SOA once it is production

Step 2: Create a Cross-Functional SOA Team

Successful SOA pilots involve the cooperation of cross-functional departments, including line of business, development, operations, security, and more. While these stakeholders may not be directly involved with a typical pilot on a daily basis, it is critical that they experience both the pain points and benefits of SOA through team participation. Regular meetings and communication will expose any concerns or territorial aspects that could inhibit SOA production.

Choose the team members wisely and consider their ability to participate outside of their daily responsibilities. Outline time commitments for both meeting participation and communication review and ask each team member to agree to the commitment up front. Additionally, create and share a communication strategy that includes a reasonable number of updates sent to each team member. Then it is important to stick to that schedule because it lends credibility to the pilot and elicits the best feedback and participation.

Moving toward an SOA will most certainly dictate a change in the way applications are developed and deployed. Most believe the team and organizational change is the toughest challenge in migrating to SOA because many people see this change as the destruction of the silo-based organizations that exist today. Therefore, the ability to pick the right cross-functional SOA team may be the most critical aspect of the pilot.

Step 3: Determine the Appropriate Pilot

In order to accurately and successfully exhibit the benefits of SOA to those who may be skeptical, you must first choose the appropriate pilot. It must clearly demonstrate the promise of SOA without causing material harm to any aspects of the business – that is, a service with the best risk/reward ratio. Early pilot success lends credibility and therefore leads to production SOA.

Consider the Following Questions When Determining an Appropriate Pilot

1. Create a new service or leverage an existing one?

New services can be easier to create initially, but wrapping a Web service around an existing legacy system may provide measurable benefits in less time

2. Select a high or low visibility pilot?

Low visibility pilots:

- Are less critical if problems arise during implementation
- Allow triage without constant scrutiny

High visibility pilots:

- Higher visibility pilots come with multiple opinions and the associated political complications
- Benefits are delivered earlier and to more departments
- These pilots determine stakeholder requirements and whether or not a visible success is necessary

3. Choose a revenue-based or back-end application?

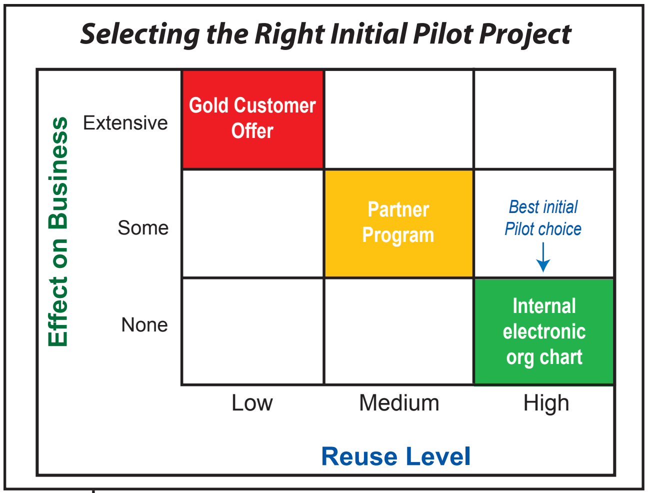
The most common pilots are often related to user-facing systems such as portals and Web sites – for example, a portal that provides a single view for customers or employees based on data gathered from one or more different systems. These are often good choices because the end results are tangible and can provide a hands-on experience with the benefits of SOA. In comparison, back-end integration, such as synchronizing data between systems, can be valuable, but it is difficult to clearly demonstrate the benefits.

4. Choose a customer-facing or internal application?

There are many pros and cons associated with choosing either a customer-facing or internal-only application. Internal services, such as one in Human Resources used to enable employees to update their personal information via a corporate intranet portal, can protect customers from potential issues, but feedback may not be relevant to wider-scale use of SOA. In comparison, a financial institution could use a currency exchange rate service for a limited volume (but highly visible) pilot.

5. Choose a transaction-oriented or query-oriented service?

Query-oriented services are used pri-



Selecting the Right Initial Pilot Project

Effect on Business	Low	Medium	High
Extensive	Gold Customer Offer		
Some		Partner Program	*Best initial Pilot choice* ↓
None			Internal electronic org chart

Reuse Level

FIGURE 2 SOA reuse initiative

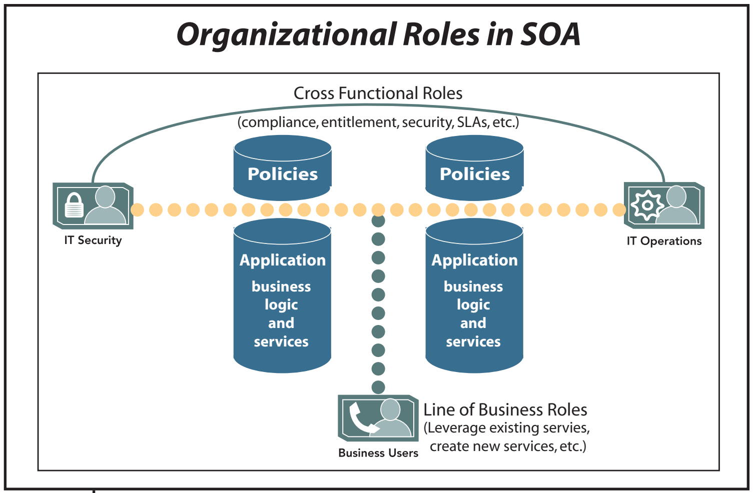

Organizational Roles in SOA
Cross Functional Roles
(compliance, entitlement, security, SLAs, etc.)
Policies
Policies
IT Security
IT Operations
Application
business logic and services
Application
business logic and services
Line of Business Roles
(Leverage existing servies, create new services, etc.)
Business Users

FIGURE 3 Organizational roles

marily for making existing data available for other uses such as retrieving lab results in a healthcare environment. In contrast, a transaction-oriented service is one that creates new information records, or triggers a new business process (such as the DMV registering a new automobile).

Transaction-oriented services are inherently more risky because problems can lead to data loss. As a result, many organizations start with query-oriented services to unlock existing data assets. Later, they extend these with transaction-oriented operations, since many services require both. For example, in telecommunications, a self-service Web site should be able to provide both self-service provisioning (transaction-oriented operations) and information on service requests, billing history, and more (query-oriented operations).

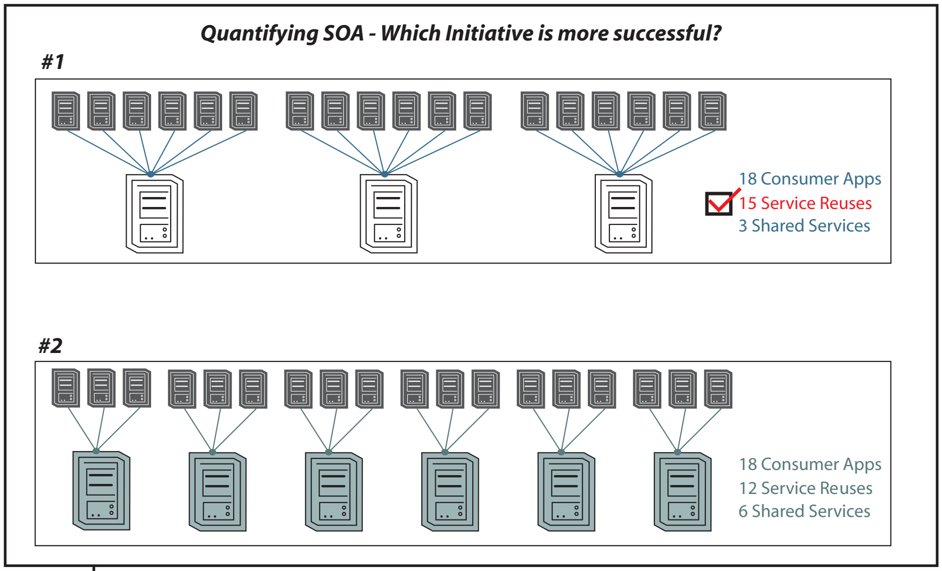


FIGURE 4 Quantifying SOA

Step 4: Quantify Pilot Results

The most important deliverable for an SOA pilot is quantified results. Senior management often requires ROI calculations as well as tangible proof of your pilot's success. Create a method for ongoing data capture, particularly if a pilot is conducted over a large period of time, so that the data is both accurate and readily available at the completion of the project. Gathering and compiling these figures is the key to budget justification for the next phase of any SOA initiative.

SOA Can Be Measured by Several Factors

Reuse of Shared Services

The number of instances in which a shared service is reused is an effective SOA measurement for ROI. Each reuse results in cost avoidance or reduction of building, maintaining, and operating a single-purpose service. In order to properly calculate ROI of shared services, some additional base metrics are needed:

- *Costs to build/operate/maintain a shared service.* This is the cost of having a shared service.
- *Cost to build/operate/maintain a single-purpose service.* This should be similar to that of a shared service if your SOA infrastructure is effective.
- *Cost to use an existing service built by someone else.* This is the cost incurred by reusing a service (the alternative would have been to build a single-purpose custom service).

Controlling this metric is critical to SOA success. The reuse of a service is effectively an integration – and so your SOA needs to be structured to control the costs of integration.

Shared Service Consumers

The number of shared service consumers (relative to total consumers) measures the breadth of SOA adoption. That is, it measures how well the "cultural shift" of SOA has permeated the organization. This is not directly correlated to SOA business benefit, but it is an important metric nonetheless.

Web Services Adoption

The number of Web services created is not actually a measurement of SOA. Instead, it primarily reflects the breadth of adoption of the underlying technology. In many cases, this metric can actually be used as a negative indicator of SOA failure. That is, if a large number of services exist, but few are reused, this may be an indication that your SOA initiative needs some revision.

Business Responsiveness

Compare the time it takes to change or add a feature to a non-SOA–based application with a similar features change to a service.

Conclusion

In addition to the aforementioned methods of measurement, every project will have its own unique business justifications and associated measurements. For example, exposing customer information via a service in order to create a self-service customer portal may be used to significantly reduce call center operations costs. These benefits will, of course, differ for each project. Even if you define a formal SOA pilot project, don't do this for the sake of moving to SOA – the project should provide value to the business.

About the Authors

As chief technology officer at Actional, Dan Foody leverages his extensive hands-on experience in enterprise systems integration software toward easing integration through Web services. He is an active participant in the Web services standards community, including WS-I and OASIS, where he spearheads Actional's contributions on the OASIS Management Protocol Technical Committee and its efforts to deliver XML-based Web services management standards. Dan is the author of various application integration standards and has contributed significantly to the OMG standard for COM/CORBA interworking.

■ ■ ■ dfoody@actional.com

Alex Rosen is the manager of the service-oriented architecture practice at MomentumSI. Alex has architected and led teams in developing solutions for large enterprise clients for content management, e-commerce, and the development of service-oriented environments. He holds a Bachelor of Science degree from the Massachusetts Institute of Technology.

■ ■ ■ arosen@momentumsi.com

Web Services – Soup to Nuts

Simplified Web services development using visual tools

When you set out to develop a new Web service, do you start banging out angle brackets right off the bat, or do you sketch out the logic of your service first? For most engineers, the graphical approach to implementing business logic is much more accessible. Only after a picture is created does the complexity of translating it into code come into play.

WRITTEN BY
ERIN CAVANAUGH

Though Web services are intended to simplify communications, developing them manually can be anything but simple. Altova has added visual Web services development and implementation capabilities to two of its most popular products, giving developers the ability to build a Web service from start to finish without becoming mired in manual code writing.

Altova provides support for developing all XML-related technologies, including Web services, in its XML Suite. By using these tools, developers can design and implement a Web service using a purely graphical approach. This article discusses this visual Web services design and implementation process and how it removes much of the complexity associated with Web services development.

Graphical WSDL Creation and Editing

Best practices dictate that designing the WSDL should be the first step in architecting Web services. By building the WSDL that will act as the Web services interface first, client and server programmers can implement their respective programming contract for accessing the service using any language or operating system, since WSDL is standards-based and programming language–neutral. That means a Web services client application can talk to a server application running on a different platform without interoperability issues. This built-in interoperability also means that the service can be reused easily. Unfortunately, given the complexity of WSDL and WSDL development, this design step is often overlooked or undertaken only as an afterthought.

Altova XMLSpy 2006 is an XML development environment that includes a graphical WSDL editor that removes much of the complexity associated with WSDL development by enabling a completely visual process. Instead of working solely with a text view, developers can build their WSDL files graphically, with full validation and editing help. This removes the

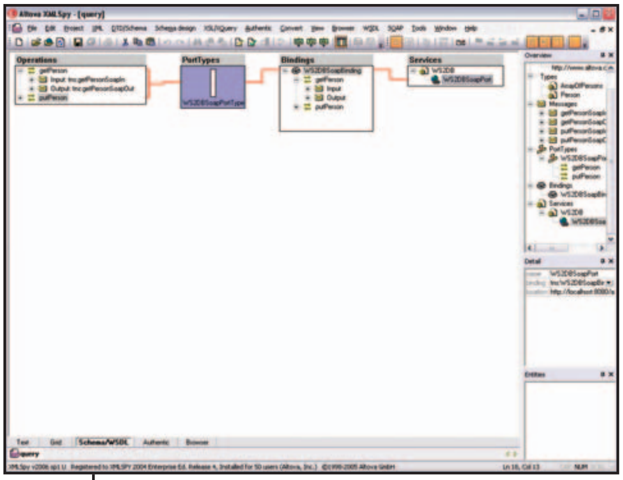


FIGURE 1 Altova XMLSpy 2006 graphical WSDL editor

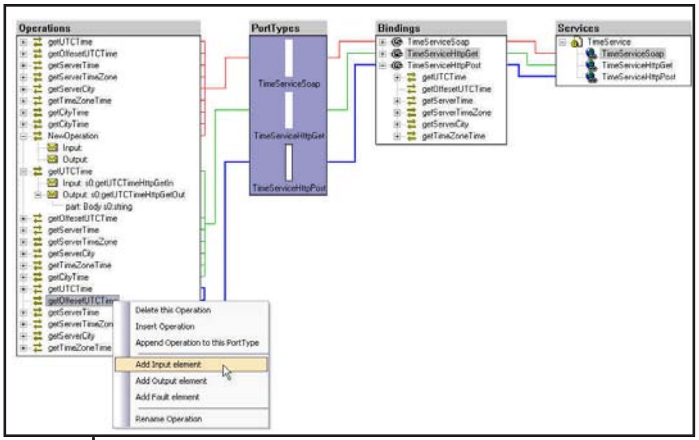


FIGURE 2 **Editing WSDL operations using the right-click menu**

complexity barrier to the design-before-implementation development approach.

The XMLSpy graphical WSDL editor in Figure 1 supports creating, editing, visualizing, and validating any WSDL file. It displays the WSDL file structure as well as the WSDL elements grouped by operations, portTypes, bindings, and services. You can manipulate the file by dragging and dropping elements, and context-sensitive windows and entry helpers provide useful editing options.

When you create a new WSDL file from scratch, XMLSpy presents you with a graphical representation that consists of a box for each type of WSDL element: operations, portTypes, bindings, and services. This visual representation allows you to easily see the structure of the WSDL and edit each portion in a logical way. Since the "skeleton" WSDL structure has already been created, you can edit each portion separately in any order – whatever makes the most sense to you. In general, to create a WSDL, the following must be completed:

- Name the service
- Associate ports with the service
- Associate a binding name, type, and location for each port
- Add operations in the operations component
- Name each operation, then add input, output, and/or fault elements
- Validate the WSDL file and correct any validation errors

There are several options and helpful features available in XMLSpy for completing these steps, as described below.

To the right of the design window, the overview window reveals the structure of the WSDL document. It displays the components of the WSDL file in a tree view in the sequence in which they appear in the code. You can expand and collapse components to reveal and hide subcomponents. Selecting a component in the overview window or from within the design itself displays that component and its properties in the details window, for example, the name, binding, and location of each port. These entries are editable by hand or by using context-sensitive drop-down menus.

The graphical WSDL display helps you immediately understand the relationships between the different components in the file and add new elements in a straightforward way. The operations component of the design window lists all of the operations in the WSDL, and each is connected to its respective port type with a color-coded line. Expanding any operation displays its input and output components, and a context-sensitive right-click menu lets you append, insert, or delete operations (Figure 2), as well as add input and output messages to any operation. To associate a port type with an operation, simply drag the operation to the desired port – the same is true for binding and services associations.

In addition to the colored links, clicking any port type in the port type component highlights all associated operations in the operations component and in the overview window. Each port type is also linked to the associated binding with the same color-coded line, and bindings are linked to their respective services in the services component. At a glance, you can tell which operations are associated with which port types, bindings, and services, and you can adjust these associations by dragging and dropping elements.

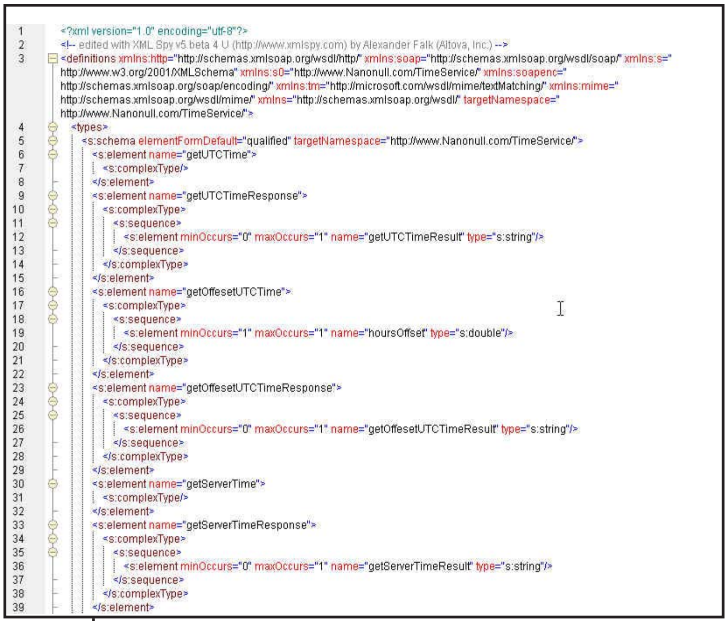
Selecting a binding lets you view or edit all relevant parameters in the details window:

- The name attribute
- The type attribute, which refers to the specific port type
- The type of binding: SOAP, http/GET, http/POST
- The transport URI, which defines the transport URI
- The style attribute: RPC or document

You can also add, edit, and delete these connections using drag-and-drop and context-sensitive menus (right click, tool bar, or drop down – whichever you prefer).

In addition to the visual WSDL design view, XMLSpy also includes a graphical XML Schema editor. This means you can switch to and edit any schemas referenced or imported in the WSDL very easily, using a similar visual paradigm.

While you're working in the graphical view, the corresponding XML markup is built



```
<?xml version="1.0" encoding="utf-8"?>
<!-- edited with XML Spy v5 beta 4 U (http://www.xmlspy.com) by Alexander Falk (Altova, Inc.) -->
<definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:s="
http://www.w3.org/2001/XMLSchema" xmlns:s0="http://www.Nanonull.com/TimeService/" xmlns:soapenc="
http://schemas.xmlsoap.org/soap/encoding/" xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:mime="
http://schemas.xmlsoap.org/wsdl/mime/" xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="
http://www.Nanonull.com/TimeService/">
  <types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://www.Nanonull.com/TimeService/">
      <s:element name="getUTCTime">
        <s:complexType/>
      </s:element>
      <s:element name="getUTCTimeResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="getUTCTimeResult" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="getOffesetUTCTime">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="hoursOffset" type="s:double"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="getOffesetUTCTimeResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="getOffesetUTCTimeResult" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="getServerTime">
        <s:complexType/>
      </s:element>
      <s:element name="getServerTimeResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="getServerTimeResult" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
```

FIGURE 3 **Snippet of code that was autogenerated from the WSDL design in Figure 2**

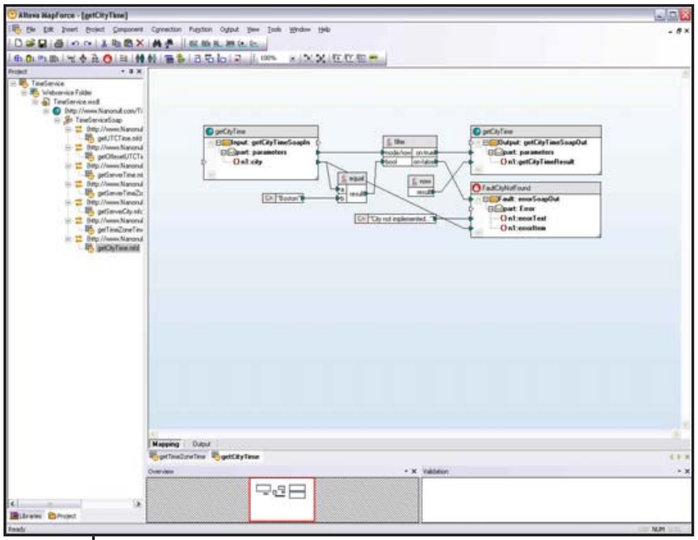


FIGURE 4 **Altova MapForce mapping of a WSDL operation defined in Figure 2**

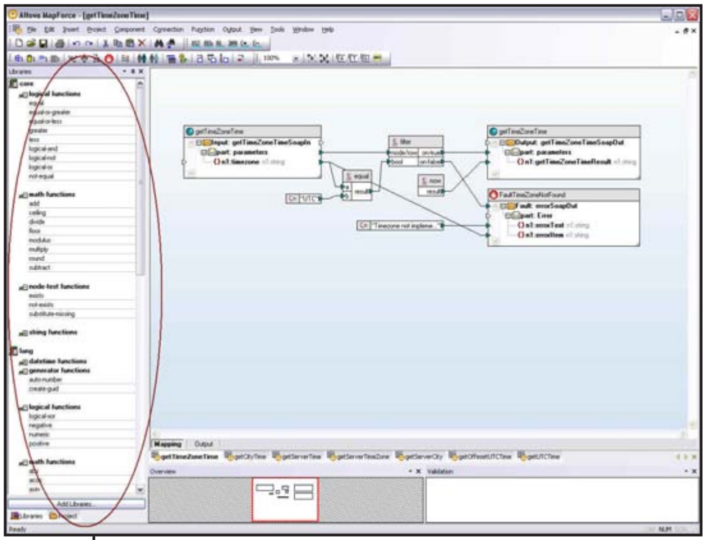


FIGURE 5 **MapForce drag-and-drop data-processing function library**

behind the scenes and is available under XMLSpy's text view tab, and you can, of course, edit the text directly in that view (Figure 3).

One last note about XMLSpy is that it also includes a WSDL documentation generator that will output documentation for any standards-conformant WSDL file. Documentation can be in HTML or Microsoft Word and is useful for giving business partners, other developers, or customers a detailed overview of the Web service interface. This feature is also useful for understanding WSDL files that are generated by other platforms.

Overall, designing WSDL using a graphical paradigm makes WSDL development more accessible and productive, which in turn makes the Web services implementation phase more straightforward.

Visual Web Services Implementation

Once a WSDL is defined, implementing the Web service that it describes involves writing the code to access the data required for each transaction. Given that even simple Web services can require thousands of lines of code, this process quickly snowballs. Altova MapForce 2006 adopts the same visual design strategy used in the XMLSpy WSDL design view for building Web services.

MapForce itself is a multifaceted tool. It's a visual data mapper with full support for XML, any relational database, flat files, EDI messages (EDIFACT and ANSI X12), and with the latest release in October 2005, WSDL. For converting data and building data integration applications, MapForce allows you to map any combination of the above data formats, insert data processing rules, then immediately convert data to the required form or generate code to implement the mapping programmatically.

With its new WSDL mapping support, MapForce automates the Web services implementation process by allowing you to connect data sources and operations visually and to then autogenerate the code required to implement the service server-side.

When you open an existing, standards-conformant WSDL file (like one created in XMLSpy), MapForce creates a project view that displays the WSDL and all of the operations that it includes. This is shown in the left-hand panel of Figure 4. Double-clicking any operation opens its input and output schemas as graphical components in the mapping design window.

To implement each operation, you have to supply the input data or connect a data source. The MapForce function library, which you can toggle to by clicking its tab in the project window, lets you drag functions onto the mapping design to supply data values with constant components, or to filter and manipulate data before returning it as a response. The data processing function library (highlighted in Figure 5) contains an extensive array of functions: string, mathematical, Boolean, language-related, logical, node-tests, etc., that you can drag and drop onto the mapping design to process source

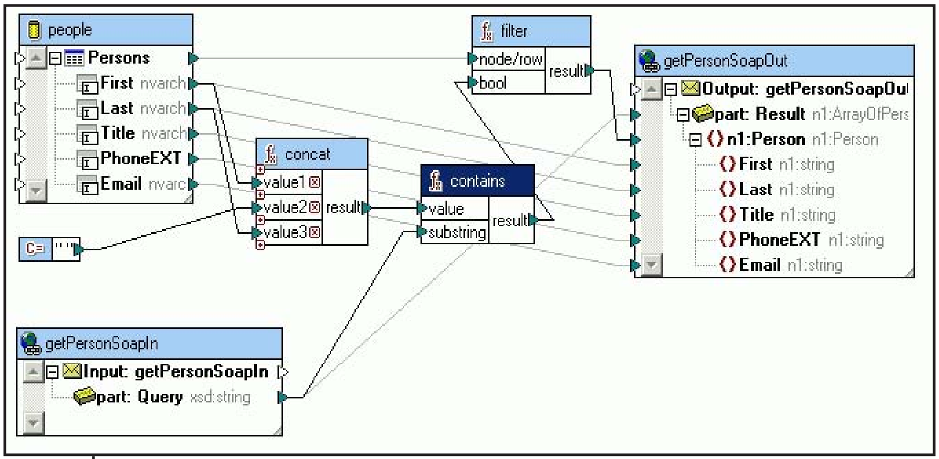


FIGURE 6 **A WSDL mapping utilizing a database source**

data. You can create and save user-defined functions as well.

Figure 5 shows a WSDL operation for which the data supplied is a constant value. You can also connect dynamic data sources in XML, database, flat file, or EDI formats. This way, you can access and combine data from multiple disparate systems within the same operation. This is still a visual drag and drop process – you insert the required data source(s), and MapForce lists their elements in a hierarchical tree view. Then you map the WSDL operation by dragging connecting lines from elements in your data sources, through the data processing functions, to the corresponding targets in the response component. That's all there is to it. A Web services mapping that uses a database source is shown in Figure 6.

After mappings are defined for all operations in given WSDL, MapForce will autogenerate code for the entire project in either Java or C# to implement the service on a server. Alternatively, you can generate code for one or a few of the operations in the WSDL separately. All that's left to do is to compile the generated code and place it on a Web server.

There are several advantages to using this tool-generated code. First, it greatly simplifies Web services development by allowing you to concentrate on implementing the business logic of a service – instead of manually writing thousands of lines of complicated code. Code generation based on visual design also ensures that the code is written consistently across the entire project, since it's produced according to industry standards and globally defined parameters, rather than having multiple engineers manually writing the code. This high degree of software code consistency ensures that code is standards-conformant, interoperable, and reusable.

Complete Web Services Development

By using Altova XMLSpy 2006 and MapForce 2006 together, you can create Web services from start to finish in a visual manner. This speeds development and reduces the occurrence of errors that may be introduced by manual coding.

By creating a WSDL file in XMLSpy and then building the corresponding Web service and generating program code in MapForce, you can effectively build a complete Web service without having to write a single line of code. All that's left to do is compile the code generated by MapForce and deploy it on a server. You've gone from soup to nuts in a completely visual manner, with increased standards-conformance, productivity, and code quality.

XMLSpy and MapForce are available separately or as part of the Altova XML Suite. Free trial versions are available at www.altova.com/download.

About the Author

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

erin.cavanaugh@altova.com

Choosing the Best Testing Tools to Increase Project Productivity

Start preventing errors throughout the software development life cycle

■ The primary mission of information technology is to improve business processes and increase profits. Companies are constantly rethinking and struggling with how to use IT to a competitive advantage, reduce IT operating and maintenance costs, and reduce the total cost of ownership… all while attempting to deliver increased value.

WRITTEN BY
DR. ADAM KOLAWA

Most of these problems can be traced to the same source: the struggle to make software work – without incurring unreasonable costs. Thus, it all seems to lead back to cost, which raises the question of why software development is so costly. Most people in the industry would agree that low IT productivity is the culprit here. But why are IT teams, with all their expertise and hard work, suffering from low productivity? The root cause of this low productivity can be traced to errors that result from mistakes made throughout the software development life cycle. These errors include everything from performance errors, to security errors, to misimplemented functionality, to errors that crash an entire system. They essentially stifle IT teams' ability to produce working software in a reasonable time and at reasonable costs. In fact, if you look at virtually any IT team, you will see that its team members spend about 80 percent of their time chasing and fixing bugs, and only about 20 percent of their time on tasks that deliver value and improve the business. This practice is far from productive.

The logical reaction, then, is to prevent errors in order to increase project productivity, but how can IT teams prevent errors? The answer lies in choosing the best testing tools available that will help your team be productive. However, while reading this article, it must be noted that not all tools are created equally. Simply testing your code is not enough. You must use only the tools and solutions that aim to prevent errors rather than simply detecting them.

In the case of Web services, testing tools and solutions, such as Parasoft's SOAtest, can increase your team's productivity by preventing errors early in development, and in doing so, improve quality and reliability, and accelerate time to market. The sooner you detect a problem, the easier it is to fix it, thereby leaving your team more time to code and be more productive. Therefore, you must employ a wide variety of tools and testing techniques that immediately and thoroughly exercise Web services and check their reliability. This article will explain issues specific to Web services and will illustrate how to choose the right testing tools that can increase productivity and ultimately ensure complete Web service functionality, interoperability, and security.

Learning from the Automobile Industry

Many other industries, such as the automobile industry, have also struggled with low quality, high costs, and low productivity as a result of human error. These industries recognized that although mistakes cannot be eliminated, they could, indeed, be controlled. Industries then modified their production lines to prevent as many errors as possible. By preventing scores of errors from ever entering the products, they addressed their most critical industry problems and were able to remain viable and productive industries.

The software industry still has not learned this lesson. Many people do not think that error prevention is even possible in the software industry; they believe that because each piece of software is different, the lessons learned from working on one piece of software cannot be applied to other pieces. Thus, instead of trying to prevent errors from entering software,

> “ Any mission-critical Web service must be strictly tested and verified as functioning correctly, 24 hours a day, seven days a week – no exceptions ”

SPECIAL OFFER!
$295 REGISTRATION RATE
(A $600 SAVINGS!)
CALL 800-605-5849 FOR DETAILS

IDC
Analyze the Future

Service-Oriented Architecture Forum

Enabling IT and Business Alignment

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

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

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

SPECIAL FEATURES:

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

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

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

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

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

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

Diamond Sponsor

IBM.

Sponsors:

Blue Titan

sonic SOFTWARE

Media and Association Partners:

ADTmag.com ES Enterprise Systems NetworkWorld SupplyDemandChain

THE WALL STREET JOURNAL. PRINT & ONLINE WebServices JOURNAL WITI build. empower. inspire.

the software industry tries to test errors out of software. In the case of Web services, we build a service, then we attempt to use testing to determine whether the service works and finally remove any errors that the testing process exposes. Throughout this process, we cross our fingers and hope that the most insidious and embarrassing problems will be identified before the release. However, a consideration of the number and impact of software errors suggests that this "quality through testing" approach is not yielding the desired results.

> "The right testing tools can provide improved productivity and labor savings"

This belief that testing can create quality software systems is a fundamental problem in the software industry. We don't think of the whole process of building and deploying software in a way that would prevent errors because we don't believe that it can actually be done. Yet, this error prevention approach is not only possible; it's necessary, provided the right testing tools and solutions are utilized.

If the software industry is serious about reducing the error rate and resolving the issues that stem from errors, we can't afford to continue hoping that our current approach to testing will miraculously start yielding quality software. Instead, we need to follow in the footsteps of other industries and start preventing errors throughout the software development life cycle. Therefore testing, which is important for any development project, is even more crucial for Web services. Extensive testing of Web services, particularly those that are externally facing and business-critical, is essential to securing the enterprise from significant business risks. A down time of an hour can not only cost substantial losses in revenue but, more important, the perceived lack of quality and reliability of the company in general. Any mission-critical Web service must be strictly tested and verified as functioning correctly, 24 hours a day, seven days a week – no exceptions.

Common Web Service Testing Problems

There are many complexities specific to, and inherent in, Web services that complicate testing and lead to low productivity. However, before delving into specific technical issues, it would be wise to first examine the general testing inefficiencies that can occur in any Web service, and how the proper tools can prevent these problems from occurring. There are a number of testing inefficiencies that plague companies and cause severe business implications such as extending project timelines and increasing project costs. Moreover for each of these problems, viable testing tools and solutions are available, such as Parasoft SOAtest – an automated solution for testing Web services and increasing the productivity of your development team.

Inefficient Use of Developer Assets

Organizations continuously re-create the same test cases at different points of the development process. Developers create tests to ensure the functionality of their Web services, however these tests are not leveraged in the downstream process. This duplication of effort is not only inefficient, but also introduces an opportunity for errors to be injected.

The right testing tools can provide improved productivity and labor savings. With automated generation of test cases, testing tools can save significant time and resources, as opposed to the manual processes that may be currently employed.

Heterogeneous Testing Platforms

By re-creating test cases on different platforms, there is a significant risk of extending the time it takes to verify functionality. For example, testers who discover errors in one format may go back to the developers who then try to reproduce these errors in their own format. This rework loop is time consuming and inefficient.

Choose a uniform solution for Development, QA, and Performance teams. It is best that test cases be reused, combined, and extended across teams, thus providing seamless transfer of knowledge and test case data. Functional and unit test cases created by development and QA should be used by the performance team to generate scenarios for use in load and performance testing, thereby eliminating the need for extensive script writing and script maintenance. Such practices save time and improve accuracy of test creation/execution.

Development and Maintenance of Homegrown Tools

Some IT teams may create their own tools, such as client emulators, in-house in order to facilitate functional testing. At the same time, a load generation tool may be purchased that requires proprietary scripts to be written. The creation and use of such homegrown tools and scripts not only adds overhead to the project, but such tools are also impossible to maintain as Web service complexity and industry standards evolve and mature.

Choose a single tool that can automatically generate a suite of test cases using WSDL and HTTP traffic. In addition, the ability to leverage tests from unit testing through load testing allows more test cases to be generated – ensuring greater coverage and quality of the service while minimizing business risks of production failure or production-level error.

Incomplete Testing

The scope of testing provided by in-house applications is generally limited and does not cover all of the aspects needed to verify a Web service. Aspects such as security, interoperability, functionality, reliability, and availability are not verified due to these limitations. As a result, businesses face an exponential increase in financial risk.

For example, rather than testing a Web service directly at the code level, organizations may utilize some GUI-based client to consume the Web service. However, dependence on GUI-based testing of a Web service is limited and restricted to only the types of messages that the GUI-based client is built to generate. With most Web services, it is impossible to anticipate exactly what types of requests clients will send. Real-world partners and consumers of a Web service are certain to send different types of requests, negative inputs, and bad data to the Web service that the GUI-based client has not even considered. In addition, maintaining and expanding a GUI-based testing application would result in numerous and unnecessary productivity losses in order to perform only limited, indirect testing of a Web service. In order to completely and thoroughly test a Web service, you must be able to emulate many of the types of clients that might access the server, and verify that the server will behave as expected in relation to any type of client request.

The best testing tools increase the scope of testing by providing the flexibility to manipulate messages in order to directly and thoroughly test the Web service. Tools should be able to perform code-based testing of the Web service and isolate and test the data at the API and message layer of the service. By testing the API and services layer, developers can verify whether the Web service can handle a wide range of request types and parameters. By checking for the conditions and inputs that are not expected, you enable more thorough tests for what cannot be foreseen. By performing such testing at the unit and application level, you can quickly and easily identify and correct any weaknesses before security breaches have the opportunity to occur.

Summary

As seen throughout this article, there are countless opportunities for things to go wrong during Web services development – a slight mistake in any component or interface will cause problems that ripple throughout the system. Developers must ensure that each part of the system is reliable, and that all of these parts interact flawlessly and securely.

As organizations rely more heavily on Web services to drive their business processes, it is critical for Web services developers to have a proven solution to test the complex layers of Web services and to create an efficient and bulletproof testing process without being riddled with the common problems discussed earlier. The Parasoft Web Services Solution not only provides clear and practical guidelines for testing, but also provides a cohesive team workflow that makes the most efficient use of project assets across the organization. To learn more about Parasoft Web Services Solutions, or any of Parasoft's Automated Error Prevention Solutions, please visit www.parasoft.com/SOAtest_WSJ.

About the Author

Dr. Adam Kolawa is cofounder and CEO of Parasoft, a vendor of automated error-prevention software and services based in Monrovia, CA. Dr. Kolawa, who is the coauthor of *Bullet-proofing Web Applications* (Wiley, 2001), has written and contributed hundreds of commentary pieces and technical articles for publications such as *The Wall Street Journal*, *CIO*, *Computerworld*, *Dr. Dobb's Journal*, and *IEEE Computer*. He has also authored numerous scientific papers on physics and parallel processing. He holds a PhD in theoretical physics from the California Institute of Technology.

■ ■ ■ ak@parasoft.com

MagooClient 2.1 XML Messaging Client

A lean and mean client for XML-based business processes

When I first looked over MagooClient from Magoo Software, it was difficult to categorize. I expected that it would be another composite application builder, but that's not what I found. Instead I found a tool that not only allows users to interact with business processes, but that also becomes part of the business process itself. MagooClient can not only reach out and perform Web service requests, but it is designed to also be inserted in the path of XML messages.

WRITTEN BY
PAUL T. MAURER

XML Messaging Client

Magoo Software describes MagooClient as an XML messaging client. That is very fitting, since it has been modeled to operate in a fashion similar to an e-mail client. As you can see in Figure 1, the MagooClient main window looks very much like a stripped-down e-mail client. Users compose messages via forms that are constructed on the fly. The messages can hang around in the "drafts" folder until the user is ready to send them to a particular destination. Once sent, messages are moved to the "sent" folder.

The "inbox" is where inbound messages land. What types of messages would I expect to see here, you might ask. Well, the most obvious type would be responses to outbound synchronous Web service calls. More interesting though, is the fact that MagooClient can receive XML messages over various transports. It even has an embedded HTTP server and can listen for inbound XML messages. Users can inspect and edit these inbound messages and forward them on to other services.

Essentially MagooClient can initiate business processes by creating and sending XML messages to services. It can allow for review, validation, and editing of messages, and finally, it can route messages and perform exception handling.

Let's peel the onion and look deeper.

Message Types

MagooClient maintains a message type catalog. Although the software can display any XML as a form, the catalog enables it to support message creation, validation, and editing.

MagooClient provides a wizard that simplifies population of the catalog. It allows import of either XML Schema or WSDL directly from a file or over the network from an arbitrary URL.

The MagooClient uses its own proprietary XML Schema capability for efficiency, but allows validation against the Apache Xerces parser for comparison. The WSDL and schema support are excellent. I keep a few large WSDL and schema around from old projects that I typically use as litmus tests for new tools. They are valid, but have caused other tools to fail. I was able to import them without trouble into the software.

Note that message types need not necessarily exist in the catalog before message creation can occur. The software has a nifty feature that allows a message to be created from a WSDL or XML Schema on the fly. This process automatically populates the catalog.

Forms

MagooClient automatically renders forms from the XML content. There is no form design here; it's neither supported nor required. MagooClient does a beautiful job of creating a usable layout.

Figure 2 displays the message window. Note that the form is flat and arbitrarily long based on the size of the XML, but navigation is made possible by tree representation of the message, which is displayed in the navigation panel to the left of the window.

Clicking on nodes in the tree navigation panel scrolls the appropriate form section into view. Clicking on a field in the form causes the tree navigation panel to highlight the appropriate node. It is a nice clean way to move around and edit the document.

I have always been a fan of tightly typed

Magoo Software Limited
2 Princes House,
52 Kensington Park Road,
London, W11 3BN

Web:	www.magoosoft.com
E-mail:	info@ magoosoft.com

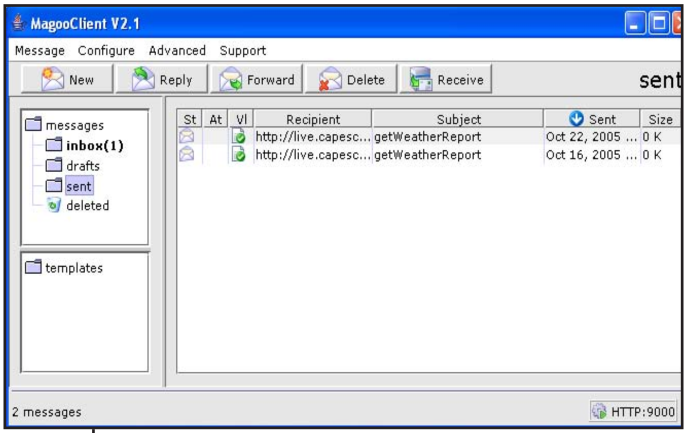
Licensing Information

Per User:	$500
10 Users:	$2,000

Note: MagooClient is free to students/researchers attending academic institutions

Testing Environment

OS:	Windows XP Professional (Service Pack 2)
Hardware:	Intel Pentium M Processor (1300MHz) 1.29 GHz – 1 GB RAM



St	At	Vl	Recipient	Subject	Sent	Size
			http://live.capesc...	getWeatherReport	Oct 22, 2005 ...	0 K
			http://live.capesc...	getWeatherReport	Oct 16, 2005 ...	0 K

2 messages HTTP:9000

FIGURE 1 MagooClient main window

schema definitions. These not only allow for much of the validation to be provided by the schema automatically, but also have the added benefit of supporting automated form engines such as MagooClient. MagooClient leverages the XML Schema information in order to render and validate forms. For example, enumerated values are presented as drop-down lists, value ranges as radio buttons or spinners, and input fields are sized from the schema accordingly.

It is one thing to create forms based on static structures, but XML Schema allows for variable message structure. The software allows for adding, removing, or replacing elements at any node in the tree where deemed valid by the schema. This is done by right clicking on a node in the tree navigation panel. A pop-up menu will list the valid schema types that can be inserted.

Transports

Remember earlier I stated that MagooClient can become part of the XML business process. It does this by supporting receipt and transmission of messages over a number of transport mechanisms. Transports map the ability to send and receive messages to a particular protocol or vendor implementation. This allows a message to be sent simultaneously using multiple transports to various destinations. Transports, just as message types, must be configured in the catalog. Currently the software supports HTTP, POP3/SMTP, and JMS as transports. The HTTP support is relatively straightforward, so I won't go into it here.

As mentioned earlier, the software was modeled on an e-mail application and although atypical, e-mail as a transport can be a practical, cost-effective form of store-and-forward XML messaging in some instances. The MagooClient can support multiple e-mail accounts simultaneously. The software can be configured to pull messages from the mail server upon request, or a polling interval can be configured in order to retrieve messages automatically. I was able to quickly route a set of messages through one of the many free Internet e-mail services and retrieve them again as a test.

MagooClient also supports transport over the Java Message Service (JMS). This allows the software to communicate with enterprise-class messaging providers. The MagooClient uses the asynchronous listener model to ensure that messages are received as soon as they arrive at the provider.

MagooClient provides a nifty transport monitor that can be launched from a menu. Figure 3 shows the monitor, which logs all communication to and from the application. This allows the user to view all of the low-level message details such as handshake, properties, and headers.

Scripting

Embedded within MagooClient is the Mozilla Rhino scripting engine. Rhino is an open source JavaScript implementation with extensions that add native XML support.

The software provides an integrated script editor with drag-and-drop capabilities from the tree panel of the message window. Scripts can create new messages, manipulate existing messages in the environment, call out to external Web services, and interact with the user via a built-in dialog object.

Essentially, scripts can be used as macros and invoked from a pop-up menu, as extended validation rules to check complex interfield relationships, or they can provide dynamic validation and population of form fields using external callouts. Also, message types can be configured so that scripts automatically run on receipt or when open.

Rules

MagooClient supports rules that can either route messages on to a destination or file the message in a particular folder. Rules are basically a fixed set of conditions that when satisfied, fire off the configured routing or filing operation. The conditions vary from matching on various addressing information to message type, validity, transport, and direction.

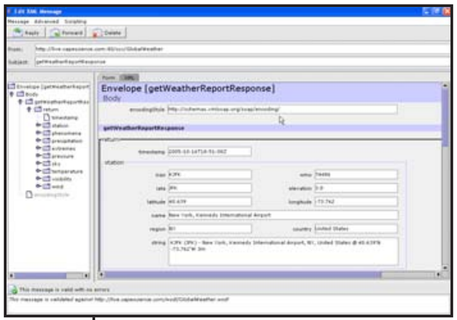
Conclusion

Magoo Software has leveraged a simple e-mail–style messaging model to deliver a product that is easy to learn and straightforward to use, yet still powerful. MagooClient XML support is excellent. The software's ability to support multiple transports and destinations, dynamically render forms, and apply scripts and routing rules makes this product worth a look. I definitely recommend that you check out MagooClient.

About the Author

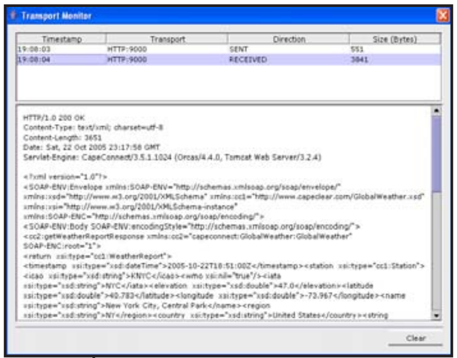
Paul T. Maurer is a principal in the financial services practice of a leading consulting services company.

paul@paulmaurer.net



Envelope [getWeatherReportResponse]
Body
getWeatherReportResponse

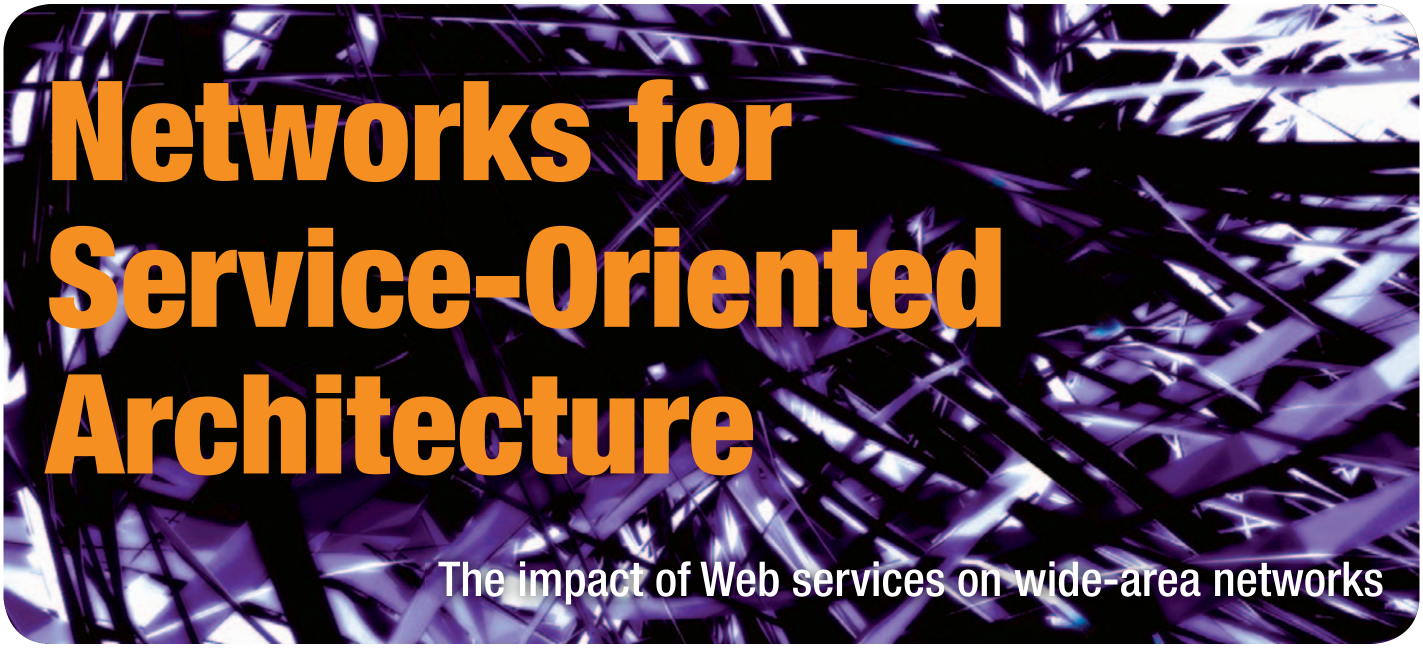
FIGURE 2 MagooClient message window



Timestamp	Transport	Direction	Size (Bytes)
19:08:03	HTTP:9000	SENT	551
19:08:04	HTTP:9000	RECEIVED	3841

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 3651
Date: Sat, 22 Oct 2005 23:17:58 GMT
Servlet-Engine: CapeConnect/3.5.1.1024 (Orcas/4.4.0, Tomcat Web Server/3.2.4)

<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:cc1="http://www.capeclear.com/GlobalWeather.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<cc2:getWeatherReportResponse xmlns:cc2="capeconnect:GlobalWeather:GlobalWeather"
SOAP-ENC:root="1">
<return  xsi:type="cc1:WeatherReport">
<timestamp  xsi:type="xsd:dateTime">2005-10-22T18:51:00Z</timestamp><station  xsi:type="cc1:Station">
<icao  xsi:type="xsd:string">KNYC</icao><wmo xsi:nil="true"/><iata
xsi:type="xsd:string">NYC</iata><elevation  xsi:type="xsd:double">47.0</elevation><latitude
xsi:type="xsd:double">40.783</latitude><longitude  xsi:type="xsd:double">-73.967</longitude><name
xsi:type="xsd:string">New York City, Central Park</name><region
xsi:type="xsd:string">NY</region><country  xsi:type="xsd:string">United States</country><string
```

FIGURE 3 MagooClient transport monitor



Networks for Service-Oriented Architecture

The impact of Web services on wide-area networks

The widespread adoption of distributed computing has been prognosticated for many years. Finally, a critical mass has been created of both enabling and demand trends that will ultimately realize the promised wave of distributed computing applications. This wave of distributed computing will have a profound impact on businesses and government agencies in many ways – and one area that will be particularly affected is wide-area networks (WAN).

WRITTEN BY
MICHAEL MULLALEY

SOA-Enabling Trends

Web services–based service-oriented architecture (SOA) is an evolved form of distributed computing that can trace its roots to earlier architectures such as CORBA and JINI. However, XML (the key protocol for Web services) has a level of sophistication and widespread acceptance that CORBA and JINI never achieved. Also, there's the ubiquitous nature of Web services tools, APIs, IDEs, and SOAP stack implementations. They're everywhere.

All major operating system vendors are on board and have, or soon will have, major releases that implement Web service entry points and tools. A major milestone for SOA adoption will be when Microsoft finally releases Vista (formerly known as Longhorn). Vista has been called the Web services release for Windows. It's now in its beta 1 stage (what Microsoft has previously called release candidate 2). Before long we should see the market-dominating operating system join Linux, Solaris, and HP-UX in enabling SOA-based applications.

Middleware vendors have been busy publishing their most popular applications services in Web services registries. For example, Oracle recently claimed that they have published over 400 services in their "Fusion Service Registry" – their name for a Web services registry. Oracle and other middleware vendors are devoting millions of people-hours to publishing more. These published services will be the building blocks for countless end-user applications, so they need to be rock solid. Middleware vendors are being particularly careful about testing, interface protections, metadata, etc.

Among major software vendors, SOA is the biggest initiative. These initiatives include:

- Oracle's Fusion Strategy
- Microsoft's Windows Server System
- IBM's WebSphere Software Platform
- BEA's WebLogic Platform
- SAP's NetWeaver
- Sun's Sun Java System
- Novell's exteNd Enterprise Suit

Operating system and middleware vendors have not just been quietly implementing Web service entry points and tools for their next major release. They have been actively training software developers throughout the world to use SOA. For many years now, software vendor developer conferences have had significant portions dedicated to SOA. Also, there are now what seems like libraries of books devoted to SOA, XML, and Web services.

Perhaps most important is the software development culture. There is an ongoing battle for the hearts and minds of software developers, and vendors are competing based on who has the best SOA tools. This is creating a culture within the software profession that dictates that anything other than SOA-based applications is passé.

SOA Demand Trends

SOA is not just a "build it and they will come" story. There are overwhelming-demand side reasons for increasing interest in SOA-based applications. Here are some of the major-demand side reasons, listed in order of priority:

1. ***Business Agility:*** Almost all business managers contend that business agility is directly proportional to business profitability. SOA enhances business agility by hastening the ability to respond with new and updated applications as business conditions change. This faster time to market for new business processes translates into a major competitive advantage. It's no wonder then that business managers – much more than IT managers – are driving today's deployments of SOA-based applications.
2. ***Information Access:*** Previously, businesses had to engage large system integrators such as Accenture, EDS, SAIC, and Lockheed to write gateway programs to get access to valuable information within legacy systems. These expensive gateway programs handcuffed many organizations. Every time a change (configuration, software revision, etc.) to either side of the gateway occurs, the system integrator needs to be reengaged to update, retune, and debug the gateway. (Typically, only the original system integrator has sufficient knowledge to update the gateway, so gateway programs are an endless drain on IT budgets and a job for life for the system integrator.) Through standard interfaces, SOA promises to minimize, if not eliminate, these resource-sucking gateway programs. (Insurance companies with valuable data locked up within old COBOL and FORTRAN applications are ideal candidates for exploiting SOA interoperability.)
3. ***Lower Development Costs:*** Software development is more economical with Web services. SOA standardizes what has been an informal practice within the software profession of reusable code. With standardization, developers have dramatically better access to proven software and can focus on their value-add rather than on rewriting what has previously been solved. Furthermore, the proliferation of Web services tools by operating system and middleware vendors greatly simplifies integration of published Web services.

Use of Web Services	Percentage of Respondents
We currently make extensive use of Web Services	33.7%
We currently make only moderate use of Web Services, but it is likely that we will significantly expand our usage	32.0%
We currently make only moderate use of Web Services, and that it is unlikely to change	18.6%
We don't currently use Web Services, but it is highly likely that we will in the next 12 months	5.2%
We don't currently use Web Services, nor do we intend to in the next 12 months	10.5%

TABLE 1 **Percentages of projected Web services involvement**

SOA Adoption

It's valuable to see where SOA is receiving considerable market traction. Examples abound within many industries, including government, financial services, and healthcare:

- ***Government:*** Fratricide can account for almost 20 percent of all US military casualties (e.g., an Air Force ordinance dropping on US Army infantry). Anything that reduces fratricide has the highest military priority. Many within the Department of Defense (DoD) are convinced that SOA will lead to better interoperability and information sharing among disparate Army, Navy, Marine Corps, Air Force, and Coast Guard systems. SOA can better enable their vision of "Net-centric warfare." Thus as it was with the Internet, the DoD is one of the earliest adopters of SOA technology.
- ***Financial Services:*** Many financial institutions believe the key to success is finding more profitable customers faster. Because SOA promises more rapid application development and better interoperability between heterogeneous systems, many financial firms are embracing SOA technology.
- ***Healthcare:*** Hospital administrators believe that better patient care can be given with improved information sharing among healthcare professionals. Also, the likelihood of error and their exposure to malpractice claims can be reduced. (Remember: avoiding only a handful of malpractice lawsuits can more than justify the cost of IT investments.)

A 2005 Kubernan survey indicates that almost 90 percent of enterprises will employ Web services within the next year – almost two thirds significantly (see Table 1).

SOA Impact on WANs

SOA governs how internal and external systems interact. As these systems are developed, they are published as Web services that can be used by others. SOA defines how Web services communicate with each other. This communication can involve either simple data passing or two or more services coordinating an activity. Web services can be combined to create new applications and even more Web services. As Web services proliferate within organizations, they will have four major networking effects:

- **Site-to-site traffic volume will grow exponentially.** Today, applications are normally executed within a single physical location. When Web services are used, code is executed at the location of the particular Web service. Since new applications will be created from some original code and a collection of Web services, code execution will be distributed – typically among multiple sites. Another element to the traffic growth is the employment of the XML protocol for inter–Web services communications. XML is a particularly verbose protocol that can take two to 10 times as many bits to send the same information as a more frugal protocol. As a consequence, a cottage industry of XML accelerator products has arisen (e.g., vendors such as Cambridge, MA–based Data Power).
- **Low latency will be an increasingly critical network characteristic.** Applications based on Web services depend more on machine-to-machine communications, and machine-to-machine interactions are always the most time-sensitive traffic on networks. Applications today mostly execute on a single server; interprocess communications are

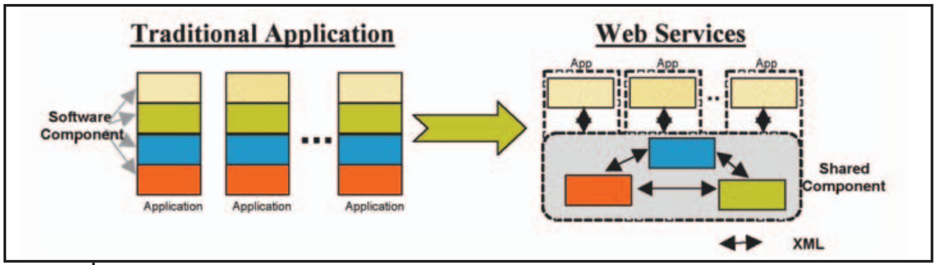


FIGURE 1 **SOA effect on WANs. With Web services, communications between the software components of applications are now transmitted across the network**

handled by the server's system bus. When processing is distributed with Web services, interprocess communications are carried over the network. That's a lot of time-sensitive traffic now traveling over the WAN. With SOA, network performance will directly affect the end user's experience.

- **Less predictable traffic patterns.** Web services can be combined in often unpredictable and complex ways. The possible combinations are endless. Network flexibility will be an increasingly important characteristic.
- **High priority packet identification will become increasingly difficult.** It will be almost impossible to ascertain when a particular Web service is being used as part of a mission-critical business process. For example, imagine an arbitrage program trading on the minor fluctuations of crop futures at various exchanges – one of the most time-sensitive applications among financial firms. It's not unreasonable to imagine a crop futures arbitrage program referring to a weather service. Most network administrators would never label communications to and from a weather program to be high priority. Yet, there's a scenario with SOA where that low-priority weather service could be in the critical path of a high-priority arbitrage program.

Many of today's packet-based networks rely on Quality of Service (QoS) to insure timely delivery of delay-sensitive traffic. QoS only works if the network can easily identify high-priority packets. What we're seeing with SOA's ascendancy is the eventual obsolescence of QoS as a viable networking technique. (There is only one way that QoS could work with a proliferation of SOA-based applications: deep packet inspection, which has significant performance, cost, security, and application transparency issues.)

Mission-Critical WANs

SOA enhances business agility by allowing quicker deployment of new and improved business processes, and applications development is more economical with SOA. Thus, businesses will be deploying more applications. Since SOA-enabled applications will tend to be for business agility, those new applications will usually be mission critical. And as we've discovered, SOA-enabled applications are very dependent upon the wide area network (WAN). Hence, the WAN is becoming more mission critical.

With this increasing adoption of SOA, it's no surprise that we're seeing more businesses and government agencies interested in carrier-grade network equipment. Some network characteristics that are in increasing demand by enterprises are:

- **More sensitivity to network downtime.** As businesses deploy mission-critical, SOA-based applications, their cost of network downtime can exceed $1 million/hour. They need high-availability network solutions that have proven themselves in production environments with five and six 9s availability. (Bear in mind that five 9s equate to five minutes of downtime every year, and six 9s equate to 31 seconds per year.)
- **Applications transparency.** Business agility is enhanced when you don't have to tune and debug your applications for the network. Some organizations spend upwards of 40 percent of their IT budget on tuning and debugging applications – well in excess of what they spend on networks. Moreover, tuning and debugging can be a constant chore if the network is not applications transparent. Most organizations experience a constant stream of version updates, bug fixes, service packs from their operating system, and middleware and application software vendors. Having to retune and debug for an applications-aware network with all of those software changes can easily become overwhelming.
- **Low latency.** Just as carriers have learned to transport traffic at the lowest possible layer, businesses and government agencies are learning that layer 1 and 2 solutions work best. They introduce the least amount of network delay and are best for time-sensitive applications.
- **Flexibility, adaptability, and scalability.** For increased business agility, organizations must embrace constantly changing business processes. Yet they need their capital equipment (which networking products are) to be useful throughout their expected life – at least until they're fully depreciated. This is a daunting task, with ever-changing network requirements. The only viable solution is to employ software-configurable networking products.

Summary

A critical mass has been created that is finally realizing the promised wave of distributed computing applications. This wave of distributed computing will be based on SOA, and will have profound impacts on the wide-area networks of businesses and government agencies. Exponential traffic growth can be expected, coupled with significant networking challenges: more delay sensitivity, less predictable traffic patterns, and increased difficulty in identifying higher-priority traffic.

For businesses and government agencies, it is becoming increasingly important for them to solve these WAN challenges. SOA-based applications are increasing their reliance upon the network. They would do best to find WAN solutions that are highly available, applications transparent, flexible, adaptable, scalable, and with low latency. Fortunately, these carrier-grade network solutions are now available to enterprises.

About the Author

Michael Mullaley is the director of enterprise marketing at Ciena. For over 20 years, he has conceived and launched networking, server, and biometric products. Michael is an active speaker in his industry. He has presented on a variety of topics at top industry events, including Networld+Interop, COMDEX, Wall Street & Technology Association conferences, Computerworld's Enterprise Management World, and HP Technology Forum.

■ ■ ■ mmullale@ciena.com

Towards Legacy Enablement Using Web Services

Leverage legacy systems with SOA

Legacy systems are a core asset at many organizations. These legacy systems have been around for decades and have a very critical impact on day to day business processes. However, owing to a variety of reasons, these legacy systems have high TCO and represent a bottleneck towards the emergence of an agile IT portfolio. In this article we'll focus on a variety of options that may be available for the seamless integration of legacy systems with an enterprise IT portfolio. We converge on legacy enablement using Web services as a viable option towards achieving an integrated, agile architecture that leverages open standards for communicating with legacy systems. We also evaluate alternative options of legacy integration and identify tools that may be used to simplify integration with legacy systems.

Overview and Pain Points of Legacy Systems

At the core of many application portfolios there are legacy information systems to support critical business processes. A legacy system can be defined as "a computer system or application program that continues to be used because of the prohibitive cost of replacing or redesigning it, despite its poor competitiveness and compatibility with modern equivalents." These are typically invaluable assets with embedded business logic representing many years of coding, developments, enhancements, and modifications. However, they are often undocumented, tightly coupled, and relatively closed and inflexible. In most cases they were developed independently without a consistent underlying architecture, resulting in overlapping and redundant functionality and data. The main pain points presented by legacy assets can be summarized as follows:

- High cost of ownership, including costs of maintenance, operation, and upgrade of both software and hardware; for example, CPU usage–based pricing for mainframes.
- High time to market because of complex and poorly understood code. This may prevent the system from satisfying the evolving business requirements because simple changes take too long to complete and test. Changes tend to cause significant ripple effects, and require more regression testing. This in turn increases maintenance and evolution costs.
- Monolithic architecture with little or no modularity together with redundant code. This is usually related to extensive patches and modifications as well as duplicated/similar functionality implemented in different systems by separate teams.
- Closed and outdated technology that is difficult to integrate and interface with new open technologies and modern distributed architectures.
- Shrinking talent pool of developers skilled in legacy systems and decreasing vendor support. Knowledge of these systems is usually restricted to a core set of people who are difficult to replace.
- Lack of application knowledge due to the departure of original developers or users as well as missing or obsolete documentation.

The important point to note is that legacy systems satisfy mission-critical operations and have been doing so for most large organizations for a long time. In addition, legacy systems have enjoyed significant investment from the IT organizations. Therefore, it needs to be noted that despite their shortcomings as illustrated above, it is not viable to replace an entire system; it requires careful planning and appropriate migration support. Rewriting all of the legacy applications and migrating data involves high cost and enormous complications. However, the burden of maintaining and extending legacy

WRITTEN BY

DR. SRIRAM ANAND

VINEET SINGH

ABHISHEK MALAY CHATTERJEE

VIVEK RAUT

VIKAS KUMAR

systems that are increasingly cut off from current technology compels the need for a planned and phased migration from legacy systems towards a more agile, IT architecture.

Options for Handling Legacy Systems

Research efforts and commercial providers have proposed various approaches to address the pain points of legacy systems. However, the major problem remains: the identification of useful legacy functionality that can be exposed for collaborating applications at an optimal level of granularity. Additionally it is important to estimate and analyze the consequences of service-enablement decisions on the system-quality attributes (such as performance and maintainability) and the solution cost. In this regard, a systematic and comprehensive method to guide the transformation and integration of legacy applications with an enterprise-level architecture such as service-oriented architecture (SOA) is currently lacking. SOA using Web services is gaining acceptance as the primary mechanism to interconnect disparate applications and ease interoperability between heterogeneous systems for internal as well as external integration. It promises improved business agility and reduced integration costs through increased interoperability and reuse of shared business services. One of the key obstacles to realizing this vision is the service enablement of existing legacy applications to collaborate in an enterprise-wide SOA. Over the past few years, significant advancements have been made to modernize and improve the interoperability of legacy systems. Figure 1 depicts our classification of existing approaches.

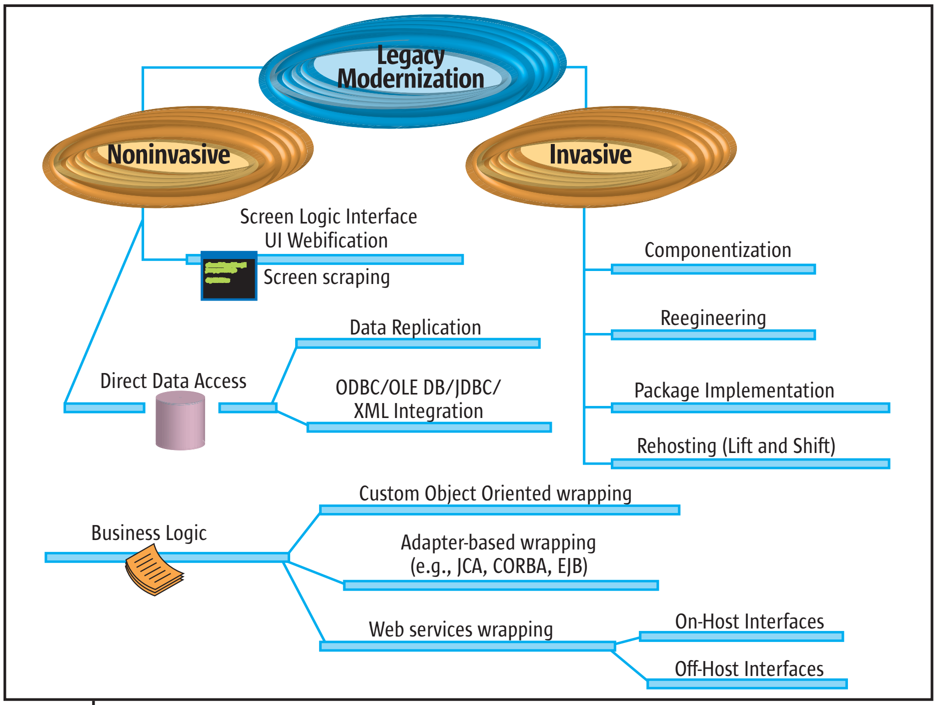


FIGURE 1 Taxonomy of legacy modernization approaches

Legacy modernization approaches can be divided into two broad categories:

- Legacy integration and service enablement is a tactical approach to align legacy systems to business needs through noninvasive wrapping of legacy assets by using new layers of flexible technologies such as EAI solutions and messaging tools, and recently with standardized interfaces using Web services.
- Legacy transformation is a strategic approach that aims to revitalize and streamline legacy systems to ease maintenance and extensions through invasive reengineering to augment the legacy systems architecture. It involves a deep and detailed analysis of the existing code base, and an understanding of the system functionality and data architecture. Subsequently, it involves the extraction and rationalization of data definitions, data, and business rules. This is followed by an iterative process that involves refactoring, consolidation, componentization, and redesigning activities to make the code more modular and to ease the incremental migration to a flexible architecture. Figure 2 illustrates the various classes of options available for legacy modernization.

Clearly, the option on the bottom left, that is, “Legacy Wrapping/Service Enablement” can prove to be a very prudent short-term solution because it promises quick turnaround and minimal changes to the existing legacy platform. The rest of this article focuses on this approach and discusses various options to perform this integration.

> “Legacy integration aims to revitalize and extend the reach and lifetime of legacy systems by exposing existing functionality through the use of wrapping”

Options for Legacy Enablement

Legacy integration aims to revitalize and extend the reach and lifetime of legacy systems by exposing existing functionality through the use of wrapping. It adds a front-end software layer to hide unwanted

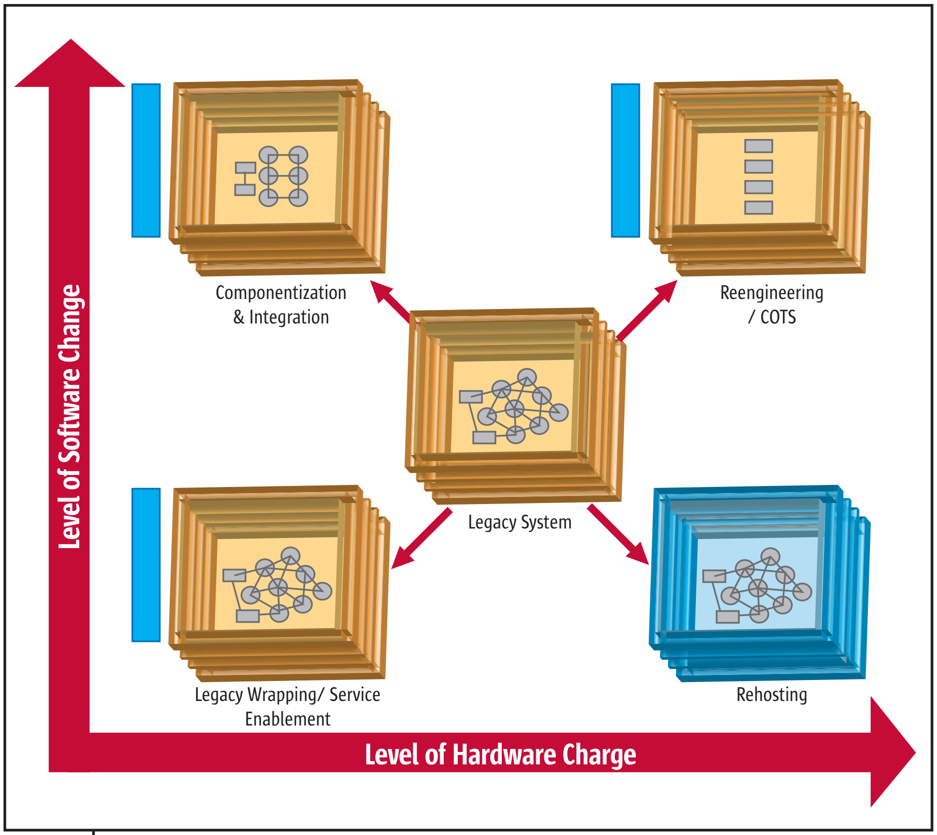


FIGURE 2 **Legacy modernization options and the level of organizational change**

complexity and exposes modern interfaces to ease interoperability. The options for integrating legacy systems are:

- User-interface wrapping or screen scraping, where legacy screens are mapped into modern graphical or Web service interfaces. Advances in the capabilities of Web-based, screen-scraping products make development of these interfaces easier with nominal development effort. However, this option makes sense only for applications that are user-interface intensive.
- Data wrapping, where new interfaces are developed for the legacy data structures to allow direct access using standard SQL or XML technologies.
- Business-logic wrapping, where legacy functionalities are wrapped and programmatically accessed through custom object-oriented wrappers, EAI adapters, or Web services interfaces.

Legacy wrapping requires less up-front architecture and design, and it can reduce integration costs and provide a roadmap to incrementally reengineer and transfer legacy components to a new platform without having to go through a "big bang" type of replacement of the system. However, legacy wrapping can only provide a tactical, short-term solution because it addresses the integration and flexibility pain points without impacting the source code significantly. True flexibility will be achieved only by understanding the legacy source code repository and refactoring it to some extent.

It is worth noting that in many cases, noninvasive modernization techniques are sufficient. However, a legacy system is not always well-suited for integration. It is not just a matter of adding an interface; in many cases, a large-scale transformation may be required. For example, the presentation logic is often intertwined and tightly coupled to the business logic, which makes it harder to create a service interface without refactoring. Another example is when the system uses an Online Entry/Batch Update pattern, in which an online inquiry and update program is used to capture data in a local database replica for subsequent batch update to the master database. Adding a straightforward service interface to the online program cannot provide a synchronous update to the master database. A transformation such as eliminating the local replica database and changing the online program to directly access the master database may be required.

Figure 3 shows a generic architectural model based on SOA that may be used to tackle legacy systems. In this architecture, legacy systems are exposed to other business applications using service wrappers that leverage a tool to perform the mapping

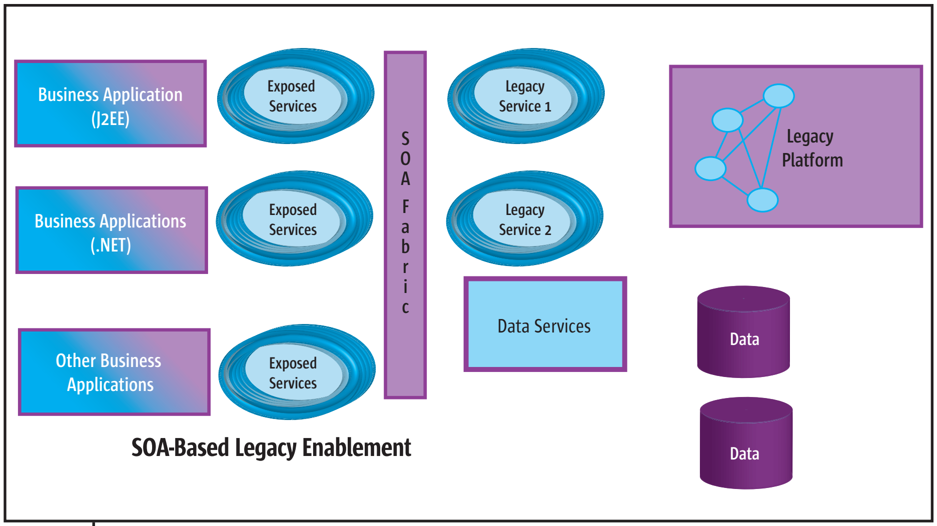


FIGURE 3 **SOA reference model for legacy enablement**

between Web services and the native mainframe platform. This type of architecture may be used in the future to migrate away from the native legacy platform without adversely impacting other applications. The advantage of this model is that legacy applications may be rationalized based on the business use cases and exposed as coarse services that encapsulate specific legacy programs. The tools will handle the complexity of communicating the legacy platform and performing the transformation. Therefore, consumer applications, whether they are in J2EE or .NET technologies, only need to deal with SOAP and XML Schema.

Web Service–Based Legacy Enablement Using NEON Systems Shadow z/Services

As organizations move toward SOA, it becomes necessary for the mainframe applications to communicate with the entire enterprise. This can be achieved using the various mainframe Web service–enablement tools that are available such as NEON's Shadow z/Services, GT Software's Ivory, WRQ's Verastream, and Jacada. These tools offer an easy way of exposing mainframe CICS, IMS, and IDMS applications as Web services without writing any program code. These tools will enable the development of a complete Web services architecture that allows legacy mainframe application to communicate seamlessly with open system platforms such as Java or .NET. These tools typically have both resident mainframe program-server components and client-side components.

NEON's Shadow technology provides a robust technical architecture to support the broad range of customer requirements for mainframe integration. Serving as both a development platform and a run-time environment for legacy integration, Shadow z/Services enables service-oriented development of applications by allowing enterprise developers to wrap existing legacy programs and transactions into ready-to-use .NET, Java, or COBOL components, as well as Web services. Shadow z/Services also provides bidirectional Web services support, not only with regard to publishing legacy transactions as Web services, but also for allowing mainframe programs to actually consume Web services. The Shadow z/Services development process is designed for ease of use – enabling developers to rapidly create enterprise components or Web services. The tool requires no prior knowledge of either the legacy system or of service-oriented development architectures. Shadow z/Services for CICS is the only run-time server in the marketplace that deploys natively on the mainframe within a CICS region. In NEON's Shadow z/Services, mainframe integration can be achieved using two major approaches: SLI (screen logic integration) or BLI (business logic integration). The SLI approach is also known as the FEPI (front-end programming interface) recording-based approach, whereas BLI is also known as the parsing-based approach.

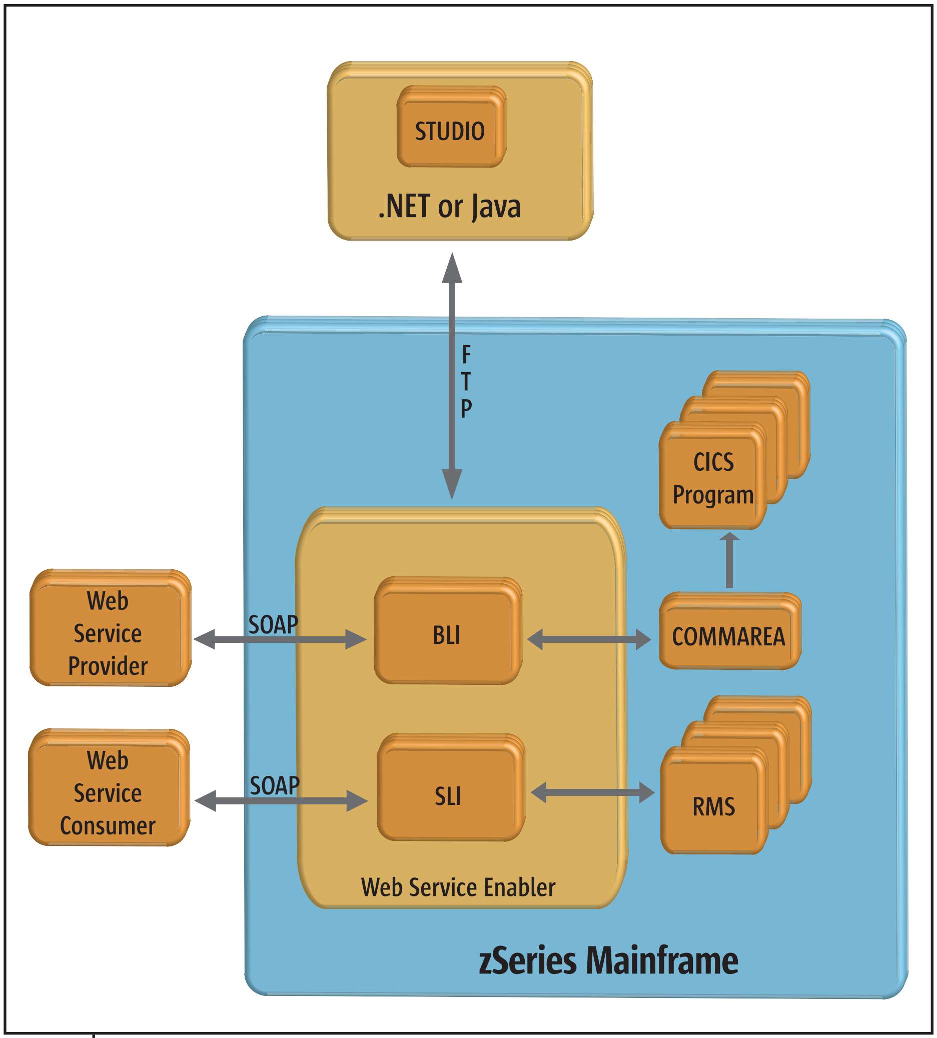


FIGURE 4 SOA-based mainframe enablement using Shadow z/Services

FEPI is used as an API to access the CICS transactions on the mainframe. Shadow z/Services simulate a normal CICS terminal, which allows running a transaction or series of screens from the IDE. The FEPI interface performs the recording and playback of these screens. The next step requires identifying the input and output fields, because this approach simulates screen scraping to transform the application into a Web service.

The BLI parsing-based approach differs from the SLI in that it allows the user to import the COBOL file and define the input and output fields. In both the cases a WSDL file is generated that controls the invocation.

GT Software's Ivory is another mainframe integration tool that contains a studio and a server. The Ivory studio provides users with an environment that contains a drag-and-drop graphical modeler. It allows the user to represent the entire business logic in the form of a flow diagram. Like Shadow z/Services, Ivory requires no prior knowledge of the legacy system or SOA. The Ivory server processes the server instructions that are XML representations of the graphical data flow deployed on the server.

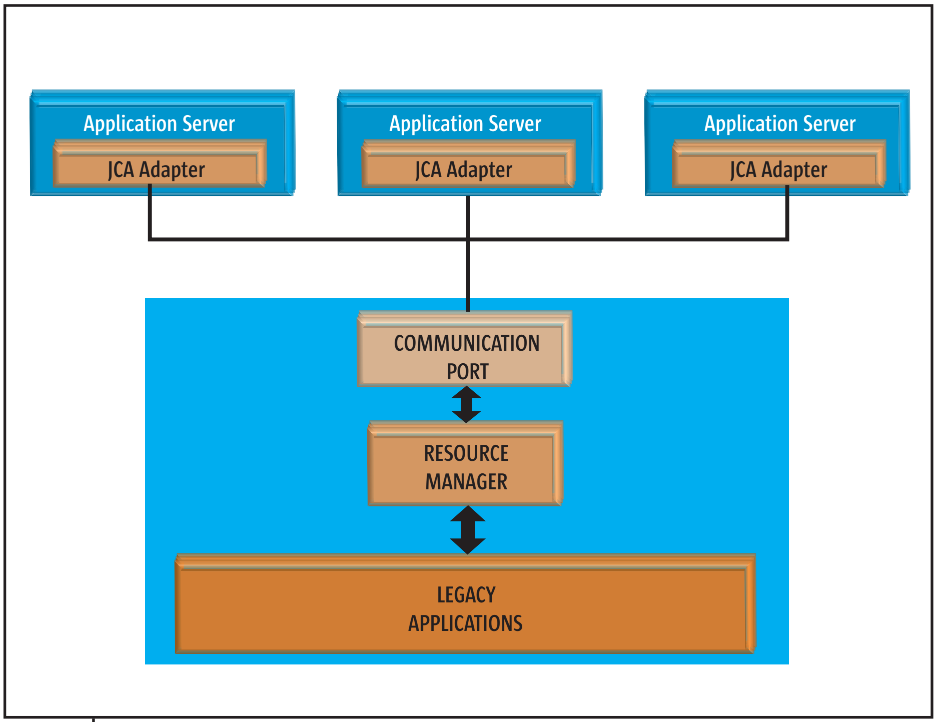


FIGURE 5 JCA-based integration with legacy applications

JCA-Based Enablement

Organizations that have an exclusive J2EE-based architecture and do not wish to use Web services for integration with legacy platforms may use JCA adaptors for integration. JCA (Java Connectivity Adapters) can be used for bidirectional communication between front-end Web-enabled applications and legacy components (such as mainframe-based applications). Since most of the business logic is already implemented on legacy components, all we need to do is use these applications from a Web-based front end to implement the business logic. There are many vendor implementations for JCA to connect to different legacy components from different front ends. For example JAM (from Iway) provides communications between WebLogic-based Java applications and mainframe applications. This helps in reusing the existing code. For the Java developer it is almost like calling any other Java class, being unaware of the background implementation. The adapters have to be configured according to application before they can be used. These adapters have two components:

- A back-end component that lies on the back-end legacy system
- A front-end component that lies on the front-end application server

Back-End Component

The back-end component is installed on the legacy system. The specific product installation guide can be used for this purpose. Each product has a different installation, depending upon the legacy server application. It has to be configured to interact with legacy applications. Depending upon the requirements, we need to configure the adapter only for those applications that can be used by the front end. Once the adapter is configured to interact with a particular application, it can be called from the front end.

This component has two distinct functions:

- To send/receive request/response to/from the front-end component
- To send/receive request/response to/from the legacy applications

The back-end component has the responsibility of receiving a request from the front end through some network protocol and creating a request that can be understood by the legacy application (this is achieved during configuration), and receiving the response. Once it receives the response, it has to convert it into a message that is understood by the front-end component and send it back to the front-end component.

Front-End Component

The front-end component lies on the application server that hosts the Web-based components. It has to be configured to connect to the back-end component that is installed on the legacy system. The configuration parameters also depend on the configurations of the adapter's component on the legacy. Once the installation is complete, it is similar to calling another class. Every adapter provides an interface that can be invoked to use the legacy application. Once such a call is made, the adapter packs the message and sends it to the back-end component on the legacy system. After the processing of the message the response is again demarshaled and returned as an object to the calling class, so it completely encapsulates the complexity involved in creating a call to legacy.

Care should be taken while selecting the adapter and its version, so that it is compatible with the existing hardware and software. Once the product is procured and installed, proper testing should be done before using it for the application integration. The following diagram illustrates the components involved in the JCA-based solution.

Pros and Cons of the Adapter-Based Approach

There are several advantages of using the adapter-based approach for legacy enablement:

- Once the installation and configuration is done, the time to market for applications that can interact with the legacy component is very low
- It is easy for the developers to use the adapters because they just need to create a simple

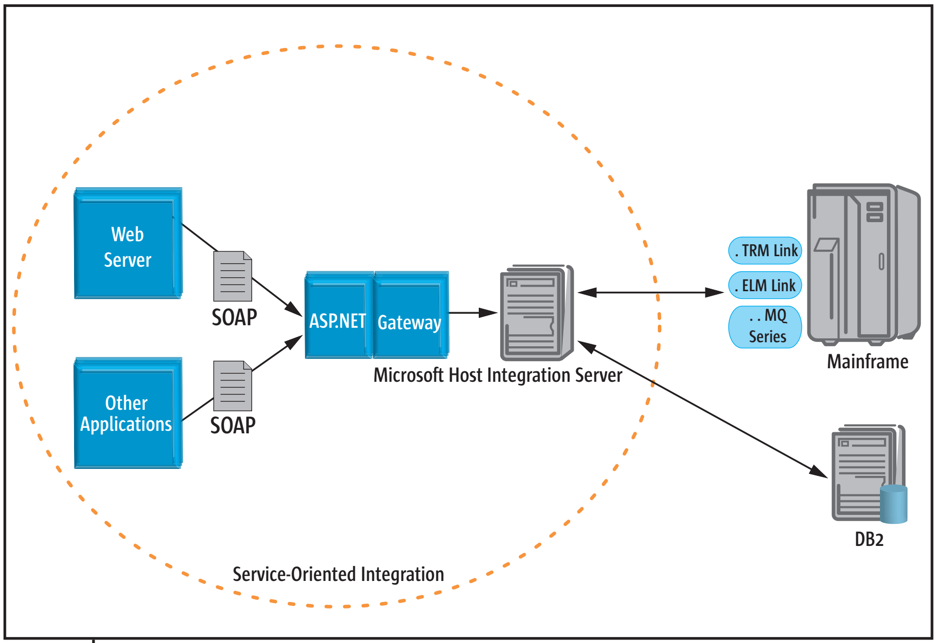


FIGURE 6 **Overall solution architecture using Microsoft Host Integration Server**

call similar to invoking a Java class

- It does not require any changes to the existing legacy applications
- It provides a bidirectional interaction with the legacy applications so that we may also use any front-end component from the legacy applications
- It is easy to use because it hides all of the complexity involved in marshalling and demarshalling requests to and from legacy applications
- This model can be easily scaled up by adding more adapters for load sharing

Although there are significant benefits to using JCA adapters for legacy enablement, these adapters have some shortcomings that may be problematic in certain situations:

- All of the adapters are tightly coupled with the applications
- Configuring adapters may be a difficult task that may require help from the system administrator for the legacy platform.
- Configuration is a time-consuming process because each function on the legacy system that has to be used by the front end needs to be separately configured
- Configuring application parameters can be difficult when the legacy application involved is complex
- There is no control over how the messages get transferred from the front end to legacy and vice versa
- It is not cost effective because the adapters are typically expensive
- Maintaining large number of such adapters on an enterprise level is difficult and time consuming

Some of the vendors in this space are:

- IWAY Software: www.iwaysoftware.com
- BEA: http://bea.com/framework.jsp?CNT=homepage_main.jsp&FP=/content
- Librados: www.librados.com

Web Service-Based Legacy Enablement Using Microsoft Host Integration Server 2004

Microsoft Host Integration Server 2004 helps to integrate IBM host applications, data sources, messaging, and security systems with new solutions developed using the Microsoft Windows Server System platform. This option may be useful for companies that are interested in using a Microsoft-centric architecture without reliance on the external tools discussed above.

The Host Integration Server 2004 features include network integration, host access with enhanced security, and application integration that enables Windows developers to publish business processes in IBM mainframe and AS/400 applications as XML Web services, which brings their Host applications and processes into a services-oriented environment.

Figure 6 illustrates the overall architecture for the solution using Microsoft Host Integration Server. The following are some key features of this product:

- Host Integration Server 2004's new Visual Studio integration piece allows us to take logic from a mainframe and expose it as a Web service from a Windows Server machine. That means we can now take legacy CICS and Information Management System (IMS) mainframe applications, publish them as Web services, and thus open it to the rest of the world or to business partners without virtually even touching the legacy application. Host Integration Server 2004 integrates with the Microsoft Visual Studio 2003, which gives it a much better look and feel.

> “ SOA-based legacy enablement can provide the loose coupling and platform independence that can free component developers from the constraints of legacy platforms ”

- Microsoft has upgraded the COM-based transaction coordinator to the new .NET-based Transaction Integrator (TI). This helps to add XML Web services capabilities to legacy applications.
- Microsoft Message Queue Services (MSMQ) MQSeries Bridge lets administrators more easily link transactional applications running on different platforms, thereby creating an interplatform message-queuing capability. The Host Integration Server feature supports MSMQ 2.0 and MQSeries 5.1.
- Host Integration Server 2004 offers a service that lets SNA-based applications running on IBM mainframes interoperate with Host Integration Server 2004 over IP-based networks. The Host Integration Server service will let enterprises replace legacy-networking hardware with standard Gigabit Ethernet networking hardware, thus saving money and improving performance.
- Host Integration Server 2004 has a functionality called Windows-Initiated Processing (WIP) that helps to call applications residing on the mainframe. Another new feature that has been added to Host Integration 2004 is Host Initiated Processing (HIP), which answers many of the WIP's limitations. Now the mainframe applications can act as clients and call other applications residing on a mainframe or a Windows Server machine. On the Windows end, Microsoft supplies listener services to make the Host Integration Server functionality work. These services use the same message formats that mainframes use and behave identically to mainframe applications. Thus, a Windows box can act as a peer to a mainframe.
- Enterprise Single Sign-On: Enterprise Single Sign-On services (SSO) have been enhanced to provide end users with a single sign-on experience when accessing non-Windows applications on mainframes and AS/400 systems. Enterprise SSO streamlines and reduces the management burden by automatically and simultaneously authenticating a user on both the Windows Server System and on the IBM host system. Enterprise SSO maps Active Directory–based domain accounts to non-Windows user credentials systems, such as RACF credentials in a credential database.
- The Host Integration Server development environment now runs from within Visual Studio .NET 2003. Programmers can use standard Visual Basic .NET or C# code, for example, to write applications that exchange data with mainframes. Host Integration Server also has a Microsoft Management Console (MMC)–based administration tool for deploying these applications, which lets you take advantage of the native capabilities of Windows Server–based machines, including load balancing and clustering.

WSJ Advertiser Index

Advertiser	URL	Phone	Page
Forum Systems	www.forumsys.com	801-313-4400	3
HostMySite	www.hostmysite.com	800-894-4678	33
IBM	www.ibm.com/middleware/soa		5
IDC	www.ibm.com/middleware/soa	800-605-5549	25
Intermedia.net	www.intermedia.com	888-379-7729	6
ISSJ	www.ISSJournal.com	888-303-5282	23
KaPow Technologies	www.kapow.com	800-805-0823	Cover IV
Mindreef	www.mindreef.com/tryout	603-465-2204	11
Parasoft	www.parasoft.com/products	888-305-0041	Cover II
Skyway Software	www.skywaysoftware.com	813-288-9355	Cover III
Smart Data Processing, Inc.	www.weekendwithexperts.com	732-598-4027	46
SYS-CON Reprints	www.sys-con.com	888-303-5282	27
SYS-CON Website	www.sys-con.com	888-303-5282	53
WebAppCabaret	www.webappcabaret.com/jdj.jsp	866-256-7973	15
Web Services Edge	www2.sys-con.com/events	201-802-3066	48
Wily Technology	www.wilytech.com	888-GET-WILY	9

Advertiser is fully responsible for all financial liability and terms of the contract executed by their agents or agencies who are acting on behalf of the advertiser. This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions.

- XML and Two-Phase Commit Support: the new Host Integration Server 2004 TI allows Windows developers to publish business processes found in mainframe CICS and IMS applications, as well as those found in IBM AS/400 systems as .NET assemblies or XML Web services, also referred to as Windows-Initiated Processing.
- .NET Framework Support: A common design environment for faster development cycles using Microsoft Visual Studio .NET. The TI Project type includes multiple views (host, COM, .NET) as well as import/export wizards for COBOL and RPG host source code. Developers derive the benefits of object-oriented, distributed applications, including rapid development, managed code, and simpler maintenance.
- Remote Diagnostics: The Host Integration Server 2004 Diagnostics tool has been improved to allow the administrator to test and troubleshoot SNA connections and resources from a central location.
- Easy DB2 Access: Host Integration Server 2004 offers a new central Data Access Tool for creating and managing connection definitions to DB2 and host file systems, and a DB2 Connect Import Wizard to define data source definitions for use with Microsoft DB2 data providers.
- Data Interoperability: Host Integration Server 2004 ships with standard ODBC, OLE DB, and .NET drivers for DB2, allowing programmers to access DB2 databases in the same manner they access other Windows-based databases. Microsoft has enhanced the DB2 network protocol client to support two-phase commit (2PC) for IP-based distributed transactions.

Summary

In this article we have examined the pain points associated with legacy systems as well as a variety of options to tackle the issues by using integration techniques. We have looked at J2EE-based techniques using JCA adapters, integration using the Microsoft Host Integration Server, as well as Web service–based integration using tools such as Shadow z/Services from NEON Systems. Based on the flexibility and openness provided by SOA, we recommend the use of Web services to integrate business applications with existing legacy applications. We would like to suggest this option as a first step even for situations where the legacy platform has a medium- or long-term decommissioning strategy in place. SOA-based legacy enablement can provide the loose coupling and platform independence that can free component developers from the constraints of legacy platforms.

References

- *Microsoft Host Integration Server:* www.microsoft.com/hiserver/evaluation/overview/default.mspx
- *JCA Adapters:* www.egeneration.com/iwaydocs/iway55/5.1/JAM/iWay_JAM_ReleaseNotes.pdf
- Bisbal, J., Lawless, D., Wu, B., and Grimson, J. 1999. "Legacy Information Systems:
  Issues and Directions. *IEEE Software*, vol. 16, no. 5, pp. 103-111.
- Cimitile, A., Lucia, A. D., Lucca, A. D., and Fasolino, A. 1997. "Identifying Objects in Legacy Systems." *5th Workshop on Program Comprehension (WPC '97)*, Los Alamitos, CA, pp. 138-146.
- Deursen, A. v., Elsinga, B., Klint, P., and
  Tolido, R. *From Legacy to Component: Software Renovation in Three Steps*. 2000.

About the Authors

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

Vineet Singh is a software engineer with Web Services Center of Excellence in SETLabs, Bangalore. His current focus is on legacy enablement to service-oriented architecture, Web services with attachments, and binary XML. He has been working on prevention and detection of XML-based denial of service attack on Web services. He has substantial experience in publishing papers and presenting papers at conferences.

■ ■ ■ vineet_singh@infosys.com

Abhishek Malay Chatterjee is working as part of the Web Services COE (Center of Excellence) for Infosys Technologies Ltd., a global IT consulting firm, and has substantial experience in publishing papers, presenting papers at conferences, and defining standards for SOA and Web services.

■ ■ ■ abhishek_chatterjee@infosys.com

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

■ ■ ■ vivek_raut@infosys.com

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

■ ■ ■ vikas_kumar02@infosys.com

Bringing SOA to Life: The Art and Science of Service Discovery and Design

Practical guidelines and experiences from real-world SOA projects

In a service-oriented architecture (SOA), a service is a unit of work performed by a service provider to achieve desired results for one or more service consumers. A service provides a function that is well defined, self-contained (for example, loosely coupled to its environment), described solely by its interface contract and behavioral attributes (for example, it hides implementation), and located anywhere on the network.

WRITTEN BY
MANAS DEB,
JOHANNES HELBIG,
MANFRED KROLL,
& ALEXANDER SCHERDIN

Essential components of an SOA – such as a service provider, service repository, service mediator, and service consumers – all rely on service definitions as the key element to describe, access, transport, and understand services.

Modern SOA as a pattern for transforming enterprise architecture is still emerging, as are the strategies for service discovery and design. At this juncture of SOA's maturation, it's highly useful to consider concepts and practical experience gleaned from serious and substantial SOA implementations. In this article we provide both. For various aspects of service design activities, we discuss general considerations that are often motivated by well-accepted software engineering ideas. We also describe the experience of Deutsche Post, Germany – one of the world's largest global logistics service providers – where SOA concepts have been widely adopted at the enterprise level.

SOA at Deutsche Post is business focused. Several SOA-specific processes and methods have been established to put the promised benefits into practice. Further, a highly sophisticated, enterprise-quality service backplane called SOPware (service-oriented platform) that leverages Oracle's Fusion Middleware Suite provides the supporting infrastructure for business-service mediation. Thus, business units don't have to worry about the technical details of service implementation and provisioning; instead, they can quickly and easily assemble high-level business services and processes.

This article focuses on the crucial aspects of bringing SOA to life: how to establish business ownership, how to discover and design services, and how to foster an SOA-based enterprise architecture transformation.

The Starting Point: Service Discovery and Portfolio Management

Business Services vs Technical Services

The principal attraction of SOA is its ability to drive IT evolution through business *and* to enable better business operations through a more flexible IT. While the basic concepts of SOA can be helpful to build more efficient IT operations, the larger benefits of SOA are obtained when business operations can quickly build competitive business applications and make such applications resilient to change. Consequently, services used by high-level business processes, which we call business services, are the main focus of this article. Technical services, in contrast, are typically embedded in the infrastructure that supports these business services. As important as these technical services are, a good SOA implementation should clearly separate business from technical services.

To realize the full transformation potential of an SOA, business people must take responsibility for identifying business services and describing their characteristics. To that end, several enterprises, including Deutsche Post, are using the concept of business domains to establish business ownership.

Establishing Business Ownership for SOA

To create a common language for business and IT people to discuss potential services and to foster ownership for specific business services, it's wise to look at the fundamental structure and relationships defining the business (for example, a customer buys a product, or an invoice is sent to a customer). Here, the scope of an organization's business is first factored into business domains, which are essentially blocks containing closely related functionality and data, such as customers, products, and contracts. In order to design a domain landscape that renders the benefits of long-term stability and thus provides solid ground for introducing SOA at all levels, it's important to understand how a specific enterprise conducts its business (see Figure 1).

Practical experience at Deutsche Post shows that identifying core business objects and their relationships provides a suitable first iteration of constructing the domain architecture. Deeper analysis of the business processes and their interactions will lead to more precise domain descriptions, as well as the definition of subdomains for some major domains. We strongly recommend checking the robustness of the domain architecture by mapping it against actual *and* potential future business scenarios.

Designing business domains must be based on deep business understanding, but it also requires some gut feeling. Let's consider two domains: customer and customer relationship. The first mainly manages key customer data required by most other domains, while the second tracks various aspects of the ongoing customer relationship. These domains may seem strongly related at first, particularly because both deal with customers and the domain names sound similar – so why not combine them into one? A close inspection of the fundamental business relationship reveals crucial differences. The customer domain provides information that changes relatively infrequently, but it needs to provide a 360° view of all aspects of a customer and is used in a similar fashion in most domains. The customer relationship domain, on the other hand, provides functionality that has a broad variety and changes frequently – as often as the customer has some dealing with the company and as the CRM strategy of the company evolves. Business people responsible for customer relationships often use the information quite differently – for instance, some use it for real-time cross-selling while others use it to render better customer service. Thus, choosing separate domains for customer and customer relationship is well justified. (For other examples of domain architecture in mobile telecom and in banking, see the fourth and fifth entries in the References section.)

This example points to another important consideration: to use SOA to transform an enterprise IT landscape, business people must take ownership of business architecture. Business owners are responsible for the scope of the domains and for providing ideas for those business services that their domains provide, as well as for identifying requirements for services

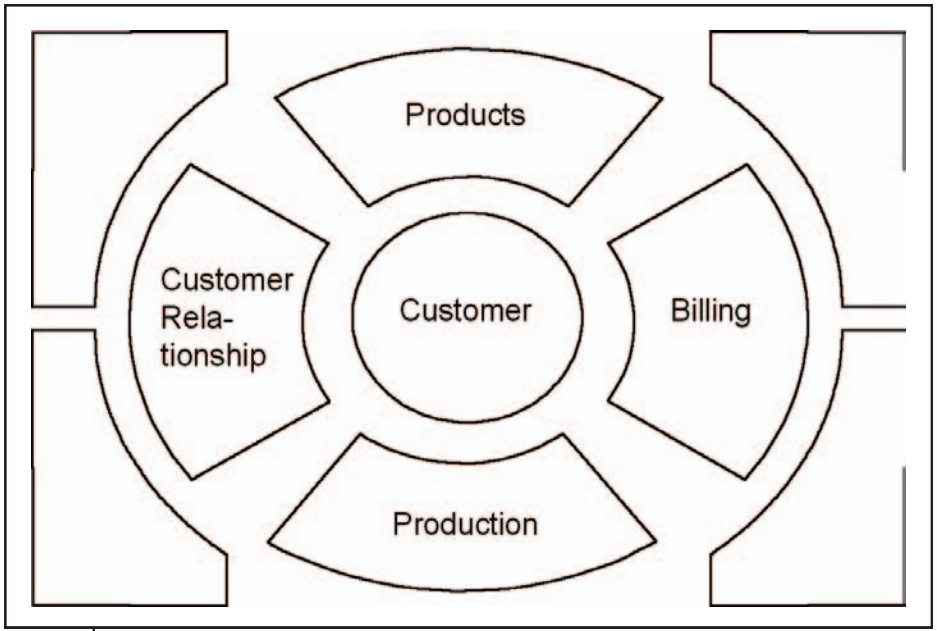


FIGURE 1 **Simplified high-level domains of a logical service architecture**

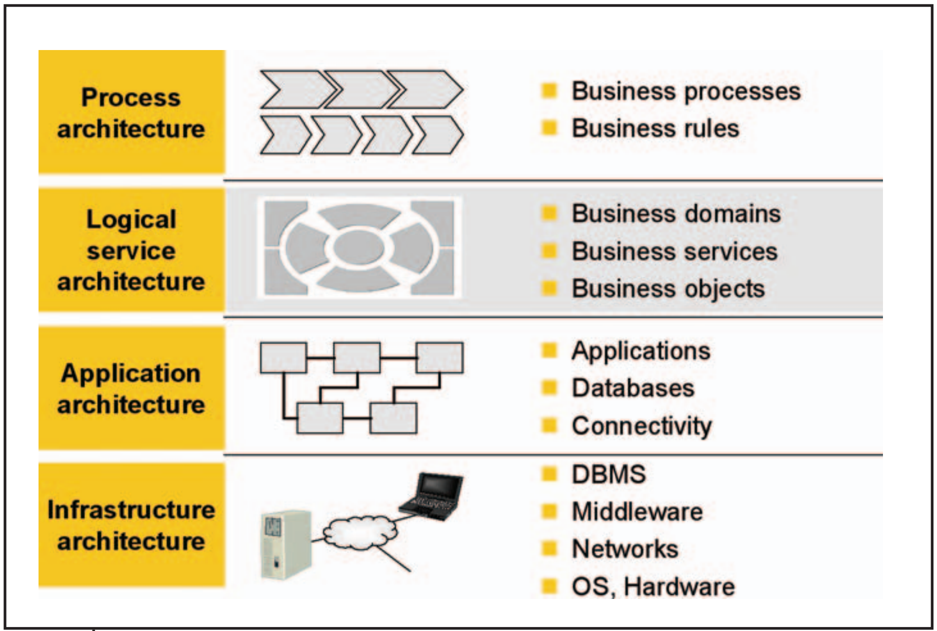


FIGURE 2 **Logical service architecture as an abstraction layer between process and application architecture**

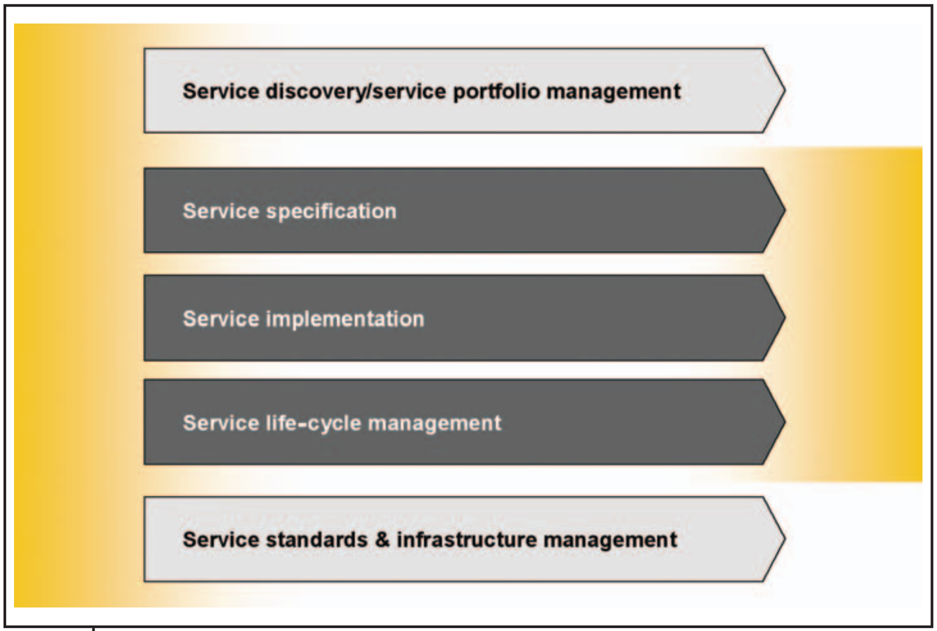

Service discovery/service portfolio management
Service specification
Service implementation
Service life-cycle management
Service standards & infrastructure management

FIGURE 3 **Service design processes**

that their domain needs (which are, in turn, provided by other domains). This establishes a federal business-IT governance, including clear definitions of data ownership. At Deutsche Post, enterprise architects and service designers from the business-IT organizations help the business owners work out the details; however, the final responsibility lies with the business side (see Figure 2).

The business domains, as well as the business services, establish an abstraction layer between the process architecture and the application architecture of an enterprise.

The benefit is clear: by decoupling processes and applications through domains and their associated services, change cycles can be managed independently. IT systems (both on the application and the infrastructure layer) may be changed or replaced without affecting business processes, as long as the service contracts are kept stable. Conversely, business processes can be changed or augmented relatively easily when many of the building blocks (the business services) already exist, waiting to be reused.

Service Discovery

While in some ways services are similar to components, they are also similar to mini-applications that must be managed in a similar fashion to applications. As shown in Figure 3, Deutsche Post defined a set of five service design processes that provide a grid for creating and evolving the service portfolio – these processes have general applicability in any industry.

The first step of creating and managing a business service portfolio is service discovery. In order to look for a candidate set of services that a domain should provide, you can investigate the ensemble of business objects and their relationships. Business objects (such as customers, invoices, and addresses) are *not* IT objects, but entities required to do business described solely in business terms without reference to the specifics of any IT system. For example, one of the key services of the domain customer is likely to be CustomerInformation, and the elementary service operations associated with this business service are likely to be CRUD (create, read, update, delete) type. However, more complex functionality that combines several basic service operations,may also be associated with this domain. For example, a customer domain may host a CustomerAddressConsistency service, encapsulating the check for existence and proper name of the customer, as well as verifying (and potentially rectifying) her address. Other interesting service candidates may be found by looking at specific combinations of key business objects (such as looking at contracts and their associated invoices).

This top-down approach to service discovery can be complemented by two other approaches. Tracing business processes, following their key interactions with their business environment, and grouping similar functionality can lead to service candidates potentially overlooked by a pure top-down approach. This business process–driven approach can provide a sanity check for completeness and give a first indication of the reuse potential of specific services. However, it requires some degree of caution: different business processes (potentially defined by different business people) often feature different semantics for the same (or similar) concepts. Our experience shows that a thorough and rigid analysis of business semantics (including translating or even harmonizing business terms) is a key prerequisite to achieving a working service portfolio. Note that this approach alone may easily lead to a too fine-grained definition of services.

Additionally, a usage-driven approach to service discovery looks for which IT infrastructure (such as mainframe or legacy) application functionalities and data have been used by business applications to date, wraps service interfaces on them, and uses these services

> “ As important as these technical services are, a good SOA implementation should clearly separate business from technical services ”

for future business applications. While this so-called bottom-up approach may complement the two other approaches for service discovery, we believe that the lack of semantic harmonization of services will not lead to a sustainable SOA transformation. We'd recommend this approach only as a starting point – for example, to prove the general working of an SOA in a political environment.

Further, we suggest starting service discovery with a small investment in creating conceptual services (for example, only specification but no realization) that combines the aforementioned approaches, and start with actual service implementation in a step-wise fashion, strictly based on business priorities and business cases for individual projects. Business analysts and enterprise business architects obviously play a major role in the business service discovery phase.

Making Change Happen: Enterprise Architecture Transformation Through SOA

Building the Service Portfolio

At Deutsche Post, service discovery yielded a portfolio of nearly 100 business service candidates (each, on average, featuring 5 to 10 service operations) that were subsequently prioritized by factors such as business value, reuse potential, and IT complexity reduction potential. These candidates provide the baseline for an ongoing service portfolio management process that refines the roster of candidates and triggers detailed specification, implementation, reuse, and life-cycle management of specific services.

There are many important issues to deal with while building up a service portfolio, such as service ownership, funding for creating and maintaining services, and life-cycle management. Based on the balance between focusing on SOA services and focusing on project execution, different service acquisition styles can be recognized. For example, one style might undertake an enterprise-wide effort to create or acquire most of the potentially useful services and then start a series of SOA projects that would implement (and use) them. This more services-focused approach aims at first creating a rich service portfolio but involves solid, long-term strategy and some degree of up-front investment (most notably for service design methodology, as well as for a service mediation platform). Another, more projects-focused approach would give priority to building services as required by some projects without much emphasis on how they might be shared. This approach would provide short-term returns but could hamper long-term SOA benefits.

From the beginning, Deutsche Post has emphasized an evolutionary approach for architecture transformation in accordance with SOA principles. This managed evolution approach aims to define smaller IT projects with positive business cases and limited scope, so as to provide short-term business benefits (tactical aspect) while complying with a target application landscape (the SOA blueprint along the business domains) defined at the enterprise level. These projects must have a business owner and will be conducted anyway, with or without an SOA. However, by identifying the services that could be either implemented or reused by a project, the managed evolution approach creates a desirable balance between the services-focused and the projects-focused approach. Each project will thus create business value *and* contribute to the evolution of a flexible IT landscape based on a broad service portfolio (see Figure 4).

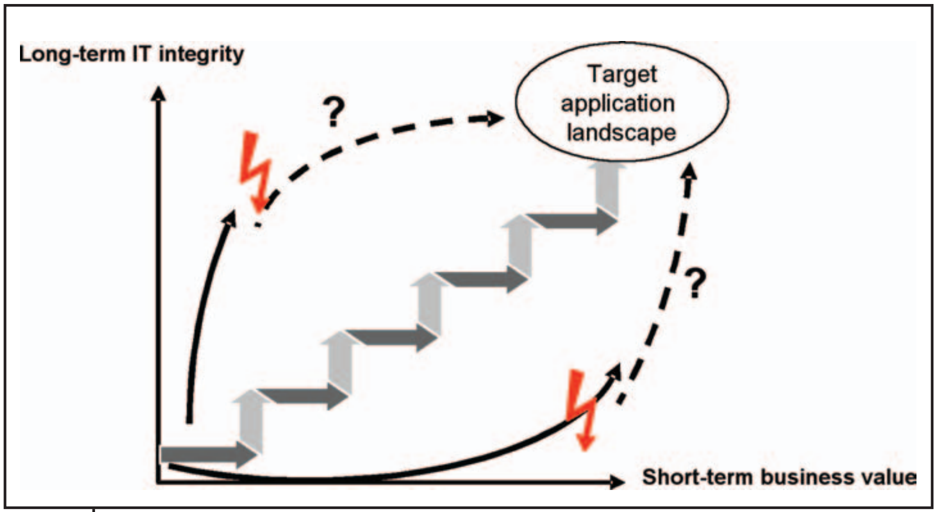


FIGURE 4 SOA adoption through managed evolution

Specifying Services

Additional refinements pertaining to various characteristics of this service are required for each candidate service to be implemented by a project. Service specification at Deutsche Post has three steps. The first step is taken during the service discovery phase, rendering an *enterprise service description* in the context of Deutsche Post's enterprise architecture model. This enterprise business object and service model describes the logical service architecture view and contains domains, business objects, and services. The services are described as interfaces in this model (UML is used as a formal language), while the description of service operations refers to business objects and other data types in the model.

The second step, the detailed *business service specification*, includes refining the service interface and taking specific aspects of the IT project (such as potential limitations of the service scope) and then realizing or acquiring the service under consideration. Business service specification is critical in the service design processes, as these specifications are the essential communication method for all parties interested in the service. You should consider the following key categories of information while creating service specifications:

- ***Definition:*** Service and service operation names, input and output parameters for each operation, admissible invocation styles (such as synchronous or asynchronous), and service classification (such as business, technical, or infrastructure; or process-oriented, business logic, or data access).
- ***QoS/SLA:*** This category has information on the service's performance quality, including

useful details on load versus response times, availability and fail-over responses, and possible SLAs.

- ***Constraints and Policies:*** This includes pre and post conditions relevant to the service invocation, admissible/expected ranges of input/output parameters, including error returns, and particular sequences in which operations should be invoked. Security guidelines and other essential policies can also be included here.
- ***Ownership:*** Information on those who own or maintain the service.
- ***Release Information:*** This groups status or planning information about past, current, and planned versions of the service, plus information on any interversion compatibility-related issues.
- ***Operational Behavior and Usage Patterns:*** This category, which is particularly helpful for complex or nonobvious services, should provide information about different and recommended ways to use the service.

The service specification exercise is completed in the last step, the *technical service specification*, where service mediation platform-specific requirements are spelled out, and other missing technical requirements (such as specific throughput or message sizes) are defined in detail. Proper classification of a service also helps you apply design and implementation best practices to that service.

The functionality, quality, and policies that govern the service render an agreement between the provider and the consumer, the so-called service contract. Currently there are no industry-wide accepted standards for service specification. However, WSDL and UDDI are becoming the de facto standards for describing and publishing the basic service definition. Tests using Web Services Interoperability (WS-I) specifications are also becoming popular with services aimed at wider reuse.

Specification iterations produce finer and more concrete details about the service interfaces and functionalities, thus making the service more flexible and cost-effective by incorporating present requirements as well as projected future needs. Service specification iterations typically require close collaboration between business service owners, business analysts, and technical analysts.

Optimal Service Granularity: Fine or Coarse?

A popular debate in the SOA community concerns how fine or coarse the services should be. While there's no standard way to quantify service granularity, we can use some ideas from component-based design, such as the number of function points or data elements affected by the invocation of a service. If a service needs to be called too many times in a business application or if only a small part of its functionality is typically used, it's likely that the service is too coarse. If too many parameters for a service are required, the service is most likely too low level and fine grained. While these observations are based on real-life service implementations, in our experience this question is resolved by effective service portfolio and life-cycle management processes: the degree of appropriateness of a service portfolio grows steeply over time by starting with high-value service operations and expanding and evolving them over time. Striving for an appropriate granularity will maximize ease of use, reuse, and manageability; an appropriate service is not necessarily either fine or coarse, but one that maximizes business value.

At Deutsche Post, mostly coarse-grained services were implemented at first. These were a rather stable set of fundamental services that could be initially established. Over time, the number of service consumers grew and more specific requirements emerged; thus, more fine-grained service operations are currently implemented. These elementary (or atomic) service operations provide a baseline for defining a rich portfolio of more complex service operations – for example, by combination (compound service operations) or by process orchestration (orchestrated service operations).

Service Implementation and Life-Cycle Management

While in most ways service implementation is not much different from software development, we'd like to point out two important differences.

First, service interface code for a chosen service mediation platform doesn't need to be hand coded, but it can – with a capable service-design tool chain – be generated. Deutsche Post uses a service-design tool chain based on the model-driven architecture paradigm. The starting point is the enterprise business object and service model, which is transformed to a project business object and service model (PBSM). After the PBSM is further refined as part of a project specification, the tool chain generates both code skeleto ns for Deutsche Post's SOPware service mediation platform and service specification documents. While the service specification artifacts are modeled with appropriate tools, XML and XMI are chosen as proven and standardized exchange languages.

Second, when setting up or acquiring a service mediation platform, one must consider support for service testing, since services by definition require a loosely coupled environment. For example, at Deutsche Post, service module tests use the DevBox, a component of the SOPware that provides service invocation libraries and simulates service providers on a developer's local machine. Additionally, Deutsche Post has set up a service architecture test lab that provides an environment that comprises all services and service providers in order to integration-test SOA applications.

While service interfaces will be stable elements of an enterprise architecture, services will grow (in terms of additional service operations added over time) and service implementations will change as service providers behind the façade change. While an SOA minimizes the impact of this change, the life cycle of services must still be managed. The elements of service life-cycle management are version control, a business service repository (including service models and specifications), and a technical service repository (service registry) that enables physical access to services at implementation and run time.

Technical Considerations: Service Platform and Standards

As we've seen above, business services can effectively decouple business change and IT change. However, what if the IT infrastructure (or elements of it) that mediates the services needs to change? We've seen enterprises entangled in an EAI or middleware legacy, unable to change at the speed required by the business. When some of those technologies became obsolete or couldn't cope with growing performance requirements, tool-specific EAI adapters and interfaces based on proprietary middleware protocols couldn't be migrated with reasonable effort. The promise of flexibility through technology proved to be futile.

Thus, we recommend a standards-based approach for service mediation, such as the new Java Standard Java Business Integration (JBI) or an approach similar to Deutsche Post's SOPware (using, among other standards, JBI). In SOPware, another abstraction layer has been set up, this time with the purpose of decoupling business service logic from the underlying IT infrastructure. This layer is purely based on standards such as J2EE (.NET would be an option, too), XML, and WS-I. Driven by technological progress and functional expansion, SOPware, created in 2001, effectively replaced or augmented nearly all major infrastructure components (such as application server, MOM, directory server, and transformation engine) without affecting the developer interface. Simple XML-based APIs to create and invoke services are the only aspect of technology that developers need to know about service mediation, while the specifics of underlying technologies and tools are hidden. Programmers can thus focus on implementing business logic, rather than on low value-add infrastructure code. This strategy, for Deutsche Post, provided the ultimate flexibility and investment protection – not just toward business logic, but also toward IT infrastructure.

Conclusion

While many scientific principles can guide service discovery and design, as with any architectural work, some gut-level experience from industry veterans can go a long way in the process – particularly in the early iteration cycles. Making SOA a reality calls for quick and easy implementations: business people should focus mainly on the business logic, not on protocols and transport. Thus, a full-featured SOA backplane can be really useful. Also, using open standards and a modular suite of technology components should promote higher reusability of SOA assets at lower TCO, minimize vendor lock-in, and reduce SOA adoption risks.

Put into a short equation, SOA = semantic integration + loose coupling + managed evolution. However, our most important message is to use SOA as a common language between business and IT people, thus bridging the gap that has, for a long time and at many enterprises, blocked the path to flexibility of both business processes and the IT landscapes that support them.

References

- SOP management paper (Deutsche Post World Net – SOP Group), 2004.
- SOP SBB technical paper (Deutsche Post World Net – SOP Group), 2004.
- SOA Is More Than Hype (Johannes Helbig), Presentation at Gartner Application & Web Services Summit, June 2005, Barcelona.
- Reusing IT Components in Mobile-Telecom Companies (Enrico Benni, Klemens Hjartar, Jürgen Laartz, and Alexander Scherdin), McKinsey on IT, Fall 2003.
- Services, Events, and Contracts – SOA at Credit Suisse (Claus Hagen), Presentation at SOA Days, February 2005, Bonn.
- Designing IT for Business (Jürgen Laartz, Eric Monnoyer, and Alexander Scherdin), McKinsey Quarterly, 2003, Number 3.
- SOA Is More Than Hype – Use It for IT Business Integration (Mark Happner, Michael Herr, and Alexander Scherdin), Presentation at SOA Days, February 2005, Bonn.
- SOA – Complexity Management Through Integration (Johannes Helbig), Presentation at OOP Conference, January 2004, Munich.

About the Authors

Dr. Manas Deb, a senior director at Oracle's Fusion Middleware Group, currently leads strategic engagements operations for Oracle's service-oriented integration solutions. He has worked in the software industry for nearly 20 years, half of which he spent architecting and leading a wide variety of enterprise-level application and business integrations projects. Manas has a PhD in Computer Science and Applied Mathematics, as well as an MBA.

■ ■ ■ manas.deb@oracle.com

Dr. Johannes Helbig is considered among the first to have formulated the concept of a service-oriented architecture, and is father of the SOA program at Deutsche Post, where he currently serves as member of the board and CIO of the MAIL division. He holds a doctoral degree in theoretical computer science and his main interests include IT architecture and IT management.

Manfred Kroll is director of Business Architecture and Processes at Deutsche Post's MAIL division. As chief enterprise architect, his objective is to implement a cooperative, SOA-based application landscape pursuing an evolutionary approach. Before joining Deutsche Post MAIL, he held several management positions, most notably at IBM Development Labs and at T-Mobile. Kroll has a Masters degree in Computer Science from University Dortmund, Germany.

■ ■ ■ manfred.kroll@deutschepost.de

Dr. Alexander Scherdin, senior professional for IT Service Design and SOA at Deutsche Post's MAIL division, is responsible for broadening the company's service portfolio and driving forward the definition and execution of its service design processes. Before joining Deutsche Post MAIL, he worked as an IT architecture consultant at McKinsey & Company's Business Technology office. Scherdin studied at the University of Frankfurt, Germany, and holds a PhD in Theoretical Physics.

■ ■ ■ alexander.scherdin@deutschepost.de

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 Edge 06

June 2006 | New York, NY

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

Suggested topics...

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

CALL FOR PAPERS NOW OPEN!

2006 ENTERPRISE> OPENSOURCE CONFERENCE+EXPO

June 2006 | New York, NY

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

Suggested topics...

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

Submit Your Topic Today! www2.sys-con.com/events

Sponsored by

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

Attention Exhibitors:

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

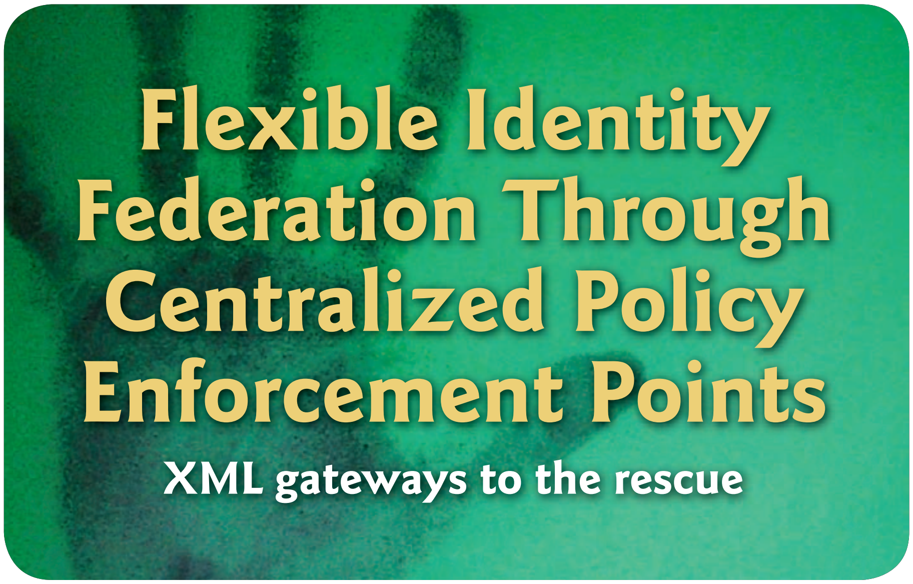
*Call for Papers email: jimh@sys-con.com

For Exhibit and Sponsorship Information ► Call 201 802-3066

Produced by SYS-CON EVENTS

XML JOURNAL SUPPLEMENT

This Month



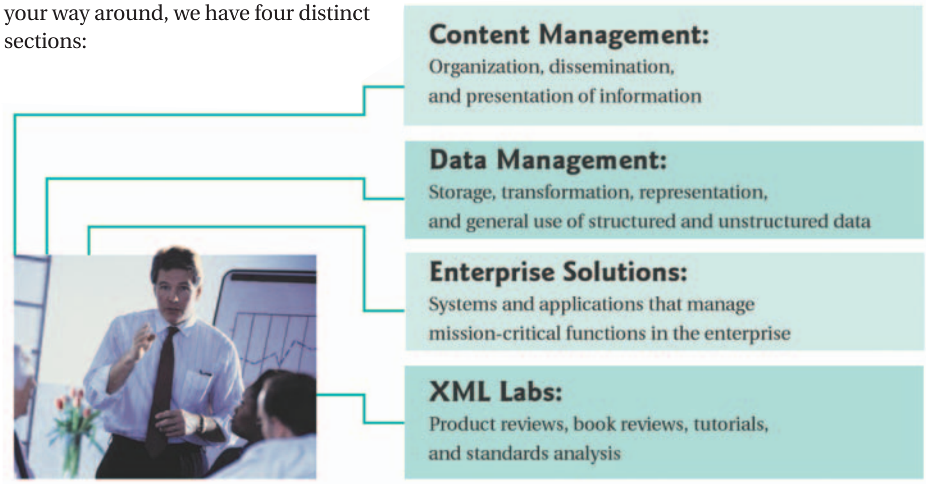
Flexible Identity Federation Through Centralized Policy Enforcement Points

XML gateways to the rescue

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

Flexible Identity Federation Through Centralized Policy Enforcement Points

XML gateways to the rescue

WRITTEN BY
Francois Lascelles

Imagine a fresh business relationship between ACME Corporation and Partner. As a result of this relationship, ACME wants to grant Partner limited access to one of its core internal applications. They do this, naturally, by exposing a Web service.

Why Identity Federation?

Boris (an employee at Partner) sends a SOAP request to the ACME Web service along with some password or proof-of-possession type credentials. Because Boris's identity is managed outside of ACME, those credentials cannot be authenticated using ACME's authentication infrastructure.

To circumvent this issue, one could imagine a setup where the ACME Web service authenticates Boris's credentials by connecting to Partner's authentication services. Another alternative might involve some sort of directory replication. These strategies were attempted in the '90s when distributed LDAP references appeared in the protocol to try creating metadirectories.

Although most commercial LDAP directories have replication functionalities, these are not typically used to replicate authentication data across enterprises. For Partner to expose his authentication system to the outside world in any way is not an option: doing so would introduce major security and confidentiality issues.

Ideally, Boris's credentials must be authenticated in the confines of Partner's identity domain *before* the SOAP request is sent to the Web service. At the receiving end, the ACME Web service will not authenticate Boris's credentials directly; instead, it requires a satisfactory *proof of authentication* before letting Boris's SOAP request through.

This mechanism where one entity delegates authentication and/or authorization to another entity is known as "identity federation."

SAML Holder-of-Key

SAML describes different scenarios that allow for identity federation. Let's first look at the holder-of-key approach.

By now you may have heard of WS-Trust. WS-Trust defines the syntax that Boris uses to request a SAML Security Token. Simply put, Boris sends a *RequestSecurityToken* SOAP request to Partner's internal *Security Token Service*. The token service authenticates Boris's credentials according to its own policy and returns a *RequestSecurityTokenResponse* that includes the SAML Signed Security Token. So far, all of this is happening inside Partner's domain.

A SAML assertion can make different types of statements about a subject: Authentication Statements, Attribute State-

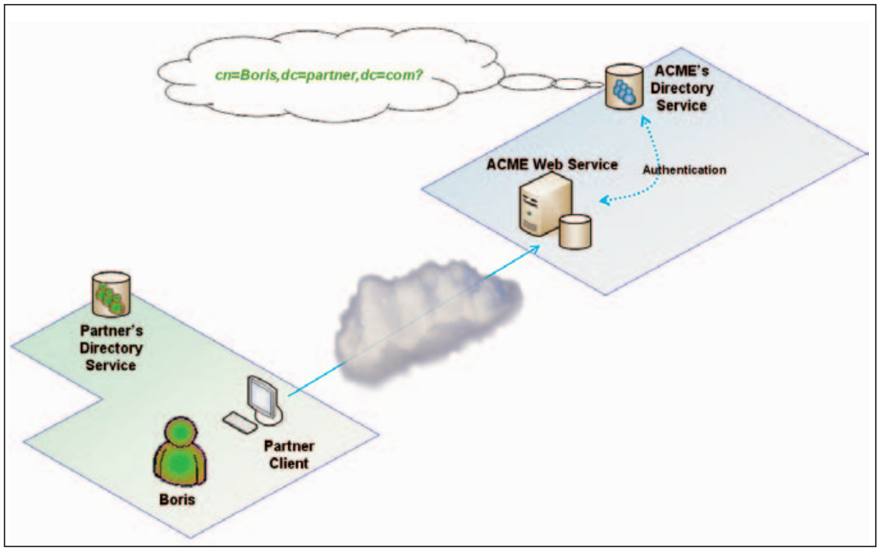


Figure 1 • Disconnected identity domains

ments, and Authorization Decision Statements. In this case, Partner's Security Token Service will make an Authentication Statement regarding the subject of the SAML assertion: Boris.

The issuing authority digitally signs the SAML assertion, and this constitutes the basis on which trust is established. Boris binds this SAML assertion to his outgoing SOAP requests. He can reuse the same assertion for future SOAP requests as long as it remains valid (the validity of a SAML assertion is typically very short). The process of binding the SAML assertion to an outgoing SOAP message involves Boris's including the SAML assertion in the SOAP message's header and signing the message with his private key. The SAML assertion includes a *SubjectConfirmation* element that contains a client certificate for Boris's private key. In order for the receiving end to confirm that the message is sent by the owner of the SAML assertion, it will verify the digital signature of the message using the certificate that is part of the SAML assertion's *SubjectConfirmation* element.

This process prevents an attacker from sniffing the SAML assertion and using it to impersonate Boris. Similarly, an attacker would not be able to substitute his own certificate inside Boris's SAML assertion, because this would break the issuer's digital signature, and the assertion would become invalid.

Back at ACME, the Web service is configured to trust Partner's Token Service as an issuing authority. Practically, the digital signature that is part of the SAML assertion can be verified using the digital certificate of the issuing authority. The certificate of the issuing authority is what the Web service needs to be configured with to allow trust of the authentication claims and thus, identity federation with Partner.

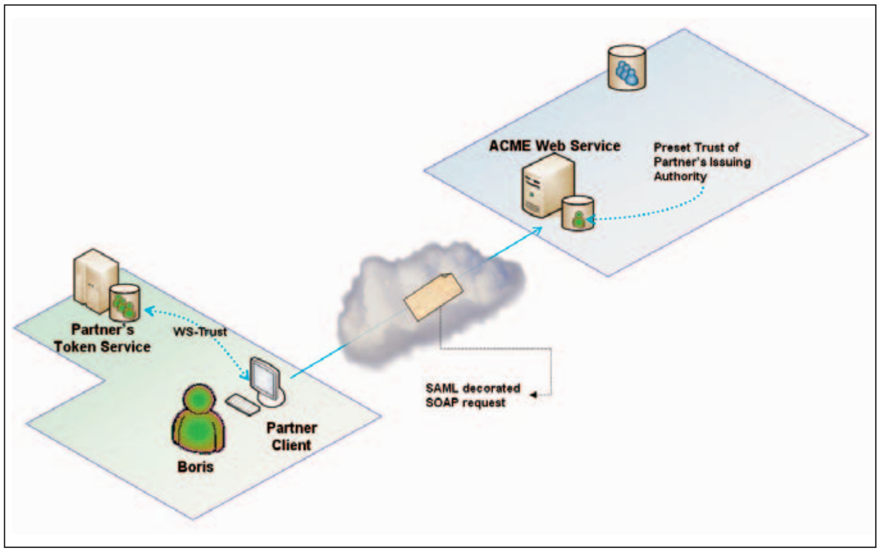

ACME Web Service
Preset Trust of Partner's Issuing Authority
Partner's Token Service
WS-Trust
SAML decorated SOAP request
Partner Client
Boris

Figure 2 • Identity federation using SAML holder-of-key

At run time, the Web service verifies the signature of the SAML assertion, checks that it trusts this particular issuing authority, then checks that the message is received within the validity period of the SAML assertion, then verifies the signature of the message itself (Boris's signature), and finally, checks that the signature uses the same cert as the one specified by the holder-of-key SAML assertion.

SAML Sender-Vouches

The sender-vouches approach's main difference over holder-of-key is that the issuing authority also acts as the sender of the message to the Web service. Boris sends his message to an issuing authority, which also acts as a proxy by forwarding the message to the ACME Web service. In this case, the SOAP message is signed by the issuing authority directly and the receiving end only needs to validate one signer.

Flexible, Centralized Trust

In practical terms, the trust of an issuing authority will require the Web service to be configured with the digital certificate used to sign those SAML assertions. The Web service may also want to keep a list of remote subjects as opposed to blindly letting through any identity authenticated by the issuing authority. You may want to allow Boris through, but nobody else from Partner for now. Alternatively, you may want to federate authorization as you did with authentication so far (refer to *SAML Authorization Decision Statements* for more details). In any case, these implementation details of the Web service reflect the particularities of a real business relationship.

Back at ACME, a new business relationship may require establishing trust with a different issuing authority, which in turns necessitates a change to the Web service. Suppose there is a new partner (let's call him "Partner B") who uses a proxy-like approach and does not support SAML holder-of-key (only sender-vouches), and our Web service was expecting holder-of-key up until now. Boris may have left Partner for greener pastures, and his responsibilities have shifted to Maurice.

Such events necessitate maintenance on the Web service. Perhaps this means an unfortunate interruption of service. If ACME now exposes multiple Web services, each implementing its own trust and authorization management, any change to business relationships could require maintenance in each of these Web services. In addition to the interruption of service and the potential risk associated with any Web service changes, there is the issue of cost for each cycle of development, quality assurance, and deployment.

Still, just as business relationships are expected to change over time, so too will the implementation details of the Web service that pertains to trust and authorization – or will they? If trust and authorization are not closely linked to the Web service's application logic (as would typically be expected), then they should be abstracted out and handled by a centralized policy enforcement point.

By letting an XML gateway enforce the authentication and authorization rules, changes to these policies become an administrative task that does not require changes to the Web service, nor service interruption. Adding or removing trust of an issuing authority is but a mouse click away. XML gateways that support SAML-based identity federation transform software maintenance nightmares into security manager dreams.

Again, supposing that ACME is deploying more and more Web services, each of these Web services may have its own trust requirement. Some may be for internal use only, some may allow for Partner to consume, others for Partner B, and other Web services may authorize a combination of partners depending on the operation being invoked.

Using an XML gateway to manage and enforce the different Web services security requirements not only allows for flexibility of trust over time, but also provides a centralized policy enforcement point that enables a global view of security across the enterprise. Perhaps one of the

HOME
ENTERPRISE SOLUTIONS
CONTENT MANAGEMENT
DATA MANAGEMENT
XML LABS

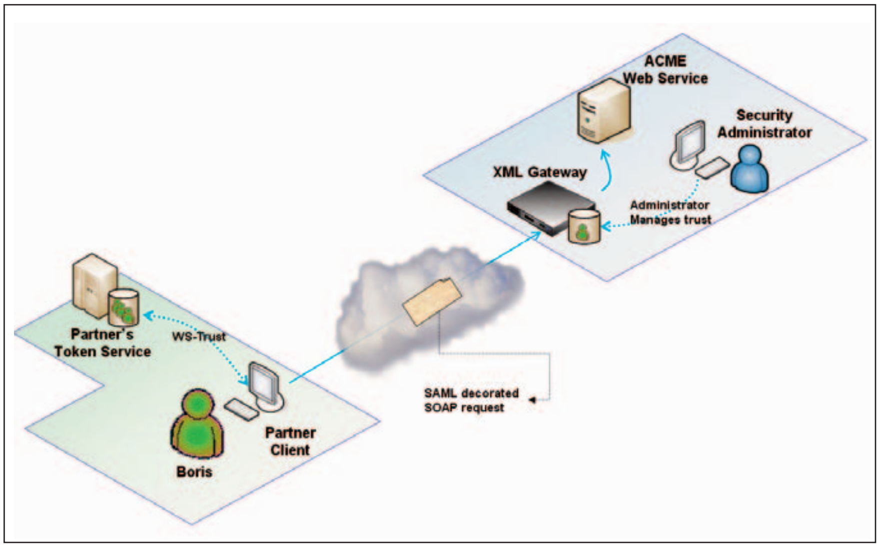


Figure 3 • Delegating trust to an XML gateway

mandates of the security manager is to ensure consistency of the security policies across all Web services exposed outside of the enterprise. If an XML gateway becomes the central entry point for all services, this becomes a lot easier to manage.

Securing the Last Mile

A concern that often arises when trust and authentication are delegated to an XML gateway is that it does not address the security between the gateway and the Web service itself. Some may argue that the added flexibility comes at the expense of *end-to-end security*. Although this is a valid concern that warrants close attention, securing the last mile between an XML gateway and a Web service is fairly straightforward. In fact, the ideal situation is one where security is dynamic up to the XML gateway (to reflect ever-changing security requirements), and static from the XML gateway down to the Web service (to minimize the Web service's maintenance). The following constitute a number of strategies to that effect.

A typical motivation for introducing an XML gateway in front of a Web service is to avoid the complexities of coding message-level security as part of the Web service. For this reason, transport-level security can be the solution of choice for securing the last mile. Presuming the Web service is being deployed over HTTP, transport-level security is well supported by most containers. The HTTP container is configured to only accepts requests through SSL (with client certificate), and to only accept the client certificate of the XML gateway. The administrator of the XML gateway simply configures the policy so that messages are routed over SSL using the gateway certificate as part of the SSL handshake. This provides confidentiality (transport-level encryption), as well as authorization (the Web service container requires proof of possession of the gateway's private key). Similarly, the XML gateway is configured to trust the SSL certificate of the Web service's container.

"XML gateways that support SAML-based identity federation transform software maintenance nightmares into security manager dreams"

A static security solution for the last mile that involves message-level security could be as simple as instructing the XML gateway to sign and encrypt all messages before routing to the endpoint. The Web service would only accept messages signed by the XML gateway, and the gateway would only accept responses signed by the endpoint service.

Although it typically poses administrative issues, it sometimes is possible to simply isolate the Web service from a network perspective so that the only way in is through the XML gateway. I would not recommend relying on this alone to secure the last mile, but this can be used in conjunction with other strategies.

Federation of Security Beyond the Textbook Scenario

Standards such as SAML and WS-Trust enable message-level solutions to federated-identity problems that arise when Web services span across multiple identity domains. By delegating these security aspects of a Web service to a manageable policy enforcement point, companies minimize the risk and the cost inherent to software maintenance. Further, this empowers the security administrator by providing enterprise-wide control of security policies.

The ACME/Partner scenario described here is simple and perhaps even cliché. However, SAML-based security tokens may very well add value to your application even if your application does not cross multiple corporate boundaries.

Web services promise to connect different computing environments and applications written in different languages. This heterogeneity by nature makes it difficult to find acceptable common security denominators. In transactions where multiple Web services are involved, SAML security tokens enable single sign-on (SSO) scenarios that remove the annoyance of having each transactional point implementing its own authentication. XML gateways can further connect disparate systems by bridging traditional Web and Web service SSO technologies (think "transport level to message level" and vice versa).

In the same way that SAML authentication statements facilitate SSO in Web services, SAML Authorization Decisions Statements can be carried alongside SOAP messages through multiple transaction points, thus relieving each Web service from having to manage its own authorization rules.

Author Bio

Long before terms like "Web service" and "SOA" were coined, Francois Lascelles was developing applications using SOAP and other XML standards. Francois joined Layer 7 Technologies in its earliest days and helped shape the vision of the SecureSpan product line. Today, as a member of Layer 7 Technologies' engineering team, Francois assists corporations in taking full advantage of Web service security technologies.

flascelles@layer7tech.com

Visit the *New*

www.SYS-CON.com

Website Today!

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

24/7

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

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

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

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

EDUCATION
The world's leading online i-Technology university

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

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

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

JUMP TO THE LEADING i-TECHNOLOGY WEBSITES:

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

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

WRITTEN BY **SELIM MIMAROGLU**

An Introduction to Berkeley DB XML

Basic concepts, the shell commands, and beyond

In this article I am going to introduce you to the latest version of the Berkeley DB XML, version 2.2.8. Berkeley DB XML (BDB XML) is built on top of the well-known Berkeley Database (BDB). BDB XML is an open source, native XML database. Like its ancestor, BDB, it's an embedded database. It provides APIs for the Java, C++, Perl, Python, PHP, and Tcl languages. It supports the popular XML query languages XQuery and XPath 2.0.

I will show you how to use BDB XML in two ways. This month I will introduce the BDB XML shell, and next month we will explore using BDB XML with Java. BDB XML has a lot of features, and I will try to cover the most important ones.

What's an Embedded Database?

Some of you may be familiar with embedded databases. An embedded database runs within another program. It is not a stand-alone server such as Oracle, DB2, or eXist. It is the programmer's responsibility to invoke the proper API for the database. For example, although Berkeley DB XML has transaction support, in order to activate it, API calls have to be made by the programmer. Many popular relational and native XML databases run in client-server mode. Oracle, DB2, and eXist run as client-server applications, while database server runs as a stand-alone application and many clients can connect to it. Clients and server programs run in different address spaces, probably even on different machines. A connection can be established between them using the JDBC, or ODBC protocols. In an embedded database such as BDB XML, this is not the case; clients and server both run in the same address space, and they are actually the same program. The client manages the Berkeley DB XML through API calls. Next month I will give some Java examples, which will clarify what an embedded database is.

Installation

Installing BDB XML on the Windows operating system is very easy. BDB XML comes with an installer for Windows. If you are using a UNIX-like operating system you have to build BDB XML from the source code. See “Building Berkeley DB XML for UNIX/POSIX systems” for detailed build instructions. I installed BDB XML on my laptop, a 1.4GHz Pentium M processor running on Windows XP Professional with 512Mb main memory, without any problems. The installer will automatically set the necessary *Path* and *Classpath* environment variables.

Using dbxml Shell

The shell is located under the *bin* directory of the BDB XML installation directory. It should be in your *Path*, so you can run it from any location at the Windows command line.

Invoke the shell by typing “dbxml,” and type “quit” to terminate the shell at any time. In order to list all available commands, type “help.” To obtain detailed information about a command, type help and the command name. For example “help createContainer” will display usage information of the createContainer command.

> “One of the new and cool features of the BDB XML is its ability to validate XML”

createContainer

First of all we have to create a container in which to store our data. A container in BDB XML is similar to a folder. Many XML files can reside in a container. In this article I am going to use XBench sample XML files and queries. You can download the XML sample files and the XML Schema from www.cs.umb.edu/~smimarog/xmlsample/.

Let's create a container called *xbench.dbxml*. You can, of course, name it anything you want.

```
dbxml> createContainer xbench.dbxml
Creating node storage container
  with nodes indexed
```

The user can choose between two

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.

storage types: *Wholedoc* or *Node*. The default is *Node* type storage. If you choose *Wholedoc* storage, the XML files will be stored as they are with all of the white space preserved. *Node* storage performs better. The BDB XML documentation suggests not using *Wholedoc* storage for documents bigger than 1MB.

putDocument

Now we are ready to put some XML data into *xbench.dbxml* container. In BDB XML the opened container is the default container. We have to open *xbench.dbxml* before working with it.

```
dbxml> openContainer xbench.dbxml
```

Actually, dbxml shell commands on Windows are case insensitive, so it's possible to write *opencontainer* instead of *openContainer*, but I am going to follow the established naming conventions here.

```
dbxml> putDocument sample_10
  C:\dictionary10.xml f
Document added, name = sample_10
```

The *putDocument* command takes three arguments: the first argument is the unique identifier of the document, the second argument is file name, and the third is either *f* or *s*. I chose the name *sample_10* as the unique identification of this XML document. C:\dictionary10.xml is the file name, including the path. Finally, the last argument *f* states that it's a file. Instead of providing a filename it's possible to provide XML data itself within quotes. There are several examples in the "Introduction to Berkeley DB XML" document, which uses an XML string instead of a file name.

> "Our query time improved from 47 seconds to 1.7 seconds – this is remarkable"

Query

As mentioned earlier, BDB XML supports XQuery and XPath 2.0 languages. There will be several examples for each language (see Listings 1-4). BDB XML found 28 results for this query. Note that in order to retrieve the results we have to use "print" command.

setVerbose

BDB XML will produce more explanation about what it does when the setVerbose feature is turned on. This command takes two numeric arguments: level and category. Using this command, let's get more information regarding the XQuery Example2 (see Listing 5).

No indices are used in this query. The execution time is around 48.5 seconds. Execution time is very high for this query for two reasons: this is not a trivial query, and there are no indices created yet. I will conceptually explain the possible execution plan for

HOME
ENTERPRISE SOLUTIONS
CONTENT MANAGEMENT
DATA MANAGEMENT
XML LABS

this query:

```
For each $entry as /dictionary/e
 For each descendant $desc of $entry
  For each value $val of $desc
   Check if ‘hockey’ is a substring
     of $val
    return $entry/hwg hw
```

There are three *for* loops in this algorithm. At the heart of the three loops there is a very costly *substring* (contains) operation. Considering that no indices have been created, execution time is pretty good. Creating indices improves the query performance dramatically (see Listings 6-7). (You can find more information about indices in the Indexing section of this article.)

Eager vs Lazy Evaluation

You may have noticed that after each query, BDB XML prints out the number of entries and the query evaluation method:

```
436 objects returned for eager
  expression
```

Eager is the default query evaluation method. Evaluating a query eagerly means that the BDB will store a result as soon as it finds any. In other words, eager evaluation grabs all of the results and stores them in a data structure, and they are available immediately after query execution. However, this is not the case when queries are evaluated lazily. In *lazy* evaluation, the database will not keep the results in a data structure. It will know how to get them (using pointers), but it will not do anything until the results are retrieved. Results are stored in sets. To get all of the results we have to iterate through the result set, using the next operator. This is what happens internally when we use the “print” command in the dbxml shell. It iterates through the entire set and gets every element of the result set. Thus, when queries are evaluated eagerly, the result set will be filled immediately after executing the query, as opposed to when the queries are evaluated lazily, and the result set either is empty or it has some of the results but definitely not all of them.

It may sound as though lazy query evaluation is never useful, but this is not the case. If you do not need all of the objects returned by the query, using lazy evaluation makes more sense. You can see this with the following query:

```
dbxml> setLazy on
Lazy evaluation on

dbxml> query ‘collection(“xbench.dbxml”)/
dictionary/e[contains(. , “the hockey”)]/
hwg/hw’

Query       - Starting query execution
Lazy expression ‘collection(“xbench.
  dbxml”)/dictionary/e[contains(. ,
  “the hockey”)]/hwg/h
w’ completed
```

Note that execution time for this query is ignorable (there is no execution time info printed out by the database). That’s because the actual results aren’t retrieved yet. BDB XML will retrieve the results only after “print” command. We know that there are 436 objects returned by this query. Instead of getting all of the results, let’s get only top eight of them. We can do this by using “print n 8” command.

> “Validation in Berkeley DB XML is very fast, which is a big time-saver”

XML Schema Validation

One of the new and cool features of the BDB XML is its ability to validate XML. First we need to create a container with XML Schema validation enabled. Listing 8 shows the XML sample (10MB XML sample with XML Schema, see the first entry in the References section) that I am going to put into this container.

This document is assigned an XML Schema. The part that shows this assignment is:

```
<dictionary xmlns:xsi=”http://www.
  w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation=
  ”http://www.cs.umb.edu/~smimarog/
  xmlsample/TCSD1.xsd”>
```

This schema is located at www.cs.umb.edu/~smimarog/xmlsample/TCSD1.xsd.

XML Schemas and XML documents can be located on the same machine or on different machines. In this example, XML Schema and XML data are located on two different machines.

```
dbxml> createContainer validate_xbench.
  dbxml d validate
Creating document storage container,
  with validation

dbxml> openContainer validate_xbench.
  dbxml

dbxml> putDocument dict_10_valid
  C:\dictionary10_schema.xml f
Document added, name = dict_10_valid
```

A natural question is whether it’s possible to add an XML document into this container without validating. Validation in Berkeley DB XML is very fast, which is a big time-saver. I have found that validating a document in BDB XML takes much less time than some commercial XML editors. However, it may be costly to validate each document when documents are huge. Besides, sometimes XML documents are not assigned to any schema. Listing 9 shows the XML sample (10MB XML sample in the References section) that I am going to put into this container.

Within this container we have two documents named dict_10_valid, and dict_10; the first document is validated, but the second is not. In some cases it’s desirable to restrict queries to a specific document in the collection. We can achieve this by using the “doc” function.

```
dbxml> query ‘doc(“validate_xbench.dbxml/
  dict_10”)//hwg’
733 objects returned for eager expression
  ‘doc(“validate_xbench.dbxml/dict_
  10”)//hwg’
```

By saying *doc(“validate_xbench.dbxml/dict_10”)*, the queries are restricted to run on only the dict_10 document.

Indexing

Indexing XML documents is very important for good query performance. In fact, indexing XML data is literally the most important task for the user. There are limited automatic XML indexing features in BDB XML, but indexing is best done manually by the programmer. In this section I will introduce you to the basics of XML indexing in BDB XML. Here is the format of an index:

```
[unique]-{path type}-{node type}-
  {key type}-{syntax type}
```

An index in BDB XML is composed of four parts:

- Path Types
- Node Types
- Key Types
- Syntax Types

Uniqueness

Uniqueness indicates that the value being indexed is unique in the XML document. For example, in an employees data set, employee number will be unique, along with the social security number.

Path Types

There are two available options for this attribute: *Node* and *Edge*. *Edge* is the path type index preferred by the query processor; therefore, it should be your first choice. It indexes the location between itself and its parent. For example, if you are indexing the *hw* element in the 10MB XML sample (in the References section), the following sub path will be indexed for *hw* entry:

```
hwg/hw
```

It doesn't make much sense to use a *Node* type path index because there are many *hw* elements present in the data. If there had been only one *hw* element, then it would have made sense to create a *Node*-type index instead of an *Edge* index.

Node Types

BDB XML supports three node types for indexing: element, attribute, and metadata.

Key Types

There are three different key types in BDB XML: equality, substring, and presence. You should use the equality key type (on *hw*) for queries such as:

```
/dictionary/e[hwg/hw="the"]
```

Use the substring key type (on *e* and all its descendants) for queries such as:

```
/dictionary/e[contains(., "hockey")]/
  hwg/hw
```

You should use the presence key type (on *qd*) for queries such as:

```
//q[qd]
```

Syntax Types

The most commonly used types are: string, decimal, float, date, and dateTime. For a complete list of syntax types, check the BDB XML documentation. The example in Listing 10 demonstrates the importance of appropriate indices.

Adding only one index only didn't help much. Having an *“edge-element-presence-none”* index on *qd* took almost as long as having no indices (~44 seconds). Verbose mode tells us that there are no indexes used in answering this query. We indexed only part of what we needed;, there is an index on *qd* only, but the query processor needs more than that. There need to be an appropriate index on *q* to respond to this query efficiently. Let's create an index on *q* too. We should have better query performance this time (see Listing 11).

Existing indexes are used efficiently for answering this query by the BDB XML query processor. Note the hexadecimal numbers. Our query time improved from 47 seconds to 1.7 seconds – this is remarkable.

Acknowledgements

Thanks to Glenn Hoffman and Saaid Baraty for providing useful comments.

References

- *XML sample documents and the XML Schema used in the article can be found at:* www.cs.umb.edu/~smimarog/xml sample/
- *Introduction to Berkeley DB XML:* www.sleepycat.com/xmldocs/intro_xml/BerkeleyDBXML-Intro.pdf
- *Getting Started with Berkeley DB XML for Java:* www.sleepycat.com/xmldocs/gsg_xml/java/BerkeleyDBXML-JAVA-GSG.pdf
- *XBench:* http://db.uwaterloo.ca/~ddbms/projects/xbench/

smimarog@cs.umb.edu

Listing 1: XQuery Example1

```
dbxml> query 'for $ent in (collection("xbench.dbxml")/
dictionary/e)
where $ent/ss/s/qp/q/qd="1900"
return
$ent/hwg/hw'

28 objects returned for eager expression '
for $ent in (collection("xbench.dbxml")/dictionary/e)
where $ent/ss/s/qp/q/qd="1900"
return
$ent/hwg/hw'
```

Listing 2: XQuery Example1 Results

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

Listing 3: XQuery Example2

```
dbxml> query 'for $a in (collection("xbench.dbxml")/
dictionary/e)
where contains($a, "hockey")
return
$a/hwg/hw'

935 objects returned for eager expression '
for $a in (collection("xbench.dbxml")/dictionary/e)
```

```
where contains($a, “hockey”)
return
$a/hwg/hw’
```

Listing 4: XQuery Example2 Results

```
dbxml> print
<hw>planks</hw>
<hw>gaul</hw>
<hw>minute</hw>
<hw>spot</hw>
<hw>slippery</hw>
<hw>adorns</hw>
<hw>homeward</hw>
<hw>serpents</hw>
<hw>gaunt</hw>
<hw>exclamations</hw>
<hw>disposer</hw>
<hw>walter</hw>
<hw>smilets</hw>
<hw>priamus</hw>
<hw>opinion</hw>
<hw>sebastian</hw>
<hw>sermons</hw>
<hw>metal</hw>
<hw>were</hw>
<hw>aching</hw>
<hw>summons</hw>
<hw>sap</hw>
…
```

Listing 5

```
dbxml> setVerbose 2 2

dbxml> query ‘for $a in (collection(“xbench.dbxml”)/
dictionary/e)
where contains($a, “hockey”)
return
$a/hwg/hw’

Query      - Starting query execution
Query      - xbench.dbxml - U : [1] 2
Query      - Finished query execution, time taken =
48431.2ms

935 objects returned for eager expression ‘for $a in
(collection(“xbench.dbxml”)/dictionar
y/e)
where contains($a, “hockey”)
return
$a/hwg/hw’
```

Listing 6: XPath Example1

```
dbxml> query ‘collection(“xbench.dbxml”)/dictionary/
e[contains(. , “the hockey”)]/hwg/hw’

436 objects returned for eager expression ‘
collection(“xbench.dbxml”)/dictionary/e[contains(. ,
“the hockey”)]/hwg/hw’
```

Listing 7: XPath Example1 Results

```
dbxml> print
<hw>minute</hw>
<hw>slippery</hw>
<hw>adorns</hw>
<hw>homeward</hw>
<hw>serpents</hw>
<hw>gaunt</hw>
<hw>walter</hw>
<hw>smilets</hw>
<hw>sermons</hw>
<hw>metal</hw>
<hw>were</hw>
<hw>aching</hw>
<hw>summons</hw>
…
```

Listing 8

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!-- generated by ToXgene Version 1.1a in Wed Jun 23
12:16:00 EDT 2004 -->
<dictionary xmlns:xsi=”http://www.w3.org/2001/
XMLSchema-instance” xsi:noNamespaceSchemaLocation=”http
://www.cs.umb.edu/~smimarog/xmlsample/TCSD1.xsd”>
 <e id=”E1”>
  <hwg>
   <hw>planks</hw>
   <pr>!gzkn+J|i@u</pr>
…
```

Listing 9

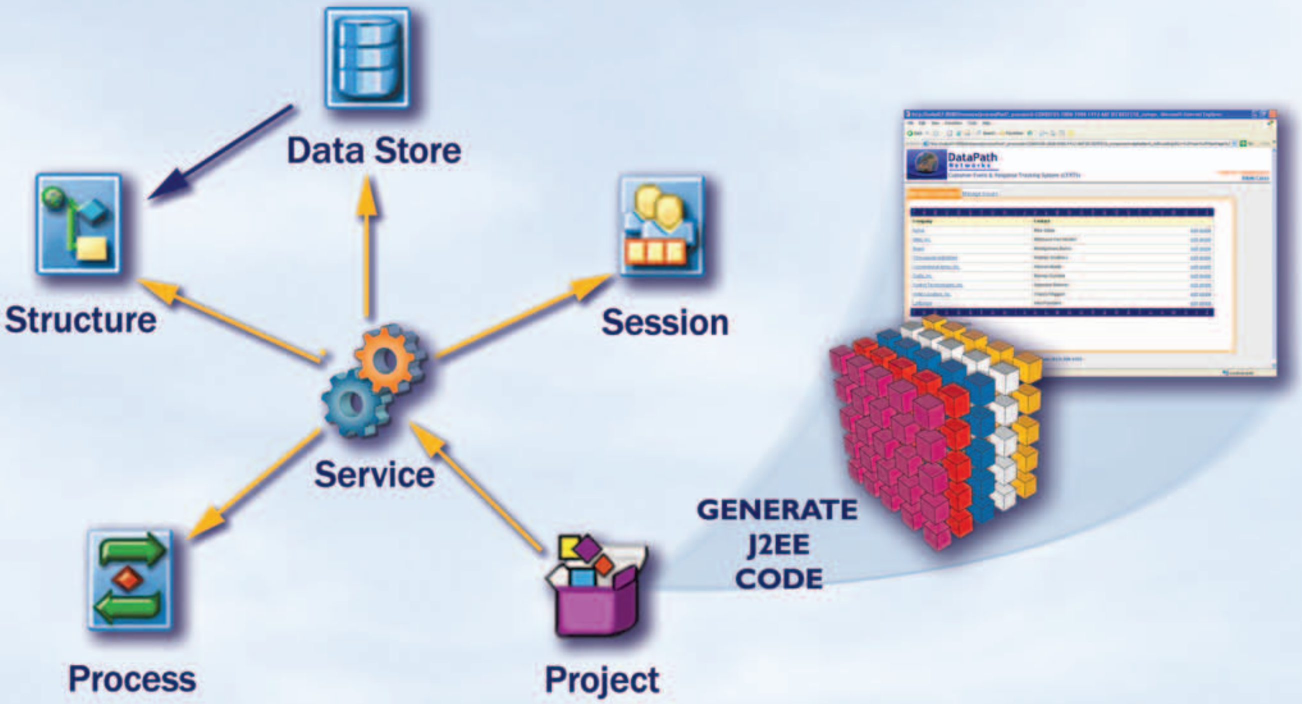
```
<?xml version=”1.0” encoding=”UTF-8”?>
<!-- generated by ToXgene Version 1.1a in Wed Jun 23
12:16:00 EDT 2004 -->
<dictionary>
 <e id=”E1”>
  <hwg>
   <hw>planks</hw>
   <pr>!gzkn+J|i@u</pr>
…


dbxml> putDocument dict_10 C:\dictionary10.xml f
Document added, name = dict_10
```

Listing 10: Example

```
 Find the q elements that have qd child element:
dbxml> query ‘collection(“xbench.dbxml”)//q[qd]’
Query      - Starting query execution
Query      - xbench.dbxml - U : [1] 3
Query      - Finished query execution, time taken =
47303.4ms

41427 objects returned for eager expression
‘collection(“xbench.dbxml”)//q[qd]’

dbxml> addIndex “” qd edge-element-presence-none
Adding index type: edge-element-presence-none to node:
{}:qd

dbxml> query ‘collection(“xbench.dbxml”)//q[qd]’
Query      - Starting query execution
Query      - xbench.dbxml - P(q.qd) : [1] 3
Query      - Finished query execution, time taken =
44062.3ms
41427 objects returned for eager expression
‘collection(“xbench.dbxml”)//q[qd]’
```

Listing 11

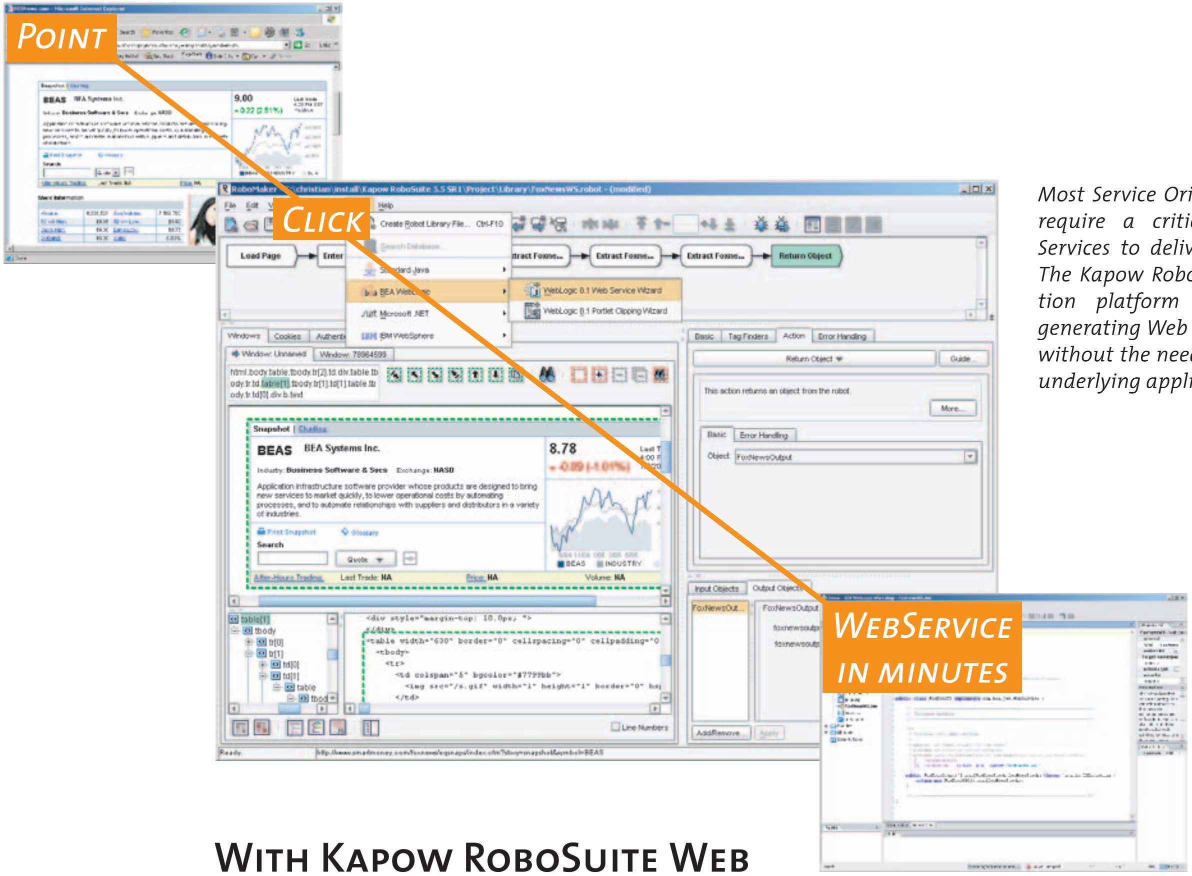
```
dbxml> addIndex “” q edge-element-presence-none
Adding index type: edge-element-presence-none to node:
{}:q

dbxml> query ‘collection(“xbench.dbxml”)//q[qd]’
Query      - Starting query execution
Query      - xbench.dbxml - P(q.qd) : [1] 3
Query      - xbench.dbxml - P(q) : [41427]
3(0x04020213) 3(0x04020218) 3(0x0402021E) 3(0x04020222)
3(0x04020227) 3(0x0402022E) 3(0x04020233) 3(0x04020238)
3(0x0402023E) 3(0x04020243) ...
Query      - Finished query execution, time taken =
1766.52ms
41427 objects returned for eager expression
‘collection(“xbench.dbxml”)//q[qd]’
```


Data Store
Structure
Session
Service
GENERATE
J2EE
CODE
Process
Project
DataPath
Networks

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.

SYS-CON

NOW INCORPORATING XML JOURNAL

SOA Web Services Journal

December 2005 Volume 5 Issue 12

Web Services – Soup to Nuts

Simplified Web services development using visual tools

pg. 20

Networks for Service-Oriented Architecture

The impact of Web services on wide-area networks

pg. 30

Bringing SOA to Life: The Art and Science of Service Discovery and Design

Practical guidelines and experiences from real-world SOA projects

pg. 42

The Global SOA

PG. 10