SYS-CON

THE WORLD'S LEADING MAGAZINE DEDICATED TO WEB SERVICES TECHNOLOGIES

FOR ALL THE LATEST NEWS AND INFORMATION VISIT:

www.WebServices.SYS-CON.com

SOA Web Services JOURNAL

JANUARY 2007 / VOLUME: 7 ISSUE 1

Are You Being Served?

PLEASE DISPLAY UNTIL MARCH 31, 2007

$6.99US $7.99CAN

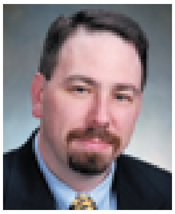
01>

0 71486 03420 9

SOA Web Services JOURNAL
www.WebServices.SYS-CON.com

EDITORIAL OFFICES
SYS-CON MEDIA
577 CHESTNUT RIDGE ROAD, WOODCLIFF LAKE, NJ 07677
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
Woodcliff Lake, NJ 07677 and additional mailing offices
POSTMASTER: Send address changes to:
WEB SERVICES JOURNAL, SYS-CON Publications, Inc.
577 Chestnut Ridge Road, Woodcliff Lake, NJ 07677

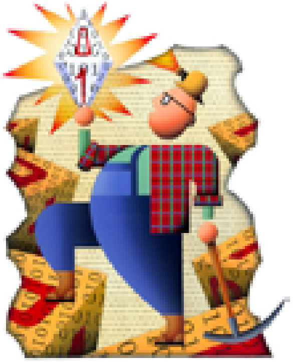
Building Codes



WRITTEN BY **SEAN RHODY**

There's a guy I know who's incredibly gifted when it comes to building things. I've watched him repair a barn, build a new shed, put up a roof – you name it; if it can be done with wood and tools, this is the guy to do it. And I've never seen him use a plan – he just knows what to do.

The trick of it is, he never takes on more than he can do by himself. While he's very skilled, probably the most tangible evidence of that skill is he knows better than to overreach himself. He knows that what he's doing is within his capabilities, and there's no way he could build an office building or a skyscraper by himself without plans, assistance, and government approval.



The task of organizing services in an SOA environment is a lot like building something. You start with elementary services – security, auditing, some fundamental business processes – and you put together a composite application, much like my friend knows how to put together a shed in the yard.

A shed is useful and does real work (well, it actually stands there, but the work it does in keeping the rain and snow off of things is good enough). Likewise, a simple composite application can be useful and do real business tasks, like taking an order or updating a customer profile.

But if everyone builds sheds without plans, you're bound to encounter someone like me who should never operate power tools. Then the neighborhood goes downhill rapidly. It's much like that without Business Process Management in the SOA world.

Business Process Management, and the things that usually accompany it such as unified security and overall management, is the blueprint and design of the SOA world. Just as in construction, you wouldn't want anyone to build a large-scale building without plans, reviews, approvals and designs based on years and years of engineering knowledge; you also wouldn't want an enterprise to go full scale into the SOA world without BPM.

Business Process Management provides the rigor and discipline necessary to build a successful enterprise architecture using SOA concepts. The design of a business process in a BPM tool allows a company to realize what their service catalog is, what it should be, and how they can use their current environment to accomplish the purpose of the process most effectively. It provides the tools necessary to utilize underlying services and coordinate the transactions and information that are necessary to accomplish the completion of a business process.

A process may encompass many things. There may be multiple transactions. Consider purchasing a book online; in addition to a transaction to actually locate the book in the warehouse and pack it, there would also be a transaction regarding shipping, another for inventory, and yet another to do the credit card processing. All of these things are likely to be separate, short-running processes, but the business process can be much longer lived and involves invoking all of them at the appropriate times. It may also involve revoking or rolling them back under appropriate circumstances, such as in the case where you're buying a book by Sean Rhody and it's on permanent back order (sorry about that folks).

Likewise, the process of governance that goes along with agreeing to the design of a business process is similar to the system of permits and reviews that a contractor needs to go through to get approval of the construction. Governance of services is critical when we have to deal with things that cross organizational boundaries and may be shared between organizations (i.e., resolving the age old question – "who's going to pay for that").

This issue will address some of the challenges of BPM. In the mean time, I'm going to find my buddy and sell him my power tools before it's too late. ■

About the Author

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

SOAs Need an Intelligent SON

A new service-oriented network approach is crucial to providing a nimble infrastructure that can accommodate SOA's agility and flexibility

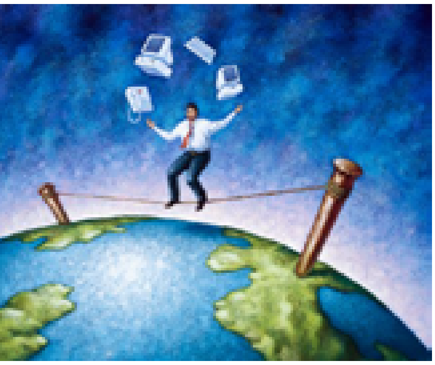
WRITTEN BY **JEFF BROWNING**

Service Oriented Architecture (SOA) is the most significant transformation in IT application architecture in the past several years – representing open standards and a more loosely coupled approach to designing, developing, and deploying applications.

While attention is given to the application side of SOA, an often-overlooked ingredient needed to ensure its successful delivery is the underlying network infrastructure. As important as the application tier is, a new service-oriented network (SON) approach is crucial to providing a nimble infrastructure that can accommodate SOA's agility and flexibility. Transforming the application tier without considering the implications of outdated network design poses a significant risk to successful SOA deployments.

A principal feature of a SON is having network devices that are far more intelligent than those found on traditional networks – devices that recognize that SOA traffic is fundamentally different from other network traffic. The SON must clearly and unambiguously understand Web Services language while loosely coupling network resources. Then the SON must virtualize these resources and the commands and business logic to execute the transaction.

Oddly enough, many IT managers have yet to grasp this reality: Yesterday's network won't support tomorrow's applications. In the last year both InfoWorld and Network World did major research surveys of the SOA plans of senior IT managers. When asked what obstacles might litter the path to broader SOA deployment, respondents in both surveys put factors like security, organizational barriers, and slow standards development at the top of their lists.



Of course, history might play a part in this mindset. Not that long ago, the network filled the simple role of transporting application requests from one endpoint to another. Most network devices were virtually "set and forget" tools that directed traffic over various ports from outside the network to internal application assets. Load balancers – the network technology most relevant to application architects and developers – were basically installed, set up for "round robin" load balancing, and never touched again.

Now, as network devices are becoming more intelligent, they can work more cooperatively with applications by offloading advanced needs such as application persistence, SSL acceleration, compression, and caching. Some even have bi-directional proxy capabilities to control application flows, enabling applications and network to work in concert.

The Old School Persists

Unfortunately, many organizations still rely on old devices that aren't optimized to handle

SOAWebServices JOURNAL
www.WebServices.SYS-CON.com

CORPORATE

President and CEO
Fuat Kircaali fuat@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 Sales Director
Megan Mussa megan@sys-con.com

Advertising Sales Manager
Andrew Peralta andrew@sys-con.com

Associate Sales Managers
Kerry Mealia kerry@sys-con.com
Lauren Orsi lauren@sys-con.com

SYS-CON EVENTS

Event Manager
Lauren Orsi lauren@sys-con.com

CUSTOMER RELATIONS

CIrculation Service Coordinator
Edna Earle Russell edna@sys-con.com

SYS-CON.COM

VP information systems
Robert Diamond robert@sys-con.com

Web Designers
Stephen Kilmurray stephen@sys-con.com
Richard Walter richard@sys-con.com

ACCOUNTING

Financial Analyst
Joan LaRose joan@sys-con.com

Accounts Payable
Betty White betty@sys-con.com

SUBSCRIPTIONS
SUBSCRIBE@SYS-CON.COM
1-201-802-3012 or 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

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 www.SYS-CON.com

_INFRASTRUCTURE LOG

_DAY 15: This project is out of control. The development team's trying to write apps supporting a service oriented architecture...but it's taking FOREVER!

_DAY 16: Gil has resorted to giving the team coffee IVs. Now they're on java while using JAVA. Oh, the irony.

_DAY 18: I've found a better way: IBM Rational. It's a modular software development platform based on Eclipse that helps the team model, assemble, deploy and manage SOA projects. The whole process is simpler, faster and all our apps are flexible and reusable. :)

_The team says it's nice to taste coffee again, but drinking it is sooo inefficient!

IBM

Rational®

Download the IBM Software Architect Kit at:
IBM.COM/**TAKEBACKCONTROL**/FLEXIBLE

XML Web Services traffic effectively. With Web Services and XML, a tremendous amount of context is embedded in the requests passed via SOAP and HTTP. From self-describing data to selectively encrypted elements, understanding the traffic and being able to alter its flow to direct requests and responses better is critical to a successful SOA. Furthermore, in an SOA world where faster application development and service reuse speeds up application change, the burden of configuration change management is significantly increased. A new breed of product and overall network design is needed to support SOA implementations.

There are two critical reasons why last-generation network technology knows little about Web Services. First of all, most traditional devices are connection-specific. They can't understand the traffic flowing through them natively, and rarely dig deeply into the application payload where much of the application logic resides in Web Services traffic.

Second, most network products don't provide the Web Services-friendly API needed to support the constantly changing configuration needs of Web Services, whether requested individually or as part of composite services.

A Real-World Example

Why is this so critical for your SOA plans? Well, consider a basic real-world scenario. You've built a mission-critical Web Service using SOAP that brokers transactions between your primary business partners and internal order processing system. But traditional network devices can't tell the difference between a Web page request and a SOAP application request. This lack of understanding will play a significant role in how reliability is orchestrated with your SOA.

One day, an unforeseen market event (say an interest rate change for a financial services firm) floods your system with transactions. Older load balancing devices might distribute these requests as needed to the next available server that can support them. But if the servers become overloaded, the older devices make it hard to understand why. Further, based on the unique priority of different Web Service requests, it's even more difficult to know where to send them to comply with the business requirements influencing your SOA strategy.

Intelligent full-proxy devices, however, can identify SOAP faults or exceptions and associate them with the actual application server returning the SOAP error codes. Advanced devices can even queue these faults and retry requests to another service endpoint before returning an error to the requesting client. This ensures a better client experience and makes error logging and troubleshooting easier due to native Web Service fluency.

Smart network devices can also prioritize requests based on enforced rules and logic, removing the burden from the application servers that support the requests. Combine this application knowledge with bloated XML traffic and it's easy to see how a high-performance network device can come in handy to crunch the bits into faster, more secure, more intelligent application flows.

More Than Intelligence

But a network technology must be more than just intelligent to support SOA effectively. In a traditional network load balancer world, configuration changes are made via a GUI or command-line interface. Efficiency-focused network operators commonly collect hard-coded scripts written for a specific configuration of IP addresses and servers. These usually work for static environments and events, but the approach conflicts with the agility and flexibility that SOA enables. Architecturally, it opposes the loosely coupled philosophy of SOA. The brittle coupling breaks as changes are made to support business logic, creating more of the challenges that SOA is intended to remove.

The use of SNMP as a common interface language for network integration is also inherently insecure, and requires developers to learn management information bases (MIBs) that define how and what to communicate as defined by the network device.

To support SOA, a network API – or network control plane – that openly understands Web Services language and loosely couples the network resources, IP addresses, pools of servers, and configurations is needed. It will automate mission-critical configurations and make the critical connection between the application realm and the network realm, and is critical to building and deploying a SON.

To understand SON, one must view the network and servers as SOA sees applications. The underlying network and associated resources required to deliver SOA applications are active participants in the SON architecture model, which also delivers agility and flexibility through virtualization of key infrastructure resources, commands, and business logic.

From an agility perspective, this approach enables a SON to expand and contract dynamically based on any variety of performance, availability, or security issues that may develop to ensure the successful delivery of services in the SOA. Knowledge gleaned by intelligent devices about various Web Service needs (plus an API to execute dynamic configuration changes) provides the orchestration engine to support SON. New applications can be deployed faster, without a complete reconfiguration of the network resources.

To illustrate this, visualize a data center with 10 servers and a single Web Service replicated across each server. The network device virtualizes the separate server IP addresses as one "virtual IP" supported by the "pool" of servers. Requests made for a Web Service defined as one virtual IP are directed to the servers best able to support them.

Using our previous financial industry example, as requests increase, the network device can monitor requests, look for errors, and invoke configuration changes to alter device setup, adding more standby servers to the pool or even redirecting new requests to another data center running additional instances of the Web Service. Additionally, prioritized requests based on client request ID or other factors could be sent to an entirely separate pool of servers hosting the service in a manner optimized for high-demand scenarios.

Based on business priority, the number of server resources, the priority, and dynamic changes to the configuration to support automated change can be done seamlessly, with the service and network working together through more network intelligence and control.

The Bottom Line

To support a SOA, some key ingredients include a network device that is highly intelligent, able to understand SOA service flows, speak the same language as the applications, and understand the unique needs of Web Services traffic – all of which will go a long way in helping you design a SON to support your SOA. ■

About the Author

Jeff Browning is director of product management for Seattle-based F5 Networks, a leading provider of application delivery networking products.

AJAX + SOA: The Next Killer App

SOA lacks a face; that's where AJAX comes in — it puts a face on SOA

WRITTEN BY **JOHN CRUPI**

Enterprises trying to improve business unit productivity and the reuse of IT assets continue to struggle. IT organizations have achieved some success by attacking these challenges with Service Oriented Architecture (SOA), but in most cases have still only exposed small portions of the overall IT service portfolio. Much of this struggle has been to deliver a "just enough" SOA to the business unit to improve its ability to build applications and features to get to market faster, better, and cheaper. And as we've learned, accomplishing this is easier said than done.

The fact is that SOA is middleware — and middleware traditionally relies on more middleware to translate data into a consumer-friendly state. It's certainly a major disappointment when you finally get your SOA right only to find that building a composite application requires using a portal (middleware) and/or orchestrating it with a BPEL engine (even more middleware). Worse yet, you may be in an organization that deploys a UDDI registry and registers a bunch of Web services. Unfortunately, in most cases there are very few applications built to actually consume these services. How can this be?

Should we conclude that something is wrong if applications aren't being built to consume these SOA services? Is it too difficult for business unit developers to build applications that directly consume SOA services, forcing us to rely on IT to create these applications? Is the absence of a SOA governance architecture holding us back? I think the answer is "yes" to all of these questions. And there is one standout reason: it has simply been much too difficult for the business unit developer to consume and leverage the SOA services exposed by IT. What's been missing is an easy way to put a "face" on SOA — and that's precisely the benefit of using AJAX in combination with SOA.

SOA services are typically implemented as loosely coupled Web services that encapsulate and expose business functionality. This sounds relatively straightforward, but is quite complex and difficult to achieve in practice. Developers frequently argue about the granularity of SOA services, but most now agree that "business-grained" granularity is the most appropriate. However, it still takes a great deal of domain expertise and collaboration with the business unit to size services properly.

Fortunately, there's been a recent surge in SOA interest. Perhaps enterprises are finally coming to recognize that SOA can really help the bottom line. Maybe it's being driven by better tooling and Web services evangelizing by Amazon, Yahoo, and eBay. Or could it be AJAX? Of course, it's AJAX — why else would I be writing this article? Seriously, I do think that AJAX is driving a renewed interest in SOA, especially in the mashup space. But how can two very different technologies combine and connect to provide something far greater than the parts? Take a look at Wikipedia's current definition of AJAX. It talks about Web pages, but there's no mention of SOA. It says:

"AJAX, shorthand for Asynchronous JavaScript and XML, is a Web development technique for creating interactive Web applications. The intent is to make Web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire Web page does not have to be reloaded each time the user makes a change. This is meant to increase the Web page's interactivity, speed, and usability."

The absence of SOA in this definition comes as no big surprise since the early buzz around AJAX has centered on enhancing Web page functionality and usability. This has been showcased in many applications such as Google Maps, Flickr, and Yahoo Mail. It's not these consumer-facing applications that personally get me excited about the potential of AJAX though. Instead, it's the business applications that run behind the corporate firewalls that can really exploit and benefit from AJAX, because it gives us two key features: a client-side programming model and the ease of making asynchronous calls to the server. These two key capabilities — the ability to apply logic in the client (browser) and to access server data without disrupting the Web page — are what al-low the new Web 2.0 paradigm to open up so many enticing possibilities for rich enterprise applications.

Earlier I said that SOA was lacking a face. That's where AJAX comes in — it puts a face on SOA. Let me explain this a bit further. Think about what happens when SOA services come online. They usually get registered in a registry/repository (if we're lucky) and become available for consumption. For instance, take a look at what StrikeIron (www.StrikeIron.com) provides. StrikeIron has created a "Web services marketplace" for the general population. At first glance, StrikeIron's catalog looks like a list of mini-business applications. But then you realize that these aren't applications — they're actually Web services. The concept of a company providing WSDL/REST Web services for general consumption makes a lot of sense. But before we get too excited, let's take a look at what's selling. According to StrikeIron, which licenses access to these services, its most popular Web services include:

- *U.S. Address Verification*
- *Global SMS Pro*
- *Sales and Use Tax*
- *E-mail Verification*
- *Reverse Phone Lookup*

Without a doubt, all of these Web services are useful and can be applied to many different domains. But at the same time, they're also pretty commoditized. In other words, I may not care about who provides these services, only that I get the desired information. On the other hand, would I use just any Web service to transfer money from my checking account to my savings account? I don't think so. I need to trust the service, so I'd have to have some sort of relationship with the vendor providing that service. This "circle of trust" between me, the consumer, and the service provider is exactly the kind of relationship enterprises have internally and with partners.

The same approach could be taken by enterprises to begin offering their Web services to a wider audience. Through a Web services marketplace, enterprises could register various Web services that would normally only be available internally and/or to their partners. The marketplace vendors would obviously love to see this happen, but more importantly, I see this as an opportunity to begin to apply AJAX + SOA to drive a whole new class of Web 2.0 business applications.

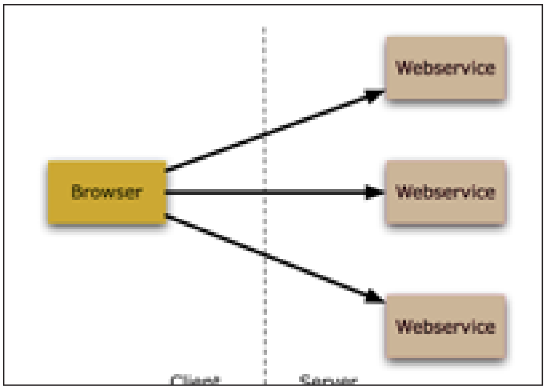


Figure 1

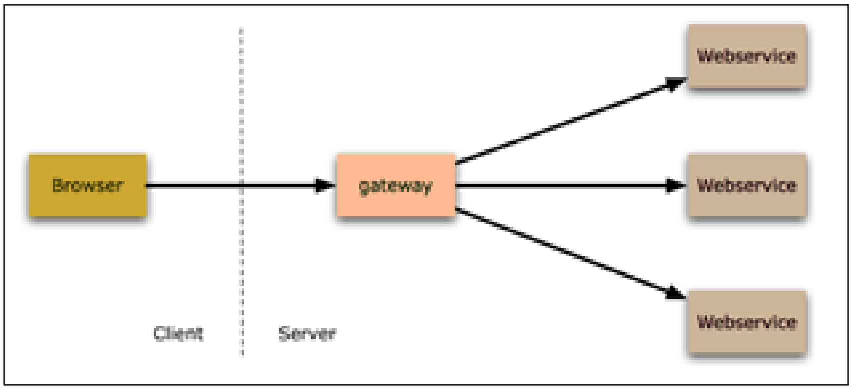


Figure 2

For the first time, it's starting to feel like our application de-velop-ment and SOA efforts are coming together. We have business functionality represented in a reusable form – SOA services. We have ubiquitous connectivity – the Web. We have what's turning out to be the new application container – the browser. We have a programming model in the application container/browser – JavaScript. And they are all using open standards! What more could we ask for? Actually, quite a bit more.

In particular, I'd like to see a more rapid solution for developing applications based on all of this — a way to build applications without having to rely on more middleware to integrate with the SOA services. This is what I'll call the ability for Web applications to perform "direct connect SOA." Direct connect SOA conveys the ability to punch through the traditional curtain of portals and heavyweight process engines and directly (at least conceptually; more about this later) access SOA services. I don't just mean Web services either. It could be BPEL orchestration services, coarse-grained POJO services, RSS feeds, or anything else that can be exposed as a "service," albeit at the right level of business granularity. And of course the interfaces should be exposed using open standards.

This novel development and runtime model creates a new way to build application-driven composite applications. It has the appeal of client/server, without all the traditional heavyweight client/server baggage. It runs in the browser and is delivered on-demand.

We've all heard a lot about "composite applications" over the past few years. But most vendors have been talking about composing services as a way to re-factor their hosted services into more palatable services or portal applications. Let me clarify by using an analogy.

AcmeGrid, a fictitious grid vendor, provides a service grid that lets you run your applications as services. Its customers tell it they want a way to "compose" a combination of services into coarser-grained services. So, naturally, AcmeGrid announces an Eclipse-based AcmeGrid Composite Application Builder (CAB). Interestingly, CAB looks a lot like a BPEL designer, but has tighter integration with the AcmeGrid deployed services. Pretty slick, but it's not really an application as much as it is a service. In essence, CAB is more like a service builder. But who wants a service builder when we're trying to build applications? Soon, another fictitious vendor, we'll call them AcmePortal, announces its Por-tal Composite Application Builder (PCAB). It too releases an Eclipse-based designer that looks and feels like a BPEL designer, but this one knows how to build portals. In many cases, a portal is fine as an application. But if you're forcing a portal to be an application, it's just adding unnecessary weight.

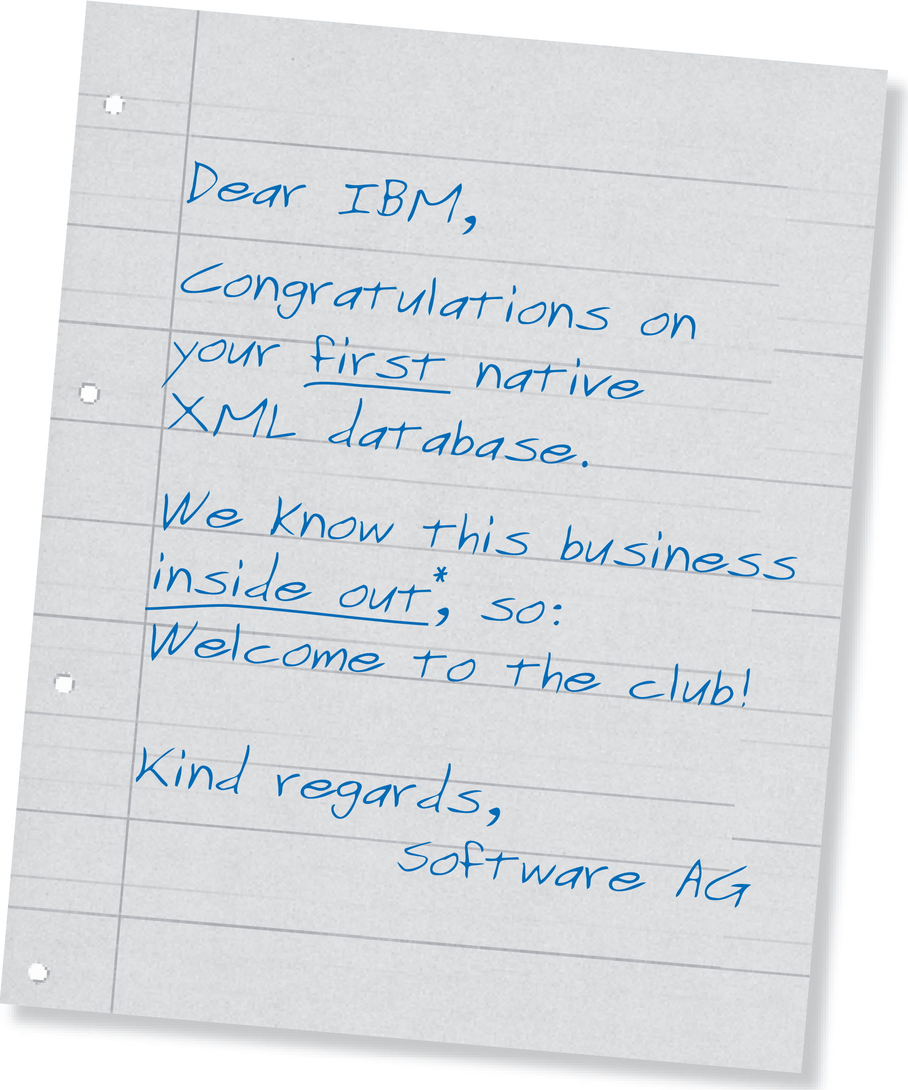
What I really want is a user-based composite application, not a middleware-based composite application. To make this happen, I need a development and runtime platform that not only speaks AJAX and SOA, but governs the two as well. Some vendors promote the concept of AJAX applications calling WSDL-based Web services directly from the browser, essentially making SOAP calls. This approach even has a name – "client/SOA." This may be fine for simple non-enterprise or pure consumer applications, but it's a no-go for the enterprise. Why? Because when you call Web services directly from the browser, governance is left to the browser — which is another way of saying, there's no governance. Figure 1 shows ungoverned Web service consumption. I've never run into an enterprise that doesn't govern its services and strongly doubt that enterprises would allow this to happen merely because we have the technology to do so efficiently. If you doubt me, just remember that enterprises never opened up the firewall (for applications) to anything but HTTP and SSL. No matter what we told system administrators, no other ports were being opened.

So what we're talking about isn't just AJAX + SOA. It's really a platform that provides the governance necessary for AJAX and SOA to co-exist in the enterprise. This is a platform that provides the ability to consume SOA services on behalf of the client, but also governs the service consumption. Figure 2 shows how Web services can be governed by a service gateway. A service gateway is a server-side abstraction that governs and mediates service access on behalf of the client, which in this case is the AJAX application in the browser. The beauty of using a serv-ice gateway is that you're not restricted to accessing only services run-ning in the enterprise. The service gateway governs any service that's registered in the enterprise. In WSDL-based Web services, an enterprise would register the WSDL, and WSDL provides the bind to the service at runtime. This might be a service running in the enterprise's data center, but it could just as easily be a service running in a partner's data center. If the enterprise allows (governs) applications to access services, it doesn't matter where they're running.

I hope you're starting to appreciate the power of combining AJAX and SOA – specifically, how the two can co-exist and deliver new Web services–based applications with the governance that enterprises require. I truly believe we're entering a new era with amazing opportunity. Web 2.0 social networks, photo-sharing and tagging are great, but the real corporate impact comes in a form of Web 2.0 for the enterprise. 🟥

About the Author
John Crupi is the CTO at JackBe.
www.jackbe.com

tamino
XML Server



Dear IBM,

Congratulations on your first native XML database.

We know this business inside out*, so: Welcome to the club!

Kind regards,
Software AG

*For seven years we have fulfilled customer needs with our powerful high-performance Tamino XML Server. Welcome to the club, IBM.

>> www.softwareag.com/Tamino

SOFTWARE AG

Are You Being Served?

Customer-Centric SOA Management

WRITTEN BY **PAUL LIPTON**

"A child of five would understand this. Send someone to fetch a child of five." – Groucho Marx

People have begun to understand that a properly implemented SOA has the potential to improve business agility and adaptability to changing business conditions, but we're still suffering from at least one innate prejudice common to IT folks. That prejudice comes, in part, from our own fascination with the technology we use. Perhaps we need to view the goals and challenges that SOA brings with the same simplicity that small children selling lemonade from a box on the sidewalk have for at least children recognize that it's neither the lemonade nor the stand that's the center of their business, but the customer. Changing our natural biases as IT people to conform to this business reality can help us understand what's most important in a SOA and how that SOA needs to evolve to be successful.

Designing Customer-Centric Business Services

It's helpful to understand SOA as a means to serve the business by serving both the internal and external business service *consumer*. Perhaps a more useful way to think of the business service consumer is as a business service customer. That way, we may go beyond the dutiful attention that we in IT too often extend to the customer and move him from the periphery to the center of our attention and actions.

Business services are everywhere and business was familiar with the concept long before we IT people got our hands on it. SOA architects would be well served in taking a lesson from more modest, yet proven "real-world" designs. For example, in a large, modern restaurant, the business service of "placing a lunch order" might be an aggregation of both serial and parallel business services such as initiating food preparation in the kitchen, bringing the beverage while the food is being prepared, bringing the food, placing a follow-up order for desserts and coffee, presenting the bill, and getting paid for the meal.

Notice that the descriptions of the customer facing business services that are visible to the customer are customer-centric. An IT-centric view would "accept a lunch order" rather than "place a lunch order," a small but important distinction that most restaurants, unlike IT organizations, are instinctively wise enough to avoid. The service description should always be about the customer, not the service. Most waiters don't say "I'm ready to accept an order now" because the transaction is about the customer, not the restaurant so the waiter is more likely to ask "May I take your order?" Reorienting your naming conventions to be more customer-centric, at least for customer-facing services, is left to the reader. It can be a truly useful exercise, one that can create a better, more customer-focused architecture, but I warn you the IT-focused perspective can be difficult to shake off.

Business Service Roles

It's important to realize that there are many *service customers* just as there are many *service providers*. We're intuitively aware of these two roles, at least to a degree. And of course, business service providers can also be business service customers. For example, a credit card approval business service might also be the customer of a geographic service to check if a credit card is being used outside its usual geographic area. In combination with other pattern analyses, this might help detect a stolen credit card.

However, there's at least one other role to consider, one that's often forgotten, and that's the *service owner*. The service owner is the business entity that bears business responsibility for that service to the customer. For example, in a restaurant the service owner might be the restaurant manager or owner, while the waiter is the part of the restaurant "infrastructure" that provides a key business service. Of course, the roles of service owner and service provider might be one and the same, but in larger businesses that is less likely. In many large businesses it's common for IT to provide a business service that's owned by a line-of-business (LOB) manager or group within the business organization such as product marketing.

Service Level Agreements

We don't always focus on satisfying our service customers as much as we should, but when we do get serious about our customers we often begin by defining a set of agreements outlining what is expected between business service provider and various business service customers under certain conditions and at certain times. These agreements are called Service Level Agreements (SLAs). SLAs can be essential to the success in a SOA.

In simple terms, a SLA is a set of expectations often referred to as Service Level Objectives (SLOs). These expectations should be measurable and reasonably attainable. Many SLOs articulate the expected reliability and availability of a business service based on calendar functions, expressing expectations for certain times of day and certain days in the week, month, quarter or even year. These expectations are often expressed in terms of the application-specific data (metrics) obtained from various components in the SOA. For example, performance and availability expectations could be expressed in terms of an acceptable range of average response times or the number of failures per hour.

What creates customer satisfaction at any restaurant? To answer that question, we should think about what makes customers different from one another. One of the most important differences is their level of expectation. Customers often have different criteria for success. The quality of the food, the pleasantness and professionalism of the wait staff, how hot the soup is, and the diversity of the wine cellar may all be contributing factors. A successful restaurant will likely need to treat the soup lover differently than the wine lover.

For that matter, does the same customer always have the same criteria for success? Customer service expectations often vary according to business circumstances. Time and day are important factors. A business lunch and a romantic dinner have different criteria for success even to the same customer. For a business lunch, rapid service might be important while the criteria for success in the later case might be quality of food and unobtrusive service. Similarly, knowing which customer is a "platinum customer" can be essential for the success of any business service. In plain language, the expectations of the customer can best be met if we know the customer's identity and have some awareness of his business transaction – its context and circumstances.

Similar criteria for success exist for SOA business services. Knowing the identity of a customer and applying a consistent security policy is a starting point, hence the necessity of cross-enterprise SOA security policy decision and federation solutions such as the popular SiteMinder and TransactionMinder products from CA, as well as identity products from Oracle and other vendors. However, besides these important identity issues, one must also provision appropriate business transaction monitoring software to help your customer business services meet their SLAs through appropriate monitoring of the actual business transactions from request to response as they flow from the customer through various levels of the SOA and the underlying infrastructure.

Different Types of Management – A Culinary Example

It's common for architects and other IT leaders responsible for SOA initiatives to underestimate the diversity, depth, and volume of the metrics needed to describe the state of their SOA and how it's serving the customer. Think of the complexity inherent even in the simple restaurant example. So many things can go wrong and larger restaurants and hotels may have many chefs and other food handlers, not to mention a wide variety of ovens, refrigerators, mixers, and other equipment.

So why might some hot dishes arrive cold and violate customer expectations? Well, monitoring the business services – like a SOA management product would – may not provide sufficient information. At the service level, it's helpful to know that all customer orders usually arrive on time. It might be possible to narrow the problem down to a particular waiter (a particular business service), but that doesn't explain why the waiter appears slow only at certain times and not at other times. Waiters can explain themselves in a way that software can't, but if the waiter couldn't speak we wouldn't be much closer to solving our problem. Think of this service-level management as just the first step, albeit an important step, towards solving the problem.

It's also helpful to know that the chefs all say that they're feeling fine and that all the ovens and stoves appear to be working. That's equivalent to traditional enterprise management – and is important information to have. The restaurant service provider will surely want to make sure that his supporting restaurant infrastructure is functioning overall. However, to understand and fix the problem, the restaurant must do more than just measure the time the order received to when the food is delivered. This information is useful and might certainly tell it that service is slow, but it usually doesn't help it understand why. It still can't solve the problem unless it can track and trace the individual transactions (the customer's individual orders) as they flow from the starting point at the table through the entire restaurant infrastructure back to the customer for consumption.

It's only by understanding and following the flow of customer transactions in the restaurant that the restaurant's management can begin to understand that one of the waiters doesn't know how to use the credit card machine properly and uses the same slow, inefficient procedure every time. This means that when the waiter processes one customer's credit card payment, the hot entrees for his other customers are cooling in the kitchen. This waiter then falls behind in taking orders and compensates by hurriedly gathering and dumping them all at once on the harried chef. In turn, this same harried chef cuts his own corners, resulting in more improperly prepared food. So, complex cascades of serious problems all emanate from one problem buried deep inside the business. And yet these events are seen as relatively rare and benign in isolation. It's only when viewed as part of the overall customer business transactions that their true impact is understood.

> **"Customer-centric enterprise application management must go beyond other forms of management by focusing on fine-grained, component-level knowledge of the underlying business logic, processes, and back-end resources specific to the individual business transactions**

Customer-Centric Enterprise Application Management

If the simple problems of a restaurant seem complex and difficult to diagnose, whatever plagues the smallest IT shop can be considerably more complex to resolve. And even without considering SOA, many organizations have yet to address how they might proactively meet customer expectations regarding their existing distributed Web applications. Critical Web applications can profoundly impact customer satisfaction and relationships and yet every server supporting these applications might only appear to be running properly from a systems management perspective.

In the restaurant example, rather than wait for customers to start complaining, it would have been better to detect and understand that a problem was brewing, ideally when the waiter first experienced confusion and delay attempting to process customer payments. IT also needs to service its customers proactively rather than wait for service levels at the customer level to change. When service levels start to change so the customer can measure them, there may be very little time left to diagnose and correct the problem. Ultimately the customer experience is the only thing that really matters so you can't afford to be blind to that experience. Being blind to any level of the environment increases the risk that the deteriorated service levels will become visible and be considered a problem by both the customers and the service owners such as the line-of-business.

Again, this does not negate the importance of understanding the overall health of your IT infrastructure using systems management products. But it's common enough in complex distributed Web applications for the overall application server environment to be healthy, while a problematic piece of business logic, or a back-end system supporting that business logic, may behave inappropriately in specific transactions. For example, under certain circumstances the business logic may make a disproportionate number of calls to a database for certain transactions, just as our waiter followed inappropriate procedures when processing credit cards but not cash. How would you know this problem-in-the-making existed before your customers start phoning your help desk? How would you identify the offending components involved?

Thus one needs an additional type of management whenever applications are distributed rather than monolithic regardless of what kinds of management software is already in place. This kind of customer-oriented real-time business transaction-aware type of management is what I call customer-centric enterprise application management. To be effective, customer-centric enterprise application management must go beyond other forms of management by focusing on fine-grained, component-level knowledge of the underlying business logic, processes, and back-end resources specific to the individual business transactions as they flow through the distributed application. Furthermore, customer-centric enterprise application management must be able to correlate this fine-grained knowledge of individual transactions as they travel through the IT infrastructure with the actual customer experience. In short, we need to know if the "IT soup" is too cold. With customer-centric enterprise application management, we can understand why this is so.

Customer-Centric Enterprise SOA Management

SOA and the distributed applications based on SOA increasingly tend to be larger in scale than conventional Web applications, since they grow to incorporate and bridge formerly separate application silos and enterprise domains. So if you can't mitigate SLA violations committed by your existing Web applications, a SOA is unlikely to improve the situation. In fact, the task is more daunting with SOAs. SOAs tend to be based on more heterogeneous platforms and services that are more loosely coupled, with more subtle and indirect inter-service dependencies. More things can go wrong and it can be harder to understand why. But the approach required to meet SLAs and locate and understand problematic components is nearly the same as in the customer-centric enterprise application management used for distributed Web applications.

Effective customer-centric enterprise SOA management merely requires additional tooling on top of the customer-centric enterprise application management that extends the monitoring of the customer experience from a human on a browser using a Web application to monitoring the messages sent by the software playing the customer role and using the business service. This applies to all the services in the SOA. SOA management extensions that feed into the overall transaction awareness of customer-centric enterprise SOA management are needed to provide additional visibility for every real-time business transaction that passes through the layers of services in the SOA. The focus is the business transaction itself.

Let's look more closely at customer-centric enterprise SOA management. In most modern SOAs, high-level business services accept customer transactions. These business services might themselves transform and route customer transactions in the form of SOAP or POX (plain old XML) messages to other subordinate business services or infrastructure services. More commonly, infrastructure services such as routing and transformation might be provided by middleware such as application servers or ESBs, perhaps orchestrating a number of related business services using a BPEL engine.

Both business services and infrastructure services are most often implemented using the .NET or J2EE runtime environments that dominate the vast majority of distributed enterprise applications today. Clearly, to achieve effective management of your SOA, your customer-centric enterprise SOA management solution must have very detailed visibility into the operation of these same runtime environments since they underpin so much of the SOA's essential operation and business functionality. In fact, given the heterogeneous nature of SOAs, often made even more diverse by corporate

> **Understanding business service roles and applying an architectural view of SOA focused on the customer rather than the IT organization is an effective aid in creating an appropriately customer-focused SOA**

mergers and acquisitions, it's common and certainly wise to require support for both of the major runtime environments from a customer-centric enterprise SOA management solution.

Ultimately, the lowest-level services in a SOA will pass their business transactions to even deeper object-oriented business logic, this logic again mostly based on .NET or J2EE. Furthermore, much of the object-oriented business logic written on these platforms will generate specific transactional activity in a wide range of back-end enterprise systems such as databases, CICS, and message-oriented middleware such as IBM's MQ Series.

In the end, the real goal of IT is to drive the success of each and every business transaction. So gathering and correlating information about problematic business service transactions as they travel through the SOA and its supporting IT infrastructure of middleware and back-end enterprise systems is a difficult, but absolutely necessary task for the different technical stakeholders responsible for keeping the IT side of the business running. Specific IT departments and technical stakeholders, most often operations and application support personnel, must be able to identify (triage) the offending component when it fails in order to engage the correct domain experts to fix the problem in that particular component.

All of the different types of technical stakeholders involved need a common and consistent source of information about the many components that constitute the overall application so they can effectively work together, but each stakeholder also needs to consider that application and its transactions from his own unique perspective and technical requirements. It's also often necessary to examine the problematic transaction from the standpoint of both real-time operations and historical data. A historical view is particularly important when responding to customer experience problems after the fact or to determine patterns when problems appear intermittently.

You can get a feeling for how this works by considering Wily Technologies' Customer Experience Manager (CEM) and Introscope customer-centric enterprise solutions. These solutions help IT stakeholders identify and fix components that are affecting business transactions in both Web applications and SOAs. They eliminate the usual finger pointing between the different technical stakeholders by providing a common and consistent source of historical and real-time information about the components and the transactions that flow through them. At the same time, they enable each stakeholder at every level to see the information that makes sense for his role. For example, an operations person might simply see a traffic light icon indicating a red alert when a particular customer business service starts to behave erratically before the customers notice the problem. At the same time, an application support person would see that the problem occurs in a specific Java component such as an EJB and only in certain types of transactions or perhaps only transactions from a particular customer. Application support could then notify the programmer responsible for that component who would get the critical information at the individual object method level. Similarly, if the problem was database-related, another domain expert such as a DBA could get the specific SQL statement that caused the problem in that transaction from the same set of tools, and so on.

Notice that this customer-centric enterprise approach to application and SOA management draws no particular distinction between the components that we call services and the ones we call objects or back-end systems. The perspective is always customer- and transaction-centric as it should be. In fact, since the customer is the focus of this approach, it's quite easy for IT to measure and assign dollar values to every type of transaction to demonstrate and measure the value that IT provides by supporting that transaction along with tracing the path of the transaction through all the supporting systems. This kind of comprehensive measurement and understanding can only be achieved by monitoring business transactions from the customer through the middleware hosting the business logic to the back-end systems and then back to the customer.

Conclusion

Understanding business service roles and applying an architectural view of SOA focused on the customer rather than the IT organization is an effective aid in creating an appropriately customer-focused SOA. Such a focus on the customer demands an equal focus on SLAs as an essential consideration for SOA. Enterprise systems management, SOA services management, customer-centric enterprise application management, and customer-centric enterprise SOA management are all required and must work together for you to meet customer expectations and remediate problems proactively.

Without a consistent and unified view of real-world business transactions as they travel through the entirety of your SOA including various back-end systems, object-oriented business logic, business services, their supporting .NET and Java-centric middleware, as well as other components such as messaging middleware and databases, it's not possible to understand and remediate problems that inevitably occur in any complex distributed system – SOA being no exception. By taking the correct customer-centric approach to management, you may be able to avoid Groucho Marx's dilemma. Your SOA may not seem overwhelmingly complicated anymore. In fact, it may be possible to avoid asking a five-year-old for assistance. ■

About the Author

Paul Lipton is a senior architect specializing in SOA and Web services in the Wily Technology Division of CA. He has served in CA s Office of the CTO as a strategist, and has been an architect and developer of enterprise systems for over 20 years. Paul has also participated in many standards organizations, is a founding member of the CA Council for Technical Excellence, a Microsoft MVP, and a Sun Java Champion. He is a highly sought-after author and speaker, and has shared his knowledge around the world covering such topics as Enterprise Architecture, SOA, Web services, management/security, governance, Java, .NET, and EAI.

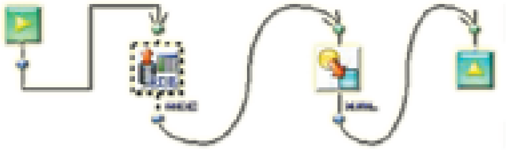
paul.lipton@ca.com

Achieve Point-and-Click SOA™

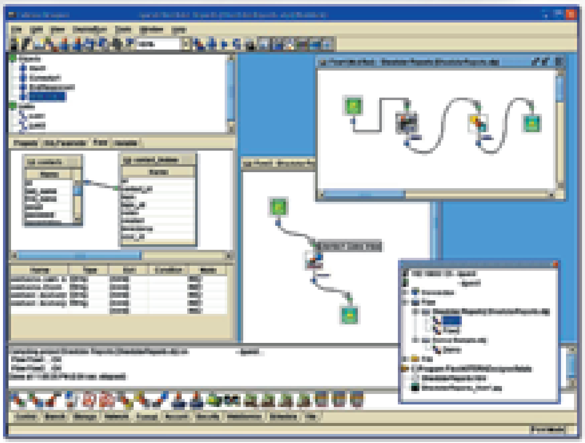
exteria™
Business Automation Platform
Get on the SOA fast track!
NEXT STOP: extentech.com for
FREE Exteria Downloads
Demos and Whitepapers

High-Speed Productivity

Exteria is the ideal modeling tool and web services platform for rapid SOA implementations.



Model data flows graphically in the code-free designer then publish with a click to XLS, PDF, SOAP, XML, and more -- all without writing any code!



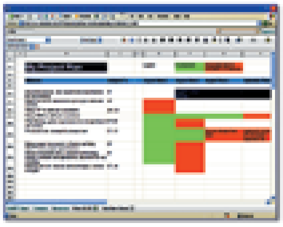
The Exteria data flow modeler features dozens of connectors including LDAP, SOAP & XML

Spreadsheet Automation

SOA is all about re-use. There is no need to reinvent business logic already available in spreadsheets. Leverage this asset by mapping data flows to formula cells and publishing them as RSS feeds, SOAP web services, or to an AJAX web spreadsheet.

works with:

sheetster



Integration with Sheetster.com web spreadsheets enables realtime reporting and collaboration.

includes:

EXTENXLS 5 JAVA SPREADSHEET TOOLKIT

Leverage the power of Java Spreadsheet Automation with the best ExtenXLS ever!

extenTECH™
exceed expectations

sales@extentech.com
call 415.759.5292
http://extentech.com/itsg

Web Services-based Inter-organizational Architectural Framework for B2B Marketplaces

Catalyzing inter-organizational collaboration and transactions

WRITTEN BY **JAI GANESH AND SRINIVAS PADMANABHUNI**

B2B marketplaces facilitate efficient search and transactions by offering services such as buyer/supplier and product/services searches and transactions such as procurement and asset disposal.

Besides their market making functions, marketplaces also offer integration services such as supply chain and ERP integration and have allied with various firms to offer value-added services such as vendor ratings, logistics, and payment processing. B2B marketplaces differ from traditional marketplaces in offering increased personalization and customization of products and aggregating and disaggregating information-based product components to match customer needs. They can overcome some of the problems related to richness versus the reach of information because they can facilitate real-time transactions. They also enable new kinds of price discovery to be employed in different markets. B2B marketplaces improve information sharing between buyers and sellers, helping lower the cost of logistics and promoting quick just-in-time deliveries and reduced inventories.

B2B marketplaces can be broadly classified as horizontal or vertical, buy-side or sell-side. Here we classify them as public/independent, private, and consortia.

Public Marketplaces

A public marketplace, also known as a neutral marketplace or third-party marketplace, brings together buyers and sellers in a particular industry for purposes of commerce. It provides content, value-added services, and transaction capabilities.

Private Marketplace

Private marketplaces are owned and operated by one company to do transactions with a select group of suppliers. They have the potential to provide high performance and tight integration with current suppliers, a facility provided by the traditional electronic data interchange (EDI) model. Examples of enterprises that have embraced this model include: Wal-Mart, Dell, Sun Microsystems, Amtrak, and Cisco.

Consortia Marketplace

Consortia marketplaces are jointly owned by several large enterprises that deploy applications and infrastructure to facilitate collaboration and conduct business among trading partners. They are highly customized and integrated with the process of its founders and hence require a large investment and have longer implementation schedules. An example would be E2Open.

Inter-Organizational Commerce Using Web Services

Inter-organizational information systems (IOIS) facilitate technology-based cooperation across organizations such as interactive product improvement, forecasting, and inventory control in addition to offering improved integration, collaboration, and access to business intelligence. B2B marketplaces offer dynamic forward and reverse auctions, payment services, valuation services, e-procurement services, industry news, market analysis and reports, virtual private marketplaces, and indigenously developed procurement software licenses. The B2B marketplaces have allied with partner firms to offer value-added services (VAS). Value-added services are defined as services that supplement the actual transaction, cataloguing, and search capability. These include financial services, logistics services, analytics, inspection, and settling disputes.

Integrating their products and services with their customers' assured B2B marketplaces of regular revenues in the form of license fees. Integration with corporate systems is achieved through enterprise application integration methods. Typically a common meta-data repository maintains process and data maps for translating the data, and a message broker routes the data to the right application. Integration also involves modeling, automating, and integrating business processes and trading relationships between partners. Integration of information refers to sharing information among participants of the B2B marketplace such as buyers and suppliers. This includes data such as inventory data, demand data, capacity plans, production schedules, promotion plans, and shipment schedules. B2B marketplaces at this stage also facilitate collaborative activities such as joint design and execution plans for product introduction, demand forecasting, and replenishment. They also facilitate workflow coordination, which refers to streamlined and automated workflow activities between supply chain partners.

To facilitate integration and collaboration B2B marketplaces are building their technological capabilities such as data mapping repositories including XML (Extended Markup Language) document exchange formats and trading partner agreements that let customers switch from one market to the other at any time. This is important since the customer may not find relevant buyers or sellers on one particular marketplace. Web Services show promise in enabling B2B marketplaces to offer inter-organizational commerce by offering open standards-based interfaces. Web Services have standardized interfaces that enable seamless interoperability among heterogeneous IT implementations. Web Services are becoming the technology of choice today for solving pressing point-to-point integration needs because of the inherent cost benefits of the approach. Interoperability is the most important capability of Web Services and has the potential to reduce IT integration costs in and across enterprises. A Web Services-based SOA reduces the redundancy by offering a high level of reuse. A further requirement relating to investment in conventional legacy infrastructure is key to justifying any architecture. SOA preserves investment in legacy by providing the endpoints for the legacy functionalities to be accessed by other applications across any channel.

Now we'll explore the viability and potential of Web Services with regard to inter-organizational commerce related to the key services offered by B2B marketplaces such as auctions, reverse auctions, catalog-based e-procurement, private marketplace services, third-party transaction services (payment), and third-party data services (valuation and industry news and reports).

Forward & Reverse Auctions

Forward and reverse auctions are the key service offered by B2B marketplaces. The various auction formats include the English auction, the Dutch auction, and the Vickrey auction. Web Services would provide an open standard way for buyers and sellers to connect to auction service interfaces. The usefulness of Web Services is pronounced when the service offerings of B2B marketplaces are orchestrated by linking up with their other services as well as linking with third-party service providers.

Catalog-Based E-Procurement

Catalogs are fixed-price catalogs that create value for buyers or sellers by aggregating demand or supply into a meta-catalog and negotiating volume discounts (up to 20%). For buyers, B2B marketplaces provide a gateway to thousands of suppliers and service providers across a spectrum of industries. This reduces the search costs of buyers and helps improve supply chain efficiencies by identifying suppliers that can match their expectations in the price-quality-delivery equation. So depending on the sourcing needs of the buyer they can post their requirements on the site so that suppliers can get in touch with them. Web Services could be adopted to integrate the buyers/sellers with the catalog offered by the marketplace resulting in more efficient and flexible transactions.

Third-Party Transaction Services

Third-party transaction mechanisms come into play to send, execute, and settle orders that have been agreed to in the marketplace. Payment services fall under this category. These rely on the features of the communication infrastructure, either to transfer a payment physically or transmit information about credits and debits to the accounts of buyers and sellers. Web Services would facilitate easy interoperability between the systems of the B2B marketplace with those of the payment service providers.

Third-Party Data Services

This includes data services such as rating and valuation services. Examples include services offered by the global rating agency Dun and Bradstreet such as DUNS numbers and Indicative Risk Scores. Other services include international quality control, inspection, testing and verification, vendor assessment services, quality and quantity certification, and valuation services. To ensure competitiveness in catering to a geographically dispersed market, it has become essential to build the confidence about the quality of the offering and the underlying process among players in the marketplace. Independent specialized agencies play a pivotal role in promoting this through services such as inspection, surveying, analytical testing, accreditation and certification, and consultancy. Other content services include content sourcing from multiple vendors such as Gartner and Forrester Research.

Third-Party Distribution Services

This includes services such as logistics services offered in association with logistics service providers. Logistics services provide the link between electronic business transaction and effecting order fulfillment. The B2B marketplace provides a medium for interaction between logistics players and sellers and buyers. This is aimed at facilitating the easy movement of equipment and other assets transacted through the marketplace.

Architectural Imperatives for B2B Marketplaces

Architectural requirements for an online auction site include application functionality, interoperability, performance, and scalability, flexibility, and manageability. The physical architecture consists of databases, servers, software applications, Web servers, and networking and Application Protocol Interfaces (APIs) to link them all together. Due to the risks of system failure, redundancy at the database and server levels is necessary. Software applications must be able to support multiple auction models (single seller-to-many buyers, single buyer-to-many sellers, multiple buyers and sellers) and auction types like Dutch and Vickrey auctions. They should also be able to support special rules such as role-based prioritization, where the best customers get priority. Further, a key requirement is to be able to interoperate with the multiple data representation formats of the buyers and sellers as well as partners. Overall, a flexible and interoperable architecture is needed to keep up with the pace of change in the industry.

One of the key components to online auction applications is the ability to manage and report dynamic information. The application runs in conjunction with a database that stores all of the information and metadata necessary for each auction transaction. The database contains data like bid and bidder information, rules for the auction, and information about the item for sale alongside metadata about the bidders and sellers and the items for sale. The application should provide mechanisms for representing items and services from both buyers and sellers in a manner so that matching buyers and sellers is seamless. The applications gather, store, and then access the matched information in real-time and display pertinent information in the GUI. The application must be able to handle hundreds of concurrent bids a second and instantly post the results. The marketplace architecture should support appropriate open standards-based messaging infrastructure besides providing a registry for repository of the buyer, seller, and other partner information. The architecture must support a flexible way of incorporating workflows or orchestration to enable typical transactions.

The architecture at a successful online auction concern should streamline client services including e-mail notifications, searches, automatic bidding (including pre-set bid thresholds), customization capabilities, and auction management features for sellers. At a minimum clients also expect customer service, security, fraud prevention, and ancillary features like online payment and logistics services. All of these services and applications must be part of the application architecture or integrated with the auction application via standard APIs over standard messaging standards. Because of the number of features and functionality a B2B marketplace must possess to be competitive, it should be able to integrate new functionality, applications, and partners continuously. Interoperability needs to be quick and seamless.

A Proposed Service-Oriented Architecture

In this section, we'll describe a typical transaction scenario for the three types of B2B marketplaces, public, private, and consortia and then describe the proposed Web Services-based architecture to address this scenario. The key requirements of the IT architecture supporting a B2B marketplace are:

- Ability to integrate with buyer and seller systems
- Ability to integrate smoothly with other partners (e.g., credit card firms, courier companies, etc.)
- Ability to configure/reconfigure services
- Ability to match the information of the participants in a transaction

Given the nature of the scenario and the number of stakeholders involved (e.g., the B2B marketplace, sellers, buyers, credit card companies, courier companies, etc.), a Web Services-based Service Oriented Architecture would be the ideal solution to the problem. Due to the loosely coupled nature of Web Services, the B2B marketplace doesn't have to have hardwired connections with the buyers, sellers, credit card companies, or logistics service providers. This lets the B2B marketplace have access to more services, offering more options to its customers. This wouldn't only address current needs but would also address future needs when the B2B marketplace may be needed to make fast business connections with partners without going through the conventional pattern of making large-scale changes to the system. Web Services would enable the B2B marketplace to isolate the business logic from integration. Most conventional integration solutions embed part of the business logic in the integration layer thereby requiring considerable effort in making modifications. Web Services address the key requirements of this scenario. Based on open standards like XML and SOAP, they define a means by which the services of the B2B marketplaces and their partners can be published, discovered, and invoked.

A Web Services-based Service Oriented Architecture for public B2B marketplaces is shown in Figure 1. At its core a marketplace architecture should support the appropriate messaging infrastructure for seamless connectivity with buyers, sellers, and partners. This may rely on industry standards like SOAP or ebMS (ebXML messaging) to provide universal messaging interoperability.

A marketplace also needs an appropriate registry to maintain the list of buyer and seller services and items. This kind of registry should be open to partners, buyers, and sellers via the messaging infrastructure and should be capable of registering and de-registering services, items, and documents. Ideally this can be a UDDI registry or an ebXML registry.

The registry is mandated to provide a matching infrastructure between buyers and sellers of the exchange. This matching infrastructure has to be in multiple tiers for reasons that will be explained a little later. Over and above the infrastructure layers of messaging, registry, and matching infrastructure, there's a need to provide a range of horizontal services to the buyers, sellers, and partners. These services are enabled by proving a B2B service management layer, which provides the point of interaction with the registry of the exchange as well as with the buyers, sellers, and partners. A key function of the B2B service management platform is to provide

REGISTER TODAY AND $AVE!

Rich Internet Applications: AJAX, Flash, Web 2.0 and Beyond...

www.AjaxWorldExpo.com

AJAXWORLD™ EAST
CONFERENCE & EXPO

NEW YORK CITY

THE ROOSEVELT HOTEL LOCATED AT MADISON & 45th

SYS-CON Events is proud to announce the AjaxWorld East Conference 2007!

The world-beating Conference program will provide developers and IT managers alike with comprehensive information and insight into the biggest paradigm shift in website design, development, and deployment since the invention of the World Wide Web itself a decade ago.

The terms on everyone's lips this year include "AJAX," "Web 2.0" and "Rich Internet Applications." All of these themes play an integral role at AjaxWorld. So, anyone involved with business-critical web applications that recognize the importance of the user experience needs to attend this unique, timely conference – especially the web designers and developers building those experiences, and those who manage them.

BEING HELD MARCH 19 - 21, 2007!

We are interested in receiving original speaking proposals for this event from i-Technology professionals. Speakers will be chosen from the co-existing worlds of both commercial software and open source. Delegates will be interested in learning about a wide range of RIA topics that can help them achieve business value.

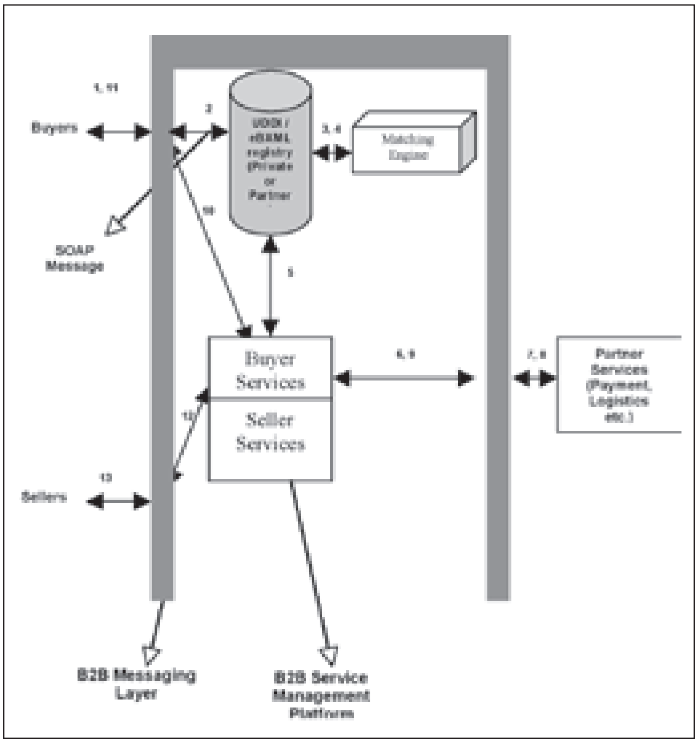

1, 11
Buyers
2
UDDI / ebXML registry (Private or Partner
3, 4
Matching Engine
10
SOAP Message
5
Buyer Services
Seller Services
6, 9
7, 8
Partner Services (Payment, Logistics etc.)
12
13
Sellers
B2B Messaging Layer
B2B Service Management Platform

Figure 1: Web Services-based Service Oriented Architecture for public B2B marketplaces

lifecycle management for the different services provided by the exchange. In particular, the platform performs key functions related to security including authentication, access control, and defenses against DOS attacks. It is also responsible for the key tasks of registry management, messaging, semantic matching, and process orchestration.

1) **Service Lifecycle Management:** The main function of a B2B service management platform is managing the lifecycle of the infrastructure as well functional services offered by the exchange, including auction-related services, listing-related services, etc.
2) **Security:** The platform accepts standard credentials from the service requestor. If a seller wants to access a particular Web Service on offer (say an auction service) the request is passed on as a SOAP message across the service gateway. The validity of the request is checked against the customer's login ID and password.

   Due to the ease of availability of crucial functionalities as services, there's an increased possibility of repeated or malicious invocations of these services. The platform performs the function of SOAP content inspection enabling it to detect repeated attacks or malicious content sent as part of SOAP requests. The platform also ensures that all access control rules and policies internal to the organization are followed in service invocations
3) **Semantic Matching:** The management platform has to provide the appropriate matching infrastructure to interact with the registry as well to enable transactions. This infrastructure can be multi-tiered depending on the complexity of the transaction. The bottom tier of the matching infrastructure has to include data item-level interoperability that can be ensured by adopting industry standard classifications like UNSPC or domain-specific vocabularies like ACORD for insurance, or may rely on horizontal standards like ebXML core components, or in highly specific case there could be an ontology specific to the transaction.

   Above the data item-level interoperability there has to be the document-level interoperability that ensures that both sides of a transaction understand the documents made up of the mutually understood data items. This again can be based on horizontal standards like UBL, vertical-specific vocabularies like ACORD, or specific ontologies pertaining to the transaction.

   At the top is the functionality-cum-service-level interoperability where both participants in the transaction understand the capabilities of the other by getting to understand the functional details of the services offered on the auctions. This kind of capability could be described in standards like WSDL or ebXML CPP. However, the matching process may involve some kind of exchange between the participants leading to a shared understanding of the services or functionality.
4) **Registry management:** This is the component that interacts with the registry and with the matching engine.
5) **Process Orchestration:** Some marketplace processes involving coarse-grained functionality require the orchestration of multiple partner services. For example, a buy request coming in from the customer would first invoke the Catalog listing service and then other partner services such as Payment, Logistics, and Delivery Tracking.
6) **Transaction Management:** It's responsible for maintaining transactions across multiple invocations.

The Workings of the Architecture

Now we'll explain the workings of the proposed architecture for a public B2B marketplace (Figure 1) using a typical buyer scenario. In this scenario, the buyer intends to buy 1,000 batteries (UNSPC Classification No. 26-11-17-09) through the B2B marketplace by placing a listing in the buyer *Catalog*. The buyer wants to invoke the *Reverse Auction* service as well as the partner services such as *Payment* and *Logistics*, which are offered by third-party vendors. In Step (1) the buyer sends a request to the B2B marketplace systems. This request is verified at the marketplace layer and transmitted to the UDDI/ebXML registry to check for the availability of the Reverse Auction service in Step (2). In Step (3) the service integration bus transmits this service request to the matching engine and gets a response in Step (4). In Step (5) the UDDI/ebXML registry contacts the *Buyer* service and invokes the *Catalog* listing service followed by the *Reverse Auction* service. In Step (6) the *Buyer* service application sends a request to the B2B marketplace layer, which in turn directs the request to the third-party *Payment* and *Logistics* services in Step (7) that are offered by the partners. Steps (8) and (9) capture the responses from the third-party services as well as the B2B marketplace layer. Step (10) depicts the response from the Buyer services to the B2B marketplace layer and Step (11) depicts the response to the buyer about the results of the auction. In Steps (12) and (13) the seller of the batteries is notified of the sale. In this scenario the B2B marketplace layer acts as a key intermediary, redirecting the queries from various applications and services to and from the external UDDI. On implementing the Web Services-based SOA, the B2B marketplace acts as the service provider by offering services that are self-contained, self-describing, and modular. In most cases the B2B marketplace's services either have to be invoked by customers or it may have to interact with the services of its partners (e.g. *Payment* service).

Conclusion

The technology adoption trend is shifting to a scenario where the customers of a B2B marketplace need to interoperate not only with

— continued on page 25

Automation: A New Imperative for SOA Application Success

Figuring out what to manage

WRITTEN BY **CHRIS FARRELL**

For several years, software vendors have been creating development and infrastructure products for the latest IT architecture style – Service Oriented Architecture (SOA). Recognizing the immense value SOA can bring to IT, companies like BEA, IBM, and Microsoft have delivered products to help customers design and build SOA-based applications. These vendor-driven initiatives are beginning to sprout actual customer-developed applications that are built on the promise of SOA, such as better flexibility, agility and reuse.

In fact, enterprises of all sizes are increasingly using SOA concepts and technologies to create a new generation of mission-critical business applications, ranging from "simple" portals to incredibly complex and feature-rich integration applications. They're taking this approach in an effort to improve IT's ability to deliver business applications to the right employees, partners, and customers on-demand.

The reason? SOA applications have the potential to turn around many of the problems plaguing IT organizations today, including lack of interoperability, schedule delays, budget overruns, and an inability to respond quickly to "the business." SOA also provides easy tactical paths to create economies of scale for mergers, acquisitions, spinouts, spin-ins, and other unique organizational relationships.

Chances are you'll eventually be asked about your company's SOA strategy, if you haven't been asked already. The actual types of SOA implementations run the gamut of possibilities and should evolve in line with your organization's unique migration path for SOA. You could be building SOA applications already. Maybe you're in the early stages of selecting the specific applications, third-party service vendors, and/or products you'll need to build your first application. Or, you might already be leveraging a SOA environment and related technologies to create new applications.

Regardless of your particular situation, SOA has matured into more than just an architectural theory. It's a philosophy and an attitude that's revolutionizing the way business and IT work together. No matter where your organization stands today, one thing is for sure: if you're in IT, SOA environments will be part of your life.

SOA's main premise revolves around how application functions are designed, built, and deployed as they relate to the business processes they're supposed to serve. Each business application is broken down into discrete services. Rather than build a single application that contains all the necessary services, each particular service is built and deployed on its own, allowing massive reuse and flexibility as business needs expand and change. This allows the business applications themselves to remain relatively stable, while individual services can (and usually do) change as often as necessary to deal with business problems (such as acquiring a new division), and technology problems like adopting a new Identity Management model. Another benefit of SOA applications is that some of the services can be "bought" from third parties, either for internal deployment or on-demand runtime requests, rather than building each and every piece of the application.

So SOA isn't just vendor-neutral. It's also platform-neutral. To borrow a programming concept, an SOA environment is essentially an object-oriented design for business processes. Each service behaves like a "black box." The platform, technology and programming inside each particular service are hidden from service requesters (i.e., the rest of the application(s)). Black box engineering allows the greatest flexibility when designing large projects across multiple organizations. But black box application development and deployment always come at a price – the inability to monitor and manage application performance easily. Some experts have compared managing black box technologies to seeing inside a black hole.

Like middleware projects a decade ago, application performance management (APM) is a primary factor slowing SOA deployments down. So how can the industry move forward with its management solutions and realize the full potential of SOA?

Application Management History

The biggest problem with any monitoring project, from network or business performance to applications, is determining what to monitor. In application management, the issue of knowing what to monitor is harder because of the multiple levels of measurements. The business processes and transactions are the reason for an application's existence, but knowing them alone isn't enough. To achieve effective application management, it's critical to correlate the internal application components to the overlying processes and transactions. To understand the logistics issues associated with this, a look at APM history is relevant.

As enterprises first used Web technologies to make data and transactions available to end customers (or employees), the applications were fairly simple – accessing a piece of data or requesting approval for a credit card – and then presenting the resulting data to the customer through a browser. A complex process was simply made up of multiple individual Web pages, each one running a back-end application that performed a discrete function.

These applications were easy to monitor simply by exercising each Web page that made up a step with a synthetic user.

As the benefits of Web applications permeated businesses, the application server became the common tool. All the steps of the business logic could now be put into a single application, instead of creating one application for each page request or process step. The applications acted as integration hubs – bringing together the required back-end data and transaction requests into one place – and delivering the consolidated results to the requester.

This is where things got tricky. Java and Java Virtual Machines (JVMs) made it difficult to see what was happening at the point of integration. A set of solutions sprung up reporting on proprietary data inside the JVM, providing detailed information about the point of contact. While the ability to see that data was critical, the data itself proved to be both a commodity and a burden.

Having the data didn't solve the monitoring "problem" of deciding what to monitor. This became a larger issue as enterprises started dealing with more complex applications, quicker turns of application versions and the introduction of new technologies – such as portals – built around a Service Oriented Architecture.
Implementations typically required weeks of reverse engineering and involved professional services consultants, senior developers, and architects. The process du jour was Trial and Error, hoping to determine what application metrics correlated to the overlying processes – while simultaneously deciding on the right trade-off of having the appropriate data for triage versus keeping application overhead low. When another application came along, or a change occurred to the managed app, the exercise was repeated. While not efficient, the process could work for stable hub-like integration applications.

Recently, applications have become even more distributed – first with the introduction of integration platforms and now with increasing rollouts of Service Oriented Architecture application environments. The manual correlation exercise followed for the hub applications is no longer practical. In fact, managing SOA apps this way is literally impossible. There are too many integration

Architecture

— *continued from page 22*

the B2B marketplace but with third-party service providers. So the B2B marketplace must be able to offer a single flexible point source for inter-organizational commerce. Web Services-based SOA shows promise in enabling inter-organizational commerce at a reduced cost by leveraging the existing investment in legacy systems and thereby increasing the efficiency of these enterprises. Web Services increase flexibility by offering the chance of creating flexible new business processes out of the existing IT infrastructure. To that end, we outlined a reference architecture that leverages Web Services and handles all the functionalities of a B2B marketplace. The architecture described provides core service orientation to the B2B marketplace. It can be considered a baseline architecture for all B2B marketplace enablements. The architecture enables truly flexible integration among the internal systems of the B2B marketplace and its customers along with those of its partners using Web Services. It also enables existing legacy applications to take part in the overall business architecture. Integration with the systems of the B2B marketplace' partners is also eased.

Web Services allows the internal and external stakeholders of B2B marketplaces to protect existing IT investments and make incremental investments over existing systems. The following are some of the benefits for B2B marketplaces in implementing a Web Services-based architecture:

1. Web Services-based IT architecture would let the B2B marketplace offer its applications as services across its customers and partners providing increased flexibility and reuse
2. Web Services would enable the B2B marketplace systems to be compatible with other internal systems without considerable effort interfacing
3. A Web Services-based architecture would integrate the disparate systems so the B2B marketplace could extract data from various databases, applications, and processes. This would empower the B2B marketplace to make decisions based on the transaction trends and customer preferences and also have better planning.
4. Web Services-based IT architecture would allow the B2B marketplace to service the requests of its customers regardless of the channels through which they want to access the service (e.g., a PDA) and thereby improve customer loyalty. 🟥

About the Authors

Jai Ganesh is a senior research associate at the SOA/Web Services Centre of Excellence in SETLabs, Infosys Technologies. He got his PhD in information systems from the Indian Institute of Management Bangalore (IIMB) and also has an MBA degree in corporate strategy and marketing. His research focuses on Web Services, SOA, Web 2.0, IT standards, IT strategy, adaptive enterprises and hyper-competitive businesses. His research has been published in journals such as Information and Management, Journal of Global Information Management, and International Journal of Retail and Distribution Management and at conferences such as ICW and AMCIS. He is a reviewer for a number of peer-reviewed journals and conferences and has consulted for many software firms. Jai has also filed for three patents in the areas of Web Services and the Common Information Model and is currently writing a book on innovation in the Indian context.
jai_ganesh01@infosys.com

Srinivas Padmanabhuni is a principal researcher with the SOA/Web Services Centre of Excellence in SETLabs, Infosys Technologies, and specializes in Web Services, Service Oriented Architecture, and grid technologies alongside interests in Semantic Web, intelligent agents, and enterprise architecture. He has written several papers for international conferences. Srinivas has a PhD degree in computing science from the University of Alberta in Edmonton, Canada.
srinivas_p@infosys.com

points, transaction types, and relationships – the decision points revolve around millions of entities.

Let's look at an example in the service industry. A clearing house application that connects auto insurance companies with service providers (and vice versa) has a complex application running two different access applications on two different application server platforms, all connected by a set of more than 60 individual services. Each service runs as a separate application. While efficient in operational logistics, managing so many moving parts each its own complete system is a difficult proposition.

As a result, application management solutions have to catch up. The fact that they haven't has created a gating factor to widespread rollout of SOA applications. Because manual correlation is no longer possible, the only way to deal effectively with the problem is through automation – and the most effective way to insert automation into technology is through modeling.

Why Automation Through Modeling?

Automation solves several issues associated with SOA performance management, especially the age-old issue of figuring out what to actually monitor. The steps of APM can be described in three stages:

1) **Setup/Installation**
- Installation and configuration of monitoring agents
- Creation of meaningful UI visualization

2) **Analysis/Reporting**
- Correlation of business processes to internal application components
- Root cause analysis

3) **Change Management**
- Configuration of monitoring agents
- Correlation of application details to business processes

The best way to automate these processes is through "modeling," which lets companies leverage technology instead of hundreds of human hours to make the proper decisions in the steps listed above.

Modeling in setup and configuration lets the enterprise deploy managed applications faster and easier. Management tools with modeling have a programmatic representation of the application architecture, allowing the tool to find the correlation between the application components and the business processes. This, in turn, lets the tool intelligently select which pieces of the application to monitor.

Armed with both the correlation to business processes and the architecture, a modeling-based tool can also automatically create useful user interfaces around service level reporting, performance review, and root cause analysis.

The analysis step of APM has also traditionally been a manual exercise, especially in the data-centric tools focused on application servers. The typical steps these tool owners take when a problem occurs involve leveraging an expert (senior developer or architect) to interpret the mass of data the tool provides, hoping that the helpful data is actually available (i.e., it wasn't missed in the manual correlation process).

Modeling helps automate problem triage by starting with the architectural representation or application schematic. This allows quick diagnosis down the architectural path of a business transaction to isolate the root cause. An additional benefit is that once a problematic component is found, the application schematics let the triager determine all the impacted business processes, whether the application owners know about the problems or not.

Change management is an important issue in any application environment. In SOA, it's critical. Take the clearing house application example discussed above. With over 60 discrete applications, even if only one change is allowed per year for each service, that's at least one change a week. That's the minimum change to deal with. In reality, changes could be occurring daily. So manually adjusting to changes is not only inefficient, it's simply not possible in a scaled business environment.

Modeling lets change be dealt with by an APM solution automatically. First, in detecting that a change has occurred. Then in adjusting the correlation between the processes and the application components, deciding what to monitor – and delivering a new application schematic when necessary. Only through automation can APM tools catch up with the rest of SOA technology – and allow for complete use in production environments.

Looking Ahead

Service Oriented Architecture is a powerful concept. It allows IT organizations and their related business organizations to work more efficiently. SOA applications will be more prevalent and managing them will remain critical. There are specific steps organizations can take to make SOA management work for them:

1) **Make management a priority:** Management shouldn't be an afterthought. Make sure that it's budgeted for when pulling together SOA projects – both in time and money.
2) **Make a new decision:** Managing SOA is a new problem, one that deserves true evaluation. You might decide to use one of your existing APM vendors, but don't just write them in. SOA presents unique challenges that existing tools find difficult to solve.
3) **Start small:** SOA is a change in philosophy that creates barriers to success. Don't add more by trying to run multiple applications out of the gate. Like other technology breakthroughs, your chances of success increase when dealing with a single problem.
4) **Assume nothing:** Don't assume that what you deploy is architected and coded like you designed it. Get runtime information to help determine what the real architecture looks like.
5) **Communicate, communicate, communicate!** Involve all of the key stakeholders and often. It's important for executives to see the value in the project, for line-of-business managers to see that their applications will run better, for IT management to see efficient operations, and for developers to see innovation. Keep them all in the loop from start to finish.

Any new use of technology will hit barriers, and SOA management is no different. But the tools do exist to help. Once the mystique is gone, and you know what to monitor, rolling out managed applications as needed will be much easier. So design away, and see how many different services you will have in your next application. ■

About the Author

Chris Farrell is vice-president of product management at ClearApp. With 20 years of experience in technology ranging from system programming to wireless data to application management, Chris is responsible for setting product direction of ClearApp's QuickVision APM solution. Prior to ClearApp, he was a marketing director at Wily Technology, where he launched the first-ever production J2EE application monitor. His experience also covers a decade at IBM where, among other things, he was a product manager for the Wireless Internet Software Suite. With a BSE in electrical engineering and an MBA from Duke University, Chris has been evangelizing leading-edge technologies for the last 10 years.

> “Businesses that ignore the potential of SOA will find themselves outpaced by rivals who improve their agility and transform themselves into new kinds of enterprises
>
> — Yafim Natis, Gartner Analyst

2-DAY EVENT!

SOA WebServices Edge 2007

Plus

Enterprise OpenSource Conference & Expo 2007

TOPICS INCLUDE:

SOA Web Services
- AJAX and SOA
- Web 2.0
- Universal SOA
- Protecting Web Services
- Troubleshooting SOA
- Governance
- Open-Source SOA
- XBRL
- Service Virtualization

Open Source
- Open Source Business Models
- Open Source ESB
- OpenAjax Alliance
- SaaS and Open Source
- Spring, Hibernate and Eclipse
- Seam
- Open Source Penetration
- Monetizing Open Source
- Open Source Databases
- AMQP
- Open Source Middleware

June 25-26, 2007
Roosevelt Hotel / New York City

Register Online! www.soaeosconference.sys-con.com

SOAEOSCONFERENCE.SYS-CON.COM
REGISTER ONLINE TODAY
SAVE $200!
(HURRY FOR EARLY-BIRD DISCOUNT)

11th International SOA WebServices Edge 2007 conference+expo

2007 ENTERPRISE> OPENSOURCE CONFERENCE+EXPO

2007 is to many industry insiders shaping up to be a major inflection point in software development and deployment, with SOA, Web Services, Open Source, and AJAX all converging as cross-platform and cross-browser apps become the rule rather than the exception.

Accordingly the 11th International SOA Web Services Edge 2007 again seeks to offer comprehensive coverage and actionable insights to the developers, architects, IT managers, CXOs, analysts, VCs, and journalists who'll be assembling as delegates and VIP guests in The Roosevelt Hotel in downtown Manhattan, June 25-26, 2007

Co-located with the 2nd Annual Enterprise Open Source Conference & Expo, the event will deliver the #1 i-technology educational and networking opportunity of the year. These two conference programs between them will present a comprehensive view of all the development and management aspects of integrating a SOA strategy and an Open Source philosophy into your enterprise. Our organizing principle is that delegates will go away from the intense two-day program replete with why-to and how-to knowledge delivered first-hand by industry experts.

Visit soaeosconference.sys-con.com for the most up-to-the-minute information including... Keynotes, Sessions, Speakers, Sponsors, Exhibitors, Schedule, etc.

BROUGHT TO YOU BY:

SOA Web Services Journal
focuses on the business and technology of Service-Oriented Architectures and Web Services. It targets enterprise application development and management, in all its aspects.

Enerprise Open Source Magazine
EOS is the world's leading publication showcasing every aspect of profitable Open Source solutions in business and consumer contexts.

SYS-CON EVENTS For more great events visit www.EVENTS.SYS-CON.com

Exhibit and Sponsorship Info:
Call 201-802-3020 or email events@sys-con.com

So Many Frameworks...So Little Time: What's an Architect to Do?

The relationships between The Open Group Architecture Framework and the U.S. Department of Defense Architecture Framework

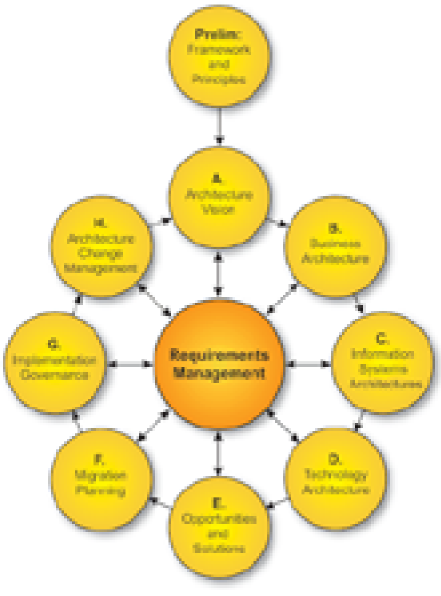
WRITTEN BY **FATMA DANDASHI, TERENCE BLEVINS, ROLF SIEGERS, AND JUDITH JONES**

Numerous architecture framework standards have been developed and have matured over the past decade. Some of these standards overlap in their focus areas, and others address completely different aspects of the architecting process. In this latter case, a natural synergy can sometimes be identified and leveraged between frameworks.

An industry working group was formed to analyze and document the relationships between The Open Group Architecture Framework and the U.S. Department of Defense Architecture Framework, identifying complementary areas between these two standards. This article is a brief summary of that initiative's detailed 50-page white paper, "The Open Group Architecture Framework and the U.S. Department of Defense Architecture Framework."

The Open Group Architecture Framework (TOGAF) is primarily focused on architecting methodology — i.e, the "how to" aspect of architecture without prescribing description constructs to document the architecture. The U.S. Department of Defense Architecture Framework (DoDAF) is focused on architecture description via a set of views without specifying any methodology. The complementary aspects of these two frameworks lead to the question: "Can architects benefit from leveraging both TOGAF and DoDAF together?" Two years ago, this is what a working group that included representatives from MITRE, Raytheon, and Architecting-the-Enterprise — supported by members of The Open Group Architecture Forum — set out to study. Core members of the group included individuals from the official DoDAF and TOGAF standards working groups. The baseline versions of the documents used were TOGAF Version 8.1 and DoDAF Version 1.0. Upcoming revisions of both these standards will heighten their support of architecting service-oriented systems, and both are valuable enablers to architects and architecture teams.

Below are some of the group's key findings, as well as insights into how architects can use these frameworks to align their business objectives better with IT infrastructure and systems.


Prelim: Framework and Principles
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

Architecture Methodology: The Open Group Architecture Framework

The Open Group first developed the TOGAF Architecture Development Method (ADM) in 1995, baselining it from the Technical Architecture Framework for Information Management (TAFIM), a series of architecture guidance documents provided to them by the U.S. Department of Defense. The DOD spent millions of dollars and several years evolving its TAFIM before turning it over to the Open Group for its ongoing maturation and dissemination across government and industry. TAFIM was subsequently retired.

The ADM is a prescriptive, step-by-step instruction guide for "how to" architect. It's presented in a series of phases that guide the architect or architecture team through the architecting lifecycle of system development. The first seven releases of TOGAF ADM (1995-2001) were focused on providing technical architecting guidance. The 2002 release of TOGAF 8.0 extended this earlier technical focus into four areas: business, data, applications, and technology architectures. This "collection" of architectures is commonly known as enterprise architecture — the interrelation and integration of business and technology. This same business and technology interrelation and integration are at the heart of service-oriented architecture design and implementation.

Architecture Description: U.S. Department of Defense Architecture Framework (DoDAF)

The primary focus of the DoDAF is architecture description — it prescribes a specific set of models that illustrate the architecture of concern. The framework defines 26 products (see Table 1) that reflect three different architectural viewpoints: operational, systems, and technical standards. DoDAF was developed to support interoperability between systems whose architectures are described with this framework. It's easier to determine how to integrate systems when they are modeled in a "common language" so that system interfaces, data formats and exchanges, implemented standards, etc. can be analyzed with the operational and system behaviors and structure.

DoDAF has formed the basis for several other frameworks such the U.K.'s Ministry of Defense Architecture Framework (MODAF) and the soon-to-be-published Standardization Agreement (STANAG) NATO Architecture Framework. DoDAF is comprised of two volumes: "Definitions & Guidelines" and "Product Descriptions."

A supplemental DoDAF Deskbook was also published to provide guidance to DoDAF users. This Deskbook consolidates supporting information such as white papers, case studies, discussion on the Core Architecture Data Model (CADM), architecture tools, Universal Reference Resources (URRs), and the Federal Enterprise Architecture (FEA) reference models.

Results Summary

The TOGAF/DoDAF working group validated its original hypothesis that there is synergy across a number of areas between these two frameworks, where DoDAF views can be used throughout the steps of the TOGAF ADM phases to develop a model of the overall architecture. The model can be used to document architectural decisions made following the TOGAF architecture methodology and through ongoing iteration and evolution of all architecture artifacts across the system development lifecycle. The general relationships between the DoDAF views and TOGAF phases are as follows:

- DoDAF's All Views primarily aligns with TOGAF Preliminary Phase and Phase A: Architecture Vision.

Applicable View	Framework Product	Framework Product Name	General Description
All Views	AV-1	Overview and Summary Information	Scope, purpose, intended users, environment depicted, analytical findings
All Views	AV-2	Integrated Dictionary	Architecture data repository with definitions of all terms used in all products
Operational	OV-1	High-Level Operational Concept Graphic	High-level graphical/textual description of operational concept
Operational	OV-2	Operational Node Connectivity Description	Operational nodes, connectivity & information exchange between nodes
Operational	OV-3	Operational Information Exchange Matrix	Information exchanged between nodes and the relevant attributes of that exchange
Operational	OV-4	Organizational Relationships Chart	Organizational, role, or other relationships among organizations
Operational	OV-5	Operational Activity Model	Capabilities, operational activities, relationships among activities, inputs, and outputs. Overlays can show cost, performing nodes, or other pertinent information.
Operational	OV-6a	Operational Rules Model	One of the three products used to describe operational activity – identifies the business rules that constrain operations
Operational	OV-6b	Operational State Transition Description	One of the three products used to describe operational activity – identifies business process responses to events
Operational	OV-6c	Operational Event/Trace Description	One of the three products used to describe operational activity – traces actions in a scenario or sequence of events
Operational	OV-7	Logical Data Model	Documentation of the system data requirements and structural business process rules of the Operational View.
Systems	SV-1	Systems Interface Description	Identification of systems nodes, systems, and system items and their interconnections within and between nodes
Systems	SV-2	Systems Communications Description	Systems nodes, systems, and system items, and their related communications lay-downs
Systems	SV-3	Systems-to-Systems Matrix	Relationships among systems in a given architecture; can be designed to show relationships of interest, e.g., system-type interfaces, planned versus existing interfaces, etc.

— table continues on next page

Table 1: DoDAF Version 1.0 – List of Products

— table continues from previous page

Applicable View	Framework Product	Framework Product Name	General Description
Systems	SV-4	Systems Functionality Description	Functions performed by systems and the system data flows among system functions
Systems	SV-5	Operational Activity to Systems Function Traceability Matrix	Mapping of systems back to capabilities or of system functions back to operational activities
Systems	SV-6	Systems Data Exchange Matrix	Provides details of system data elements being exchanged between systems and the attributes of that exchange
Systems	SV-7	Systems Performance Parameters Matrix	Performance characteristics of System View elements for the appropriate timeframe(s)
Systems	SV-8	System Evolution Description	Planned incremental steps toward migrating a suite of systems to a more efficient suite, or toward evolving a current system to a future implementation
Systems	SV-9	System Technology Forecast	Emerging technologies and software/hardware products that are expected to be available in a given set of time frames that will affect the future development of the architecture
Systems	SV-10a	Systems Rules Model	One of three products used to describe systems functionality – identifies constraints that are imposed on systems functionality due to some aspect of systems design or implementation
Systems	SV-10b	Systems State Transition Description	One of three products used to describe systems functionality – identifies responses of a system to events
Systems	SV-10c	Systems Event/Trace Description	One of three products used to describe systems functionality – identifies system-specific refinements of critical sequences of events described in the Operational View
Systems	SV-11	Physical Schema	Physical implementation of the Logical Data Model entities, e.g., message formats, file structures, physical schema
Technical	TV-1	Technical Standards Profile	Listing of standards that apply to Systems View elements in a given architecture
Technical	TV-2	Technical Standards Forecast	Description of emerging standards and their potential impact on current Systems View elements within a set of time frames

Table 1: DoDAF Version 1.0 – List of Products

TOGAF Phase	Focus	Applicable DoDAF Models
Preliminary	Framework & Principles	AV-1, OV-1
A	Vision	AV-1, OV-1
B	Business Architecture	AV-2, OV-1, OV-2, OV-3, OV-4, OV-5, OV-6
C	Information Systems Architecture: Data Architecture	OV-7, SV-11
	Information Systems Architecture: Applications Architecture	SV-4, SV-5, SV-6, SV-10
D	Technology Architecture	SV-1, SV-2, SV-3, SV-4 (Refined), SV-5, SV-7, SV-10 (Refined), TV-1
E	Opportunities & Solutions	SV-8, SV-9, TV-2
F	Migration Planning	SV-8
G	Implementation Governance	AV-1, OV-1
H	Change Management	SV-9, TV-2

Table 2: TOGAF ADM Phase Mapping to DoDAF Models

- DoDAF's Operational View primarily aligns with TOGAF Phase B: Business Architecture and Phase C: Information Systems Architecture activities.
- DoDAF's Systems View primarily aligns with TOGAF Phase C: Information Systems Architecture, Phase D: Technology Architecture, Phase F: Migration Planning, and Phase E: Opportunities and Solutions.
- DoDAF's Technical Standards View primarily aligns with TOGAF Phase D: Technology Architecture, Technical Reference Model, and Phase E: Opportunities and Solutions.

Table 2 overviews the primary relationships identified through analysis of these two leading architecture frameworks. Specific tailoring guidelines to adapt the TOGAF ADM methodology steps for DoDAF model outputs are documented in the detailed white paper of this analysis effort.

Relevance of the Findings

While the TOGAF/DoDAF working group initially set out simply to compare the two frameworks, its efforts recognized that the frameworks were complimentary. What this means for architects today is that they can immediately use the TOGAF ADM to build DoDAF architectures, and, in doing so, leverage a more robust, comprehensive, and complete enterprise architecture development method designed with business alignment in mind.

The working group also observed that both frameworks are currently dealing with potential gaps such as support for service orientation. The organizations that maintain these different frameworks would benefit should they come together to harmonize their efforts and work on evolving their respective frameworks to deal with the gaps such as security viewpoints, Service Oriented Architectures, and net centricity. For the architect, this ultimately means being better equipped to practice within a much broader space, which is a significant advantage in today's competitive environment.

Conclusion

Each complex architecting endeavor requires several key elements to be successful: repeatable methodology, standardized output models, formal validation, governance, collaboration guidelines, configuration management, tools, and patterns. The architect can address many of these needs through the application of The Open Group's Architecture Development Method as a disciplined process in developing the Department of Defense Architecture Framework set of views to model the architecture. ■

About the Authors

Fatma Dandashi is currently leading an Object Management Group (OMG) effort to define a UML profile for the DOD Architecture Framework. Prior to this she supported the development of the Air Force Enterprise Architecture for SAF/XC. Dr. Dandashi was task lead on the MITRE development team responsible for the DoD Architecture Framework Version 1.0 (Volumes I and II), and currently serves on the working group developing DoDAF Version 2.0. She has a PhD in information technology from George Mason University, a Master of Science in computer science from the University of Louisiana (Lafayette), and a Bachelor of Arts in computers/business administration from the Lebanese American University.

Terence Blevins is branch chief and lead architect on the Air Force Operational Support Enterprise Architecture at Mitre. He supports the U.S. Air Force in the role of branch chief and lead architect of the Air Force Operational Support Enterprise Architecture. Terence has been involved with the architecture discipline since the '80s when he was at NCR as director of strategic architecture. He has been involved with evolving this discipline since 1996 when he was first introduced to The Open Group's Architecture Forum. He was its co-chair and a frequent content contributor to TOGAF including the Business Scenario Method. Terence was previously vice-president and CIO of The Open Group where he contributed to its Vision of Boundary-less Information Flow. Mr. Blevins holds undergraduate and masters degrees in mathematics from Youngstown State University. He is TOGAF 8-certified.

Rolf Siegers is Raytheon's corporate director of architecture & systems integration and an Engineering Fellow. He joined Raytheon in 1984 and leads its Enterprise Architecture Process (REAP) Initiative, a standards-based unification of several architecture frameworks and a company-wide architecting process. Rolf also sits on Raytheon's corporate Architecture Review Board (ARB), leading and supporting veriious architecture-related initiatives. His program experience includes leading several multi-discipline software architecture teams for large-scale, software-intensive national and international systems since 1997. He is a certified TOGAF-8 architect (The Open Group), a ATAM Evaluator (SEI), and a Software Architecture Professional (SEI). He has bachelor degrees in computer science and mathematics from Huntingdon College and is a member of IEEE and INCOSE.

Judith Jones has served as an advisor to Brussels, CCTA and Industry Organizations on Enterprise Architecture and IS/IT Architecture and Frameworks, eGovernment, Practitioner of TOGAF, and Prince 2. Her background experience includes 10 years as an independent consultant and 20+ years as a business manager with ICL, now Fujitsu Services. She is an experienced program manager and principal consultant with significant skills in customer-centric programs including customer relationship management, enterprise architecture, operational business and IT infrastructures. Her consulting experience includes business change, strategy development and communications, business architecture, business development and marketing, sales, strategy and partnership alignment, business re-engineering to change, specify and deliver customer value, process and commercial initiatives. Judith is an extremely active member of The Open Group and has worked with its Architecture Forum to establish Enterprise Architecture development processes and strategies suitable for global government and the private sector. She currently leads the TOGAF 9 development effort.

Real SOA

An overview of SCA and SDO

WRITTEN BY **ANDREW BORLEY, SIMON LAWS, AND HALEH MAHBOD**

A challenge facing many organizations is how to quickly and effectively react to frequent changes in business requirements, whilst improving productivity and reducing costs. To achieve this, you need a flexible infrastructure that can meet the demands of a changing marketplace and seize emerging opportunities. To address this challenge, Service Oriented Architecture (SOA) promotes an architectural approach that replaces rigid proprietary systems with heterogeneous, "loosely-coupled" services. The Service Component Architecture (SCA), along with Service Data Objects (SDO), makes this architectural concept a reality and provides the programming model to build SOA solutions for agile businesses.

SCA is a powerful and simple business level programming model that extends prior approaches to be able to implement services based solutions. SCA defines how services can be described, assembled, and deployed in a meta-data driven fashion, independent of an implementation language and a deployment platform. The approach is based on the idea that each business function consists of one or more components brought together into a composite application. These, in turn, are composed into a network of services that create specific business solutions.

This article describes some of the key values of SCA by modeling an SOA based solution for a fictitious company called MostMortgage. We shall assume a simple business process in which an applicant signs up for a loan and provides his or her identity information and loan requirements. MostMortgage evaluates the new applicant based on their credit approval and searches for an appropriate mortgage rate.

By using the SCA programing model, MostMortgage's developer can build a solution for this problem quickly and effectively, separating the business logic from technology concerns and enabling re-use of existing applications. In this case, there is already a well understood credit applic-tion that can be re-used (CreditCheck); and MostMortgage has a subscription to a Web service that searches for the best loan rates (FindRates).

The solution developer first completes the following steps:

1) Define the business logic for LoanApproval and AccountVerification;
2) Define references for each component (this identifies what other services, if any, the component is dependent on);
3) Define the services provided by each component, if any;
4) Assemble the components and choose the binding to be used.

The MostMortgage solution (as shown in Figure 1) is then ready for deployment.

Components can be implemented in any language supported by an SCA runtime, including BPEL, Java, Ruby, and C++. Outside of any program logic, these components can be assembled or "wired" into a composition using any appropriate binding, such as WS-* or JMS.

Let's update the MostMortgage application by making one technology domain change and one business domain change.

First, we improve the security of the calls to the CreditCheck component, which happens to run in a remote data center. The MostMortgage company developer need not be concerned about these new infrastructure requirements. SCA separates infrastructure capabilities from business logic and allows the security requirements to be defined as policies during assembly. The resulting flexibility enables IT infrastructure policies to change at anytime without requiring a re-code.

Secondly, it is decided to introduce a bespoke rate optimization layer in front of the FindRate Web service (see Figure 2). MostMortgage can add value to the white box FindRate service by combining a mortgage account with in-house financial products. Here, the developer is involved, but he is able to reuse his previous work directly in a controlled and modular way by simply extending the assembly to include the new RateOptimizer component.

It is important not to forget the complexity introduced by handling data in such a heterogeneous network of services. A technology called Service Data Objects (SDO) addresses this problem. SDO offers a format-neutral API that provides a uniform way to access data, regardless of how it is physically stored. By using SDO, the solution developer will not pollute a business application with code to handle diverse choices of data access, such as JDBC Result Sets, JCA records, DOM, JAXB, and EJB entities.

SDO supports a disconnected style of data access and can record a summary based on any changes made to data objects. SDO's ability to maintain a summary of the changes made allows data transfers to include only the portion of data that has changed, therefore improving environments where bandwidth is a constraint. The change summary information can be used to resolve data access conflicts and concurrency issues.

SDO supplies a powerful yet simple programming model for data with first class support for XML and the ability to automatically persist data via the use of a Data Access Service (DAS). A DAS allows the data to be stored or retrieved from a relational database or another repository, and helps to link the SDO models to enterprise data storage.

SCA and SDO provide technologies that simplify the development of SOA solutions. SCA and SDO technologies work well together and independently. More detailed information about SCA and SDO will be available in future articles.

The importance of these technologies has led many vendors who experienced customer pain points to collaborate and develop specifications for SCA and SDO, and crystallize best practices that have been utilized for the past few years. More information about the Open SOA collaboration and its many participating vendors can be found at www.osoa.org.

You can try out Java and C++ implementations of the SCA and SDO technologies by visiting the Apache Tuscany open source project at incubator.apache.org/tuscany . Tuscany provides a simple "on ramp" for developers who want to create applications using a service-oriented approach. As an early implementer of SCA and SDO specifications, the Tuscany project is able to provide timely feedback on the specification to the Open SOA collaboration. Other implementations of this technology are also beginning to appear, for example, the PHP PECL SDO project at http://pecl.php.net/package/sca_sdo.

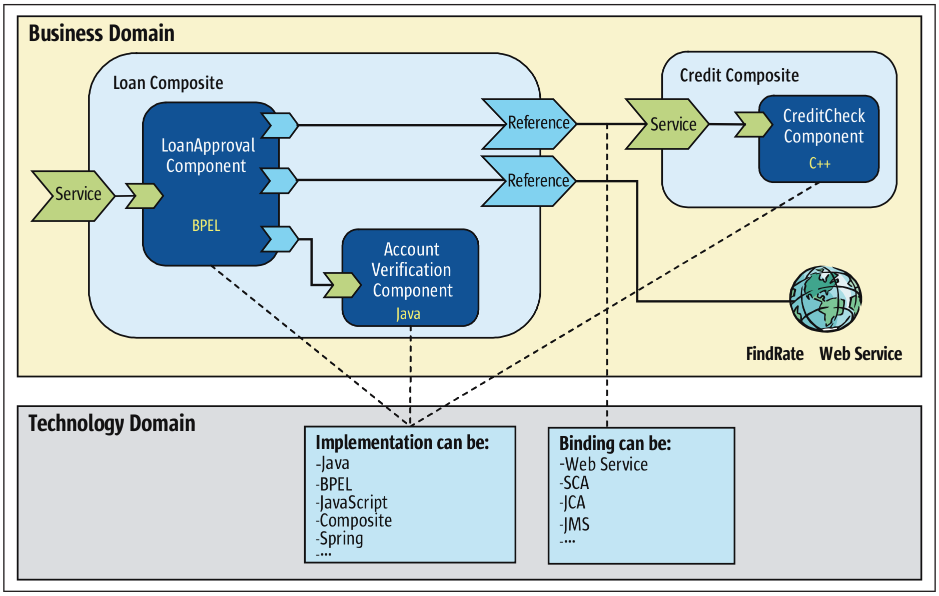


Figure 1: Loosely-coupled components in which business and technology domains are clearly separate

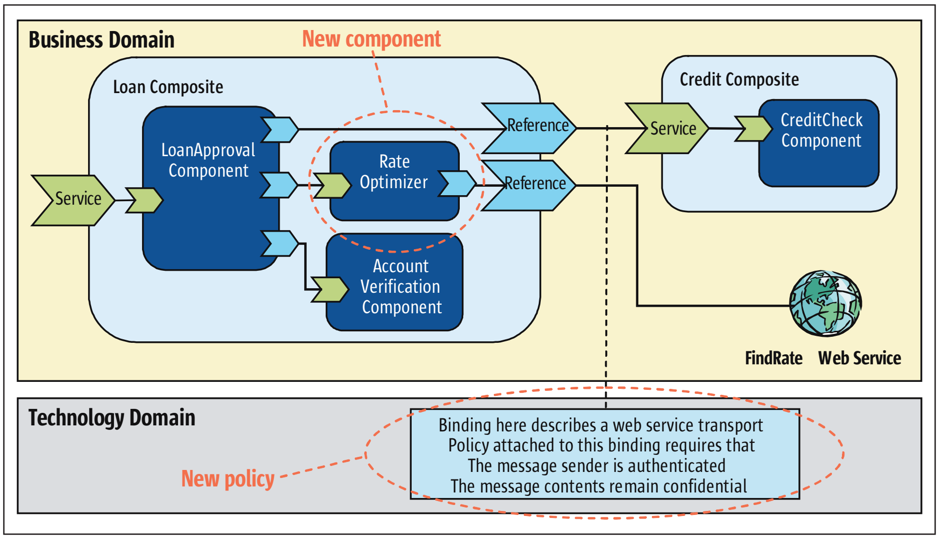

Business Domain
New component
Loan Composite
Credit Composite
Service
LoanApproval Component
Rate Optimizer
Reference
Reference
Service
CreditCheck Component
Account Verification Component
FindRate Web Service
Technology Domain
New policy
Binding here describes a web service transport
Policy attached to this binding requires that
The message sender is authenticated
The message contents remain confidential

Figure 2: Updating the application: Binding policy and an additional component

In summary, today's organizations must be able to quickly react to change. SCA promotes flexible and reusable solutions by encouraging componentization and by clearly separating business logic from underlying technology concerns. SCA and SDO independently increase developer productivity by shielding them from infrastructure complexity and the necessity to develop deep infrastructure technology skills. SCA and SDO together provide IT with a flexible model for building SOA based solutions and, more importantly, for effectively and efficiently handling change. ■

Andrew Borley is an IBMer enjoying life working on the Apache Tuscany project. He's helping to define the Service Component Architecture (SCA) specification and is a committer on Apache Tuscany, developing implementations of SCA and Service Data Objects.

Simon Laws is with IBM and is working with the open source Apache and PHP communities to build Java, C++ and PHP implementations of the Service Component Architecture (SCA) and Service Data Object (SDO) specifications.

Haleh Mahbod is a program director with IBM, managing the team contributing to the Apache Tuscany as well as SOA for PHP open source. She has extensive development experience with database technologies and integration servers.

Is Your Enterprise Architecture Healthy?

Take the Test

WRITTEN BY **DAVID S. LINTHICUM**

Working directly on SOA projects as an independent I'm exposed to many more organizations than when I was building technology. As such, I see some common patterns or issues emerging.

The largest and most disturbing is the fact that there seems to be a huge chasm yawning between the traditional enterprise architecture crowd and those looking at the value of SOA. Enterprise architecture, as a notion, has morphed from an approach for the betterment of corporate IT to a management practice, at least for some. So the person who needs to understand and implement the value of SOA is sometimes not the current enterprise architect in charge.

The core issue is an add/not change approach to architecture. While adding applications, directories, and databases to an existing architecture is easy and risk-adverse, changing architectures around systemic notions such as SOA is hard and comes with risk. Thus, many are choosing to ignore it. In many instances it's the culture, with some organizations promoting a "you fail and you're fired" approach versus a "let's try new things and seek improvement."

Another issue is that it's easier to stay high level than do actual work. Drawing diagrams, doing presentations, and writing reports is much easier than actually going out and making real changes with real benefits. Again, from above, that carries with it the notion of risk. Implementing SOA takes a lot of upfront work, as well as many changes. However, in many cases, the benefits outweigh the risks by a large margin.

Basically, if you answered no to any of the above, it may be time to look for some new ideas. In many modern Global 2000 companies, the enterprise architectures are badly broken and hinder the business's ability to change. For instance, a recent survey by the Business Performance Management Institute found that only 11% of executives say they can keep up with business demands to change technology-enabled processes. Forty percent of which, according to the survey, are currently in need of IT attention. Worse, 36% report that their company's IT departments are having either "significant difficulties" (27%) or "can't keep up at all" (9%).

In reality IT has done a poor job of supporting the business considering the amount of latency apparent when change needs to occur. CEOs pull their hair out when their IT group talks about years not months to add product lines, change markets, or merge with other companies. In many companies, the IT shop is the single most limiting factor for business success and can kill the business if left to continue as-is.

As I said, for some reason the discipline of enterprise architecture has morphed into more of a management practice, and the fundamental flaws in many enterprise architectures aren't being addressed. SOA is one approach, but in some instances SOA is not indicated; thus why I asked for an ROI study as part of the "test." However, there's always a need for good enterprise architecture.

I'm sure many enterprise architects will indeed pass, and do have most of what was mentioned on the "test" understood. Or, at least have plans in place to get there ASAP. This goes to a holistic desire to align your IT with your business. Most are out of alignment right now. ■

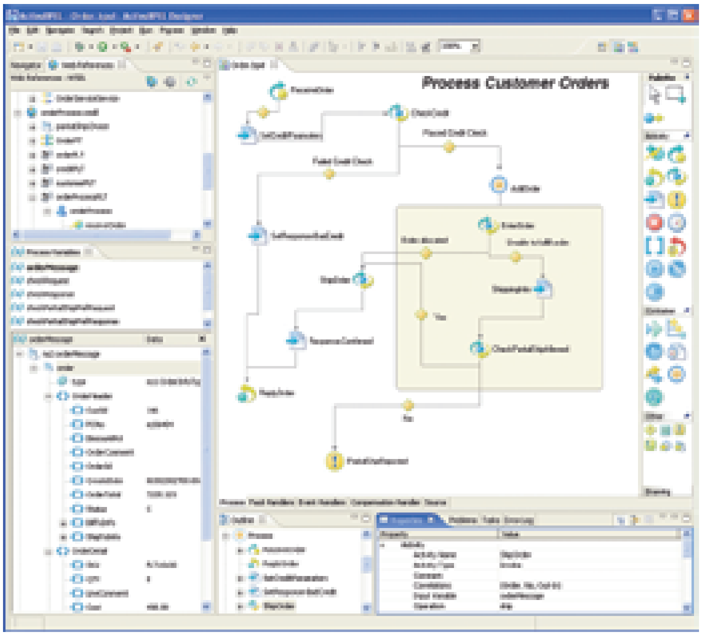
Here are a few questions to ask yourself to determine the state of your architectural standing:

1. Has someone compared the current architecture with best practices in your industry to spot issues that need correction, such as the architecture's inability to align and keep up with the business?
2. Has someone done an ROI analysis of the value of SOA, or other approaches for that matter, for the current architecture and reported it to management?
3. Do you have a complete service-, semantic-, and process-level understanding of your enterprise?
4. Do you have a common abstract model for key elements, such as customers, sales, inventory, transactions, etc.?
5. Are systems well integrated and do they communicate in real-time when needed?
6. Can you change your architecture to accommodate business changes at the speed required by management and the marketplace?

About the Author

David S. Linthicum (Dave) is an internationally known application integration and service oriented architecture (SOA) expert.

david@linthicumgroup.com


Process Customer Orders

Complex and evolving systems are hard to test...

Parasoft helps you code smarter and test faster.

Start improving quality and accelerating delivery with these products:

Awarded **"Best SOA Testing Tool"** by Sys-Con Media Readers	**InfoWorld's 2006 Technology of the Year pick** for automated Java unit testing.	Automated unit testing and code analysis for C/C++ quality.	Memory errors? Corruptions? Leaks? Buffer overflows? Turn to…	Easier Microsoft .NET testing by auto-generating test cases, harnesses & stubs	Automate Web testing and analysis.
SOAtest™	Jtest™	C++test™	Insure++™	.TEST™	WebKing™

PARASOFT®

We make software work.™

Go to www.parasoft.com/WSJmagazine • Or call (888) 305-0041, x3501