InfoWorld

GET TECHNOLOGY RIGHT

IT Strategy Guide

Service-Oriented Architecture

INSIDE

Compliments of:

ssa global™

forward faster

Introduction

To understand and apply the principles of SOA, you'd think we would have to agree first on what we mean by a "service." To a surprising degree, we haven't, but this is hardly the first time a powerful idea has been tricky to nail down. Definitions of "objects' and "components" — the ideas that powered earlier phases of software's evolution — were just as elusive.

Writing for ACM Queue, ObjectWatch CEO Roger Sessions offered one useful way to think about these successive waves of technology. All three models are ways of packaging code for reuse, he suggests. They differ in terms of where and how the code runs. Objects share a common operating system process and execution environment — for example, Linux, Windows, Java, or .Net. Components live in different processes but share an environment. Services cross both process and environment boundaries.

The environment for Web services and SOA is the global Internet. Of course, that's been true for quite a while. A decade ago programmers began using the Web's Common Gateway Interface to publish and consume services. When we build and deploy services today — using REST (Representational State Transfer) and XML-over-HTTP on the one hand, or SOAP, WSDL, and the WS-* specs promoted by Microsoft and IBM on the other — we build on that common heritage. SOA extends the tradition along two axes: data representation and data communication.

Everyone agrees XML is the lingua franca of data representation, but there's lively debate about how to use it. XML Schema, for example, is an optional feature that sharply divides communities of practice. Do interoperable services require strict formal data definition or do they require fuzziness? The perplexing answer is both — at different times, in different ways, for different purposes.

In the world of SOAP and WS-*, XML Schema typically governs the contracts between services. If the XML document that represents a purchase order isn't a valid instance of the relevant schema, it's time to throw down the warning flag. And with XML Schema, any process, running anywhere — even offline — can perform that validity check. Let's say that while flying to Chicago you use an InfoPath form to create a purchase order and then e-mail it to the approver when you land. The approver can focus on the business aspects of the order, secure in the knowledge that he or she has received and will relay to the order processing service a document that will be acceptable to that service.

What about the stuff that won't fit into the schema? Today this contextual data travels in e-mail, where we can't do much with it. Defining parts of schemas that can carry arbitrary XML content, so people can "scribble in the margins," is a key strategy. At the same time, don't ignore the growing amounts of XML data flowing through your enterprise that is not, and may never be, schematized. The prime example is RSS. All kinds of useful services, done in the REST and XML-over-HTTP style, are coming up from the grassroots. We think of RSS mainly in terms of blogging, but it also affords us a lightweight and incredibly versatile way to exchange, route, and recombine all kinds of stuff. Nearly every application that today uses e-mail to connect people and processes can be recast as an RSS-oriented service. Easier and more robust integration, no spam — what's not to like?

In fact, this low-tech approach is so appealing that many people are now discounting the WS-* stack. That's understandable and in many cases valid. While we argue about which WS-* standards will stick to the wall, a set of key capabilities is emerging. Broadly speaking, WS-* pushes aspects of data communication — security, asynchrony, reliability, routing, and proxying — up into the application layer where we can reason about these things as businesspeople rather than wrestle with them as network plumbers.

That's a lofty statement, but here's a concrete example to nail it down. Let's say your order processing service is used by a dozen applications and by hundreds of people. Suddenly, one morning, it's triple-witching time: You add a new application, you implement a mandated auditing rule, and then you have to reroute traffic because a server fails. On days like that it won't ever be easy to get home by dinnertime. The set of principles embodied in an SOA, however, may at least make it possible.

Cynics will note that we've been enumerating those principles for a couple of years now. You've heard the litany: coarse-grained messages, loosely coupled processes, data-driven integration, self-describing data, programming-language and platform neutrality, pervasive intermediation. We call this cluster of ideas by different names — grid, enterprise service bus, service-oriented architecture. It's quite possible that next year's favorite acronym won't be SOA. But many if not most of the ideas will survive — and will define the dominant style of enterprise software for years to come.

— Jon Udell

Real-World SOA: Applications as Services

SERVICE-ORIENTED ARCHITECTURE IS AN IDEA, NOT a technology. Boundless in scope, it promises both unlimited software reuse and the interconnection of everything, as long as IT is willing to wrap legacy applications in standard interfaces and construct new apps as services, the capabilities of which other software can tap into.

The idea is simple, but the execution isn't, because SOA turns the conventional model of enterprise software development on its head. Normally, programmers write software based on a set of well-defined requirements. SOA demands that organizations create an ecosystem of services that may ultimately have an army of stakeholders inside and outside the firewall. The initial challenge of SOA is knowing where and how to start — where to draw a box around a fixed set of requirements and how to build services that will yield tangible ROI while keeping an SOA fully extensible.

We evaluated dozens of SOA implementations to find a few that had a major impact on an enterprise and/or its partners. These projects are largely works in progress; some are only in their initial phases of implementation. But all can help light the way for enterprises in search of their own strategies to make a simple, powerful idea come to life.

— *Eric Knorr*

VSOA Ensures Guardian Gets It Right

Five years ago, Guardian Life Insurance decided to rethink the basic structure of its application silos, which had been developed with little attention to business goals, says Jaime Sguerra, chief architect at Guardian. "There was no standard way to build or connect applications, or any habit of reusing code," he recalls.

A new IT management team decided to change that, mainly to make application development faster, more nimble, and better aligned with business priorities. "We wanted to stay away from the one-off application and instead provide a single, common service wherever possible to reduce overall complexity. A service architecture is the way to make disparate technologies work together," Sguerra says, adding that, with an SOA in place, IT can focus on developing new applications, not reworking old ones. "Our philosophy is reuse. There's a ton of money invested in the legacy technology, and we wouldn't be able to justify a business case just for modernization."

Sguerra estimates that the SOA approach has saved approximately 30 percent of the application-development budget. After 28 months, about 60 services used by three key systems — benefits plan administration, claims processing, and policyholder administration — are now in place, as is the basic communications infrastructure. Of those services, about 50 are used by all three systems. And the work continues: Guardian plans to create 22 more services for those systems and then bring its other systems into the SOA model, Sguerra says.

At the heart of Guardian's SOA is its enterprise service manager, a collection of J2EE workflow and connector middleware tools and an IBM CICS/MQSeries message bus for managing requests. Requests come from one of three client systems — a Web portal used by customers and independent agents, a CRM system, and an interactive phone system used by customers — or from applications themselves. The enterprise service manager decides what services to invoke, in what order, and what data resources are needed. It then queues up the services and manages their interaction. At the end of the transaction, the client receives the requested result or an error message. Before the SOA was implemented, "users needed a checklist of all the system to run" for each task; "now, that workflow is built into the enterprise service manager," Sguerra says.

"We chose a central enterprise service manager because it was the best way to gain reuse," Sguerra says. Although it may make sense to have a decentralized architecture where service logic resides in multiple locations so that services communicate directly, that approach increases the risk of ending up with multiple versions of the same service as development gets out of sync across the systems, he adds.

Because independent agents use the claims, policyholder, and benefits systems as well, Guardian chose to implement its SOA through Web services. "It's harder to deploy applications to someone else's computer," Sguerra says. Developing a Web-based interface proved crucial in serving internal and external users with a single communication system.

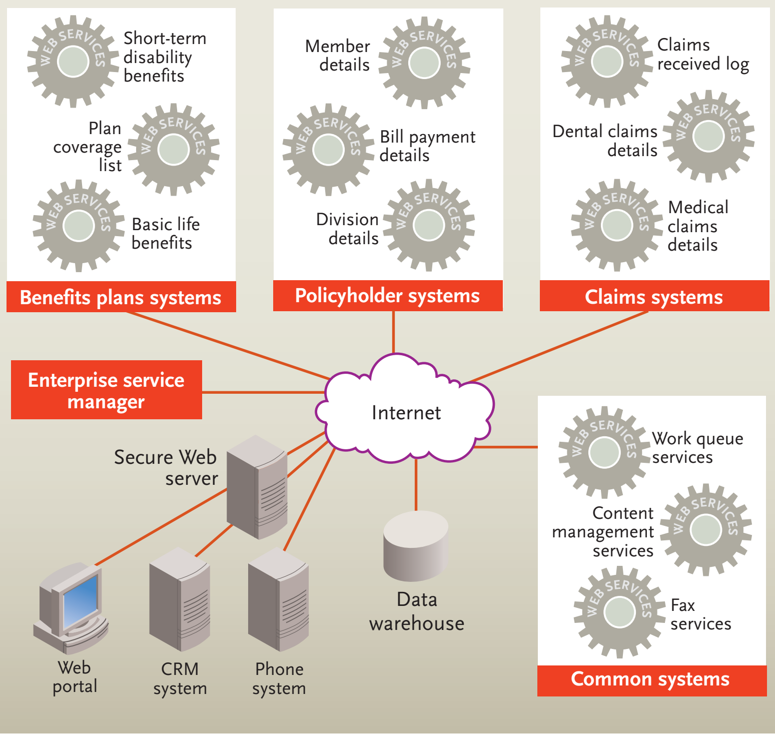
The SOA uses Web-based intermediaries in an IBM WebSphere server environment as the framework for translating and massaging data and procedure calls as needed between applications and data sources, as well as SOAP for messaging. Although services reside in a variety of physical locations — within mainframe applications, as discrete component services on other application servers, as part of a modern application, or as an external service — the Guardian SOA groups them logically by system or as a shared service.

Because Guardian uses a large number of mainframe applications, its IT team faced a challenge in exposing all those applications so they could be used as services, Sguerra notes. Guardian uses WSTL (Web Service Transaction Language) in most cases to translate service requests among services, but in some cases, mainframe applications don't support that. Rather than rewrite the mainframe apps to accommodate WSTL, Guardian uses EJBs to perform the translation outside the application. In the application itself, "we just open a door to see the EJB," he says.

Sguerra emphasizes that developing applications as part of an SOA requires a new way of thinking about application development. "You need to know up front that it is a cultural change. You can't go into it pretending it's not going to be a challenge. It takes a lot of coordination," he says, adding that business units and application developers oftentimes resist sacrificing custom interfaces that prevent services from supporting multiple applications.

Custom Policies Built From Services

Guardian Life Insurance built services based on existing applications in its benefits, policyholder, and claims systems. The company is adopting Web services standards inside and outside its enterprise to simplify service management.



Such objections usually disappear when the efficiency benefits have been proved, Sguerra adds. When a custom interface or function is truly needed, developers can usually supply a separate service that relies on a common service for the remaining functionality. There's always a trade-off, but Guardian is discovering its happy medium.

Massachusetts Takes a Spoonful of SOA

Many organizations are looking to SOA to tie together systems within the enterprise or among partners. But few face the diversity and complexity that the state of Massachusetts did when it tried to connect independent insurer, hospital, and physician systems with one another — and with the state's own systems for care, reimbursement, and billing.

"How do you craft enterpriselike functionality across hundreds of moving parts that don't interoperate with each other?" was the question the state faced in 1997, recalls Harvard Medical School CIO Dr. John Halamka, who spearheaded the effort. Because the Health Insurance Portability and Accountability Act (HIPAA) of 1998 required that every doctor, hospital, and insurer be able to exchange data for transactions, doing nothing was not an option.

The state's major hospitals and insurers examined three options. The first was to deploy a common platform and to require insurers, hospitals, and physicians who had business with the state to implement and use it. This option, however, was too complex to pursue seriously, Halamka says.

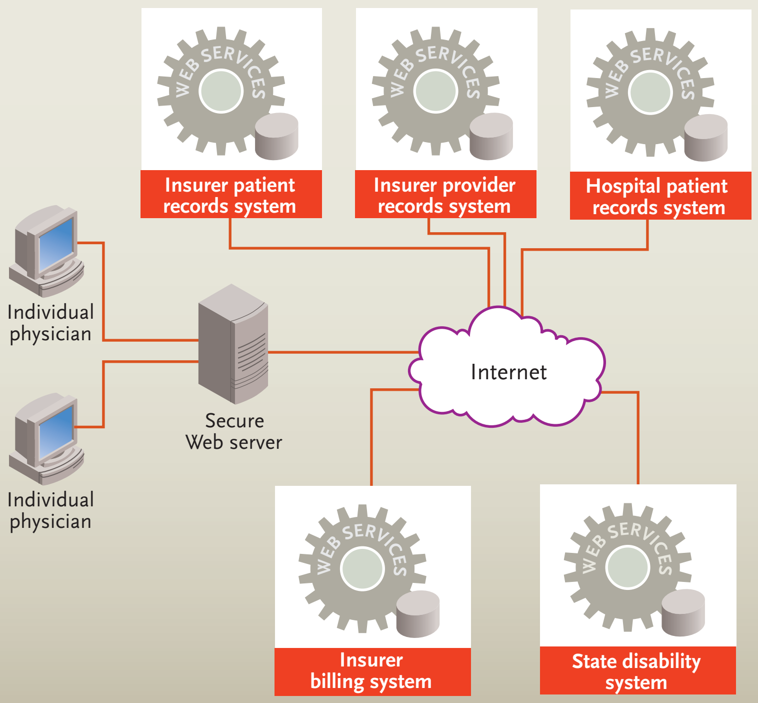
The second option was to create a unified database for patient medical, billing, and insurance data that participants could access using their own systems. That solution would have cost $50 million, Halamka recalls. But the third option — to implement an SOA that would provide the data and application translation necessary for various services to interoperate without changing their code or data structures — was viable and ended up costing just $1 million.

What Halamka calls a "Napster for health care" has also reduced the cost per transaction from $5 to 25 cents. The system now handles approximately 9 million transactions per month. To manage this network, a group of medical associations and insurers set up a nonprofit organization called the New England Healthcare EDI Network (NEHEN). Funded by the hospitals and insurers, it has one common program management officer and an annual budget of $3 million.

The result is a "closed-loop system" that ensures accurate data and validates procedures, coverage, and billing up front, thereby reducing management costs for all par-

Connecting Health Care

In Massachusetts' New England Healthcare EDI Network, hospitals, insurers, physicians, and the state government provide secure services interfaces to billing and patient information. Examples of Internet-accessible Web services include patient identification, coverage verification, physician referral, and claims approval status.



ticipants across the state. For example, "insurance companies save money by not having [to hire as much] staff to deny claims," Halamka says.

Architecturally, the NEHEN system leaves data structures and applications alone, even if they are fragmented in different locations or in different systems. "The big win is not having to rewrite old code," Halamka says, noting that some systems date from the 1970s. The system does, however, provide a central exchange that translates data structures from one system's format and standards to another's, and it maps specific transaction services from one system to another, aggregating multiple service requests and using multiple databases when necessary.

"The data and the services are very disassociated," Halamka notes. "But it doesn't matter whether they all sit in one place, as long as the doctor gets it all in a timely fashion." From a tactical perspective, as long as the system providing the data or service can communicate through TCP/IP, "that's all I need," Halamka says, adding that most of the Web services in the network were developed using Microsoft .Net, with the gateways written in Visual C++ and deployed on IIS.

Strategically, the keys have been to define the business processes along with the architecture, to understand what the data means in its various repositories, and to know what applications provide what services so that middleware can be configured to make the appropriate calls and translations. "I have the ability to control the business logic without having to modify the underlying application," Halamka says. "The middleware approach is very nonintrusive to the [individual organization's] IT agenda."

For example, patient IDs vary widely from institution to institution. Rather than require a common identifier for each patient, the NEHEN system uses a probabilistic service to check a variety of attributes — name, nicknames (such as Johnny and Jack for John), ZIP code, gender, Social Security number, insurer, and physician — and then maps patients' identities across systems. In the event that a new identifier class, such as employer ID, is used in identifying patients, the service can easily be modified to account for that, Halamka says. None of the other systems is affected or even aware of the change to the middleware, although system owners can decide whether they want their systems to use this new identifier class.

Because SOAP had not yet been developed and XML was not widely deployed, the NEHEN system initially used HTML as the common vehicle for data exchange. "In the pre-XML era, we had to use the Web for content rather than the semantic Web [XML] for data. So we used simple server-side COM components to fetch HTML pages from various hospitals and display them in a unified clinical viewer that we built," recalls Halamka, who wrote the code for this system. "It was not elegant since we had little control over the look and feel for the HTML content returned from each hospital system, but it worked. Today, with XML and XSLT [XSL Transformation] we can treat the content as data and format it as we like."

To ease adoption, NEHEN provides a Windows XP and Windows Server 2003 Web services suite that organizations can deploy to gain the required connectivity. For individual doctors, NEHEN provides a Web application that they can access either directly or through standard medical management applications, which vendors have modified to support the NEHEN system. In both cases, the underlying SOAP layer handles the communication to billing systems, medical records systems, and so forth. NEHEN may not be the answer to health care's ills, but it's eliminating lots of wasted motion in the Massachusetts system.

Countrywide Financial Simplifes Lending

For half a decade, Countrywide Financial has seen its loan, insurance, and banking services businesses grow dramatically — and its IT systems increase in complexity — as customers, products, and markets have multiplied. To meet this increase in demand, Countrywide decided to embrace a flexible SOA approach, the long-range goals for which are a familiar refrain in enterprise IT: decrease complexity, improve scalability, and reduce overhead.

Countrywide is divided into separate business units, each of which employs an IT staff that operates fairly autonomously. One unit, Countrywide Servicing Systems Development (CSSD), which primarily supports the company's loan division, began its SOA effort in 2002.

According to Peter Presland-Byrne, senior vice president of application development at CSSD, the unit chose the SOA approach because "applications support a business problem and so follow certain patterns" that lend themselves to two key attributes of an SOA: functional abstraction based on services and an emphasis on reusable components to provide those basic services. "We're trying to look at the construct of the business model from a services perspective," he says.

As it began implementing an SOA, CSSD quickly discovered that many applications had embedded within them services that duplicated functions in other applications.

"We needed to abstract the services, which is an ongoing process," and to decide which ones to choose when there were duplicates, Presland-Byrne says. He anticipates the need to abstract services further in order to support Web services because such support "doesn't come naturally" in an IBM iSeries midrange server environment, which is what CSSD uses.

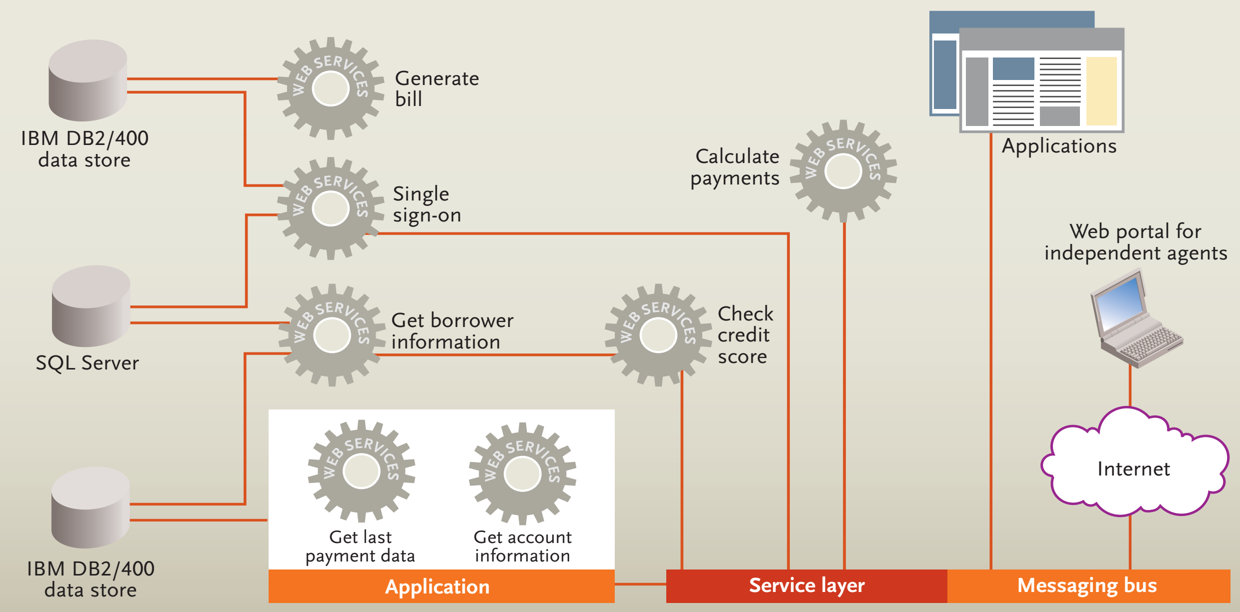
Deriving core services and having applications access common ones rather than implement their own is a key part of the SOA approach and requires a development culture that focuses on reuse, Presland-Byrne notes. To encourage adherence to the SOA, Countrywide reviews new software development to ensure it fits the SOA, provides consistent interoperability, and reuses existing services where possible.

Countrywide originally looked at SOA as its central goal but later realized the real central goal was reuse, which an SOA promotes. "If you truly support reuse, then you make SOA possible," Presland-Byrne says.

Countrywide also decided to use a messaging system

The Building Blocks of Loan Processing

Countrywide Financial's SOA abstracts services from frequently used applications, the data and functionality of which are accessible to new Web applications via messaging and service layers.



as the connectivity mechanism between applications and data sources. Because Countrywide's enterprise uses several technologies, including Java and Microsoft .Net, "messaging had to be agnostic" to ensure no proprietary dependencies were introduced into the system, Presland-Byrne says. Countrywide relies heavily on IBM's MQSeries and WebSphere MQ Integrator middleware for messaging and service handling, as well as Flashline's development environment for managing services and software components.

Although the messaging system is standardized across CSSD's applications, Countrywide does not require consistent data models. Instead, it uses middleware to ensure a consistent information flow, mapping and translating data formats as needed. For CSSD, imposing a consistent data model was believed to be unrealistic given that "the minute you bring in a third-party tool you lose the consistent data model anyway," Presland-Byrne says. "The middleware we brought in could translate these different standards. That's what integration tools are for."

More importantly, your "buffer" middleware — which contains the translation of business logic and data formats between services — must be kept separate from the service logic, Presland-Byrne says. Doing so allows separate applications to access the same service concurrently, without requiring you to touch the service code as the applications or data change. Plus, it allows you to run old and new versions of services simultaneously, either during a transition period or for different application needs. In both cases, IT can leave the services untouched.

Given that most of Countrywide's services are internal, the company does not rely heavily on Web services or associated technologies such as SOAP, although it does use Web services for a few applications accessed by customers and field agents. Countrywide has, however, tended to use XML as the semantic data standard for services and middleware because of its easy fit and wide popularity, Presland-Byrne notes.

As its lines of business deploy SOAs, Countrywide is now examining how it can extend the approach to communication among units. That will require re-examining services and eliminating duplication, Presland-Byrne acknowledges. The company has already begun consolidating identity into one service that can be accessed via SSO (single sign-on) across the enterprise.

Because each business unit was faced with different growth patterns and technology lifecycles, implementing a companywide SOA in one big bang was not possible in 2002. Now that each line of business has adopted the concept and has achieved similar levels of technology maturity, extending the architecture more broadly is something "we can now tackle," Presland-Byrne says.
— Galen Gruman

British Telecom Dials Into SOA

Telecom providers are competing tooth and nail to provide consumer and business customers with the latest and greatest value-added services. This smorgasbord of offerings includes everything from ring-tone downloads to hosted messaging, accounting, and other business services. An SOA makes perfect sense in this have-it-your-way environment because it enables providers to cobble together new offerings with those of third parties and integrate them quickly with their internal, mainframe-based billing, provisioning, and other support systems.

That's exactly the approach British Telecom (BT) wanted to take in serving its SMB broadband customers.

"SMB customers come to us for broadband access first," says Norman Street, head of Internet applications and technology at BT Retail. "But the scenario is that as they become more savvy and the Internet becomes more integral to their business they'll eventually start moving business processes online in an ASP model."

The hosted scenario is especially appropriate for businesses with fewer than 100 employees because oftentimes these organizations lack sufficient support services or suffer from understaffed IT departments. "We were looking to develop some of these service offerings in-house but also bundle them with third-party applications and mobile products," Street says.

To compete effectively, BT needed to be able to quickly test new services in the market. If a service proved profitable, it would have to scale quickly. If it didn't, BT had to be able to decommission the service and replace it with another without a lot of integration effort. BT was also looking to allow customers to manage their own subscribed services online through a Web-based interface.

It was clear that BT's current integration model couldn't support such a fast-paced scenario. "We were using a typical spoke model for integrating services with our back-end systems. We really needed to reduce the time and cost of integration and become a lot more agile," Street says.

After looking into several alternatives, BT quickly concluded that it needed an SOA. BEA Systems had supplied BT with integration technologies in the past, but for this project, BT decided to go with Microsoft's CSF (Connected Services Framework), an SOA-based service-delivery platform that functions as an extension of Microsoft BizTalk Server, SQL Server, and Windows Server 2003.

CSF provides tools and components geared specifically to the needs of service providers looking to bundle services for a variety of devices, such as PCs, PDAs, and mobile phones, and to quickly plug them in to their back-end business and operational support systems. These include a number of adapters that hook into existing BSS (business support system) and OSS (operation support system) applications and expose them as Web services. It also includes a UDDI/WSDL service directory and tools and standards for defining quality of service, managing identities using Active Directory and Microsoft Identity Integration Server 2003, and managing other service deployment and delivery functions.

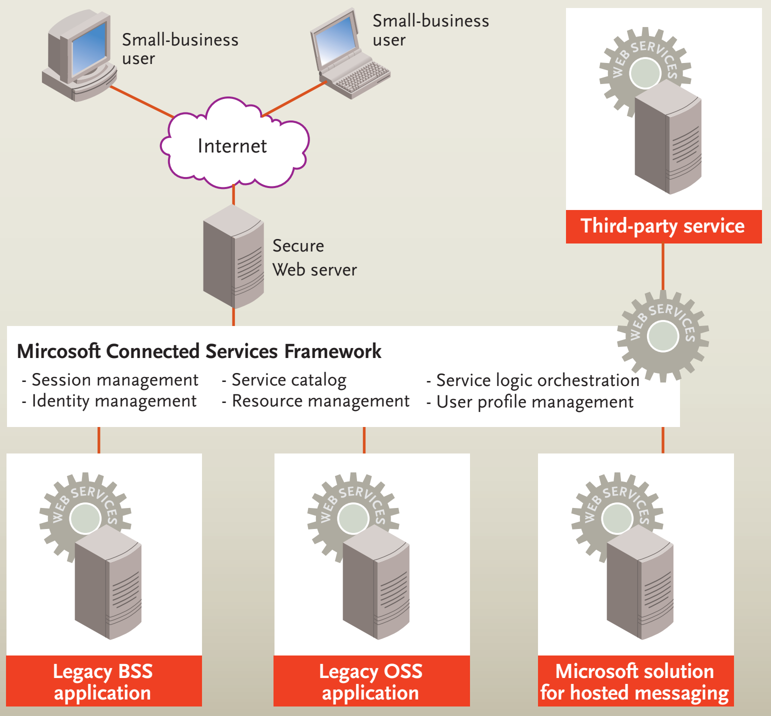
"BizTalk Server provides a workflow engine and templates to configure the business logic," Street says. It also handles message flows and other integration functions. SQL Server holds the user and product data. And of course, it all runs on Windows 2003 server.

Although BEA provided a versatile set of tools, Street and others at BT liked the idea that Microsoft had a platform specifically targeted to service providers. "Microsoft was very conscious of the advantages of a complete platform approach," Street says. "With CSF we got much more out of the box, which would take away some integration steps."

BT was also aware that Microsoft had the kind of applications SMB users would be interested in. "If we were going to be selling those applications, it made sense to get the integration technology from the same source,"

A Framework for Telecom Services

BT's SOA uses Microsoft's Connected Services Framework to expose its back-end billing and operational support systems as Web services, while BizTalk Server handles the message flows.



Street says. "At the same time, CSF had well-defined Web service interfaces and the open standards to integrate with any application." And Street was aware that Microsoft was providing tools for and encouraging .Net developers to develop to CSF. "We felt that there would be applications from third-party .Net developers that would undoubtedly be useful for our small and medium-sized business customers," he says.

CSF's standard Web service interfaces would make it easy to plug in third-party applications, build composite applications, and tie it all into their back-end systems. Introducing and retiring services would mean simple changes to the CSF interface as opposed to building or unraveling multiple layers of custom integration.

Next summer BT plans to introduce its first set of hosted messaging services based on Microsoft's Solution for Hosted Messaging and Collaboration, which provides an adapter for CSF. Future plans include bundling third-party applications, possibly migrating some of BT's existing applications off its legacy platforms to integrate with CSF, and making BT Retail's CSF-based services available to other parts of BT's business.

"CSF would make it relatively simple to, for example, provision a mailbox and then bundle it with a mobile service," Street says. As usual, when an SOA is operational, there's no shortage of ideas to expand it.

— *Leon Erlanger*

Transamerica Turns Silos Into Services

One of the real promises of SOA is enabling companies to leverage existing legacy systems as a set of core, reusable Web service building blocks that can be assembled to create new processes and applications quickly and inexpensively. That's just what Transamerica Life Insurance was looking for when it sought to provide its business partners with self-service access in real time.

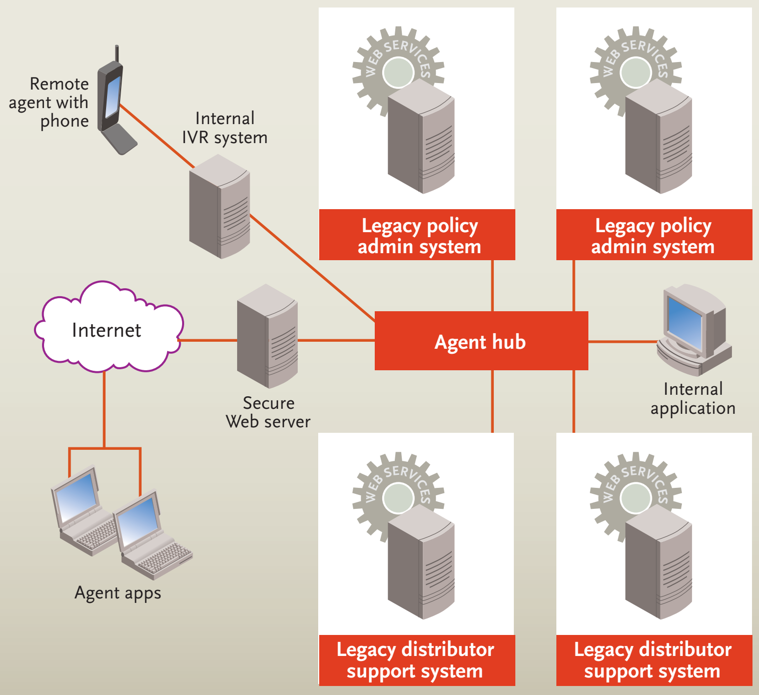
"We exchange a lot of data with our different distributors outside the firewall," says Jeff Gleason, director of IT strategies at Transamerica's annuity products and services division. "A lot of that was being done via flat-file batch data exchanges."

Gleason realized that to stay competitive, Transamerica would have to provide its business partners with real-time access to its numerous legacy back-end systems. That's a complex undertaking, however, for several reasons.

"We live in a very challenging legislative environment, with Sarbanes-Oxley, the Patriot Act, anti-laundering laws, tax laws, and other types of controls," Gleason says. "As legislation and the competitive environment change, we need to be able to make changes to our internal systems quickly, including changing rules, the ways taxes are calculated, or the way a product functions given specific criteria. At

Self-Service Apps for Insurance Agents

Transamerica's SOA exposes legacy system functionality and data as a set of core Web services, which provide the building blocks for self-service apps that can be tailored to the needs of insurance agents.


Remote agent with phone
Internal IVR system
WEB SERVICES
Legacy policy admin system
Legacy policy admin system
Internet
Secure Web server
Agent hub
Internal application
Agent apps
Legacy distributor support system
Legacy distributor support system

the same time, we often have to customize products and services for each of our different distribution channels. And sometimes we get requests from specific banks or broker dealers to create products for their particular niche markets or new areas they want to compete in. These things often impact our internal business processes."

To provide real-time access, Transamerica also needed ways to validate agents as licensed and appointed to sell specific products in specific states. "Validating an agent is not as simple as looking something up in a system," Gleason says. Depending on the commission structure there might be many different rules about how the commission hierarchy, which has up to 10 levels, is set up internally."

With all this complexity, Web services and SOA were natural choices for Transamerica. "We needed a solution that was both tightly integrated and loosely coupled," Gleason said. A lot of business logic exists within Transamerica's current back-office legacy systems. "Instead of continually recreating that logic, it made sense to create a set of core services to expose that logic so that it could be accessed by different applications, processes, and channels, whether they were batch processes, real-time processes over the Web, internal fat-client applications, or even IVR [Interactive Voice Response] systems."

To support all these methods of access, each Web service would have to be capable of accommodating not only straight SOAP Web services calls but also MQSeries and JMS (Java Messaging Service). These core services could then be mixed and matched as part of a larger group of composite services that could accommodate the needs of various channels and individual business partners.

SeeBeyond's ICAN (Integration Composite Application Network) provides the tools for exposing as Web services the back-end mainframe transactions that provided much of Transamerica's existing functionality. One such tool, eGate Integrator, is used to provide the integration broker and message transformation from one data format to another.

Other SeeBeyond tools leverage BPM capabilities to create the "agent hub" that handles the complex message routing required. So, for example, in response to a request from a user application, one of Transamerica's three or four legacy policy administration systems might return a cryptic product descriptor. That product descriptor could then be passed to a separate distributor support system that would return a more user-friendly product name. That or another distributor support system might also handle commissions and manage the information on which agents are appointed in which states to which products.

The end-user applications use an insurance industry XML schema developed by the nonprofit Association for Cooperative Operations Research and Development (ACORD) standards organization. The XML data is then transformed into the proprietary format required by each legacy back-end application. Basically, this ensures the loose coupling essential to an SOA. "If we acquire another company and we need to validate agents against their distributor support system, it's simply a matter of creating an adapter from ACORD to the proprietary format required by that system. Everything on top speaks ACORD and doesn't care what the implementation of the service is behind the scenes," Gleason says.

SeeBeyond also provides the tool for creating portals and graphical portlets. The ultimate dream is to provide each business partner with a single custom application and interface to all the back-end systems necessary to fulfill each partner's specific needs.

Gleason advises those getting involved with SOA to do as much planning and preparation as possible. "If we had it to do over again, we'd spend a lot more time up front prototyping, testing, and setting up the architecture and standards. After all, you're creating one object and one service that will be used by lots of different processes. You have to make sure you don't make changes to the service that help one project but break others," he says.

— L.E.

Debating SOA Deployment Challenges

The benefits of SOA are out there — more flexibility, faster development of business apps, for example — but getting everything up and running means planning for the long term, especially when security is still something of a question mark.

Motorola's three-year-old campaign to build an SOA has yielded deployment of 180 services so far, and is expected to expand to 1,000 by early 2006. The company anticipates that the number of services will ultimately top out at 1,500, said Toby Redshaw, Motorola corporate vice president for IT architecture, emerging technology, and e-business. He cited the competitive edge afforded by an SOA while also noting deployment issues.

"One-hundred-eighty doesn't sound like a lot, but that clearly puts us in the top 5 percent globally, maybe a little better than that," Redshaw explained.

Motorola's SOA, as would be expected, relies heavily on Web services. "We think we're in a competitively advantageous [position] because we've been playing this for three years," Redshaw said. He also emphasized the benefits of "small agile" over "big slow" in business automation.

Brought into Motorola to turn around the company's IT systems, SOA was to be the basis of the company's strategy.

"Back then, we called it a service-based architecture," Redshaw said. "We believe this will let us add business services [at a] two- to three-times rate of speed," he added. SOA also allows Motorola to do more with less, he added.

Citing the need for SOA, Redshaw said companies these days cannot afford to be less efficient with their computers than their competitors can. "Today, your company will get killed in four to five quarters," he said. Motorola also expects its IT suppliers to be supporting SOA.

"It sounds like a light beer commercial, but [an SOA is] faster, it's cheaper, it's better," at providing a more coherent IT strategy, Redshaw added.

Motorola's SOA features business activity monitoring for Siebel and Oracle applications as well as a supply chain management system. Building an SOA allows IT staff to "drill down into the legacy spaghetti and harvest the gold," by expressing legacy systems as Web services used in a component layer, Redshaw said.

Deploying an SOA, however, requires critical components such as a UDDI directory and Web services security, management, and governance. Although UDDI has been considered disappointing in enabling provision of Internet-based Web services directories, Redshaw is a believer. "If you don't have a good directory to go find these things in, it's 'game over.' I don't care how good the other parts are," he said.

The Good, Bad, and Ugly

Web services security and management are important, given corporate priorities on security and the potential of destructive payloads in a Web services message, according to Redshaw. "[The] fastest path to get fired in IT today is a big security problem," he said. A governance layer, meanwhile, enables optimization in an SOA, Redshaw explained.

An SOA allows for team-based development, building of business projects based on existing processes and reuse of components. It also lets IT staff deliver exactly what business teams asked for, according to Redshaw.

SOA has had its drawbacks, Redshaw acknowledged, including immature standards in early years, the security challenges of loosely coupled architectures, and performance concerns caused by loose coupling of software components and bandwidth-intensive XML. "The security issues are not small. You need some serious pros on your team to address this," Redshaw said.

An SOA can solve problems pertaining to information exchange among disparate business systems as well as address the need to provide services to multiple parts of an organization, said Lou Absher, data manager at the University of California, Santa Barbara.

"I do think the key ... is you have to have the rules of implementation and you have to refer back to them as you are going through this process," Absher said.

BEA Systems CTO Mark Carges also emphasizes the transformational nature of an SOA. "This is not something that happens overnight," he said.

SOA is intended to address technology "pain points," including providing more flexible architecture, application and data integration, and business process implementation. Other goals include boosting enterprise portal initiatives and enabling customized application development and composite applications, according to Carges — not to mention streamlining of supply chains, more effective integration with business partners, and allowing employee self-service.

Challenges in SOA deployments include platform heterogeneity, message brokering, data silos, security, and lifecycle management, Carges said, noting that security and the issue of data silos can be addressed through security and data services layers respectively. Metadata and service-level agreements also are critical in an SOA.

The true measure of an SOA is its ability to enable service reuse, Carges said. "At some point, someone has to stop writing code," he commented.

Improving SOA Security Handshakes

Despite the benefits, SOA users and those in the planning stages don't hold back their criticism of what Web services and SOAs need to scale beyond the four walls of a single enterprise.

Miko Matsumura, vice president of marketing at of Infravio, said the issue of provisioning services is a major hurdle and more work is needed to automate a process that in some cases is now handled by users filling out a form in a Word document.

Rick Gaccia, senior director of product management at Oracle, agreed, adding that companies are struggling with how to put details of the Web service into a directory.

"You need to know what the schema is and how the lifecycle is managed," said Wendell Lansford, a senior vice president at Systinet Software. He added that companies start out without a game plan, when what is really needed is a series of deployment best practices. "They need check points and control procedures to go from a pilot project to a production model," Lansford said.

In order to scale out an SOA, users need to figure out how services will be assimilated into different environments, added David Linthicum, CTO of Grand Central.

"How do you mediate different protocols, semantics, and security?" Linthicum asked. He added that there is no directory standard, which is another problem. "We need a standard directory everyone can agree on to make provisioning against all the SOA platforms out there easy."

As far as automating SLAs between producers and consumers of Web services, all agreed it is likely to remain a manual or person-to-person procedure that is done offline and then incorporated into the Web service.

Linthicum said the process is laboriously slow, involving legal departments and many business meetings between providers and customers.

"As services become more standard we need automated agreements, but nothing like that exists today," he explained.

Matsumura added that people still like to do business on a personal level and this becomes the gating factor in deploying an SOA with partners. He pointed out that the biggest barrier to linking portals, for example, is not technology but the legal agreements.

Yet another point of contention with SOAs is the inability to monitor SLAs in an SOA as compared to monitoring a service on the Web.

"There's no visual way to monitor an SOA," said Linthicum, explaining that SOAs have hundreds or thousands of touch points where it might be failing as one application is bound to another.

One Grand Central customer, Linthicum noted, came up with a unique solution. Instead of monitoring the service it offers to its customers, this company monitors the services they consume. "They know what they promised and so they make sure their partners meet their agreements," Linthicum said.

Regardless, the biggest shortcoming in SOAs is security, authentication, and authorization. For example, there is no easy approach to token exchange if one company uses SAML and another company uses a different security protocol. Going a step further, "two SAML versions don't even communicate. You need a middleware layer to deal with it," Linthicum added, calling it a huge mess that needs to be solved. "This is the biggest exposure in SOAs."

— Paul Krill and Ephraim Schwartz

Building SOA Your Way

A fault line runs beneath the groundswell that began a few years ago with XML Web services and continues today as SOA (service-oriented architecture). True, nearly everyone agrees that XML messaging is the right way to implement low-level, platform-agnostic services that can be composed into higher-level services that support enterprises business functions. Yet, here's also a sense that the standards process has run amok.

IBM, Microsoft, and others have proposed so many Web services standards that a new collective noun had to be invented: WS-* (pronounced "WS star" or sometimes "WS splat"). The asterisk is a wild card that can stand for Addressing, Eventing, Policy, Routing, Reliability, ReliableMessaging, SecureConversation, Security, Transactions, Trust, and a frighteningly long list of other terms. Surveying this landscape, XML co-creator Tim Bray pronounced the WS-* stack "bloated, opaque, and insanely complex."

It wasn't always so. Simple forms of XML messaging were succeeding in the field long before any of these standards emerged. At InfoWorld's SOA Executive Forum in May 2005, Metratech CTO Jim Culbert described how his company's service-oriented billing system worked back in the late 1990s. The messages exchanged among partners were modeled in XML and transported using HTTP with SSL encryption — the method still used for most secure Web services communication today. Seybold analyst Brenda Michelson, who was then chief architect at L.L. Bean, tells a similar story about that company's early experience with Web services.

Two factors were prominent at the time. First, the Web offered a simple, pervasive integration framework, one later promoted to the status of architecture and assigned the label REST (Representational State Transfer). Second, XML provided a universal way to define services in terms of the data they produced or consumed, rather than in terms of the code that produced or consumed the data. In combination, these factors were — and still are — powerful enablers.

Cranking Up Complexity

How, then, did we arrive at WS-*, which Culbert and others say is a cart that's gotten way ahead of its horse? One theory holds that the heavy-hitting vendors, working closely with key customers and partners, have ratcheted complexity up to a level that only they will be able to sustain. Because those specs are so far ahead of what most users need today, their development hasn't been an organic process driven by well-known requirements.

Patrick Gannon, president and CEO of OASIS, the standards body now coordinating a number of the WS-* specifications, reluctantly agrees that users should have been more engaged from the beginning. "I wasn't involved in creating those specs without formal user requirements on the table," he says. "But I'm a pragmatist; the specs are there."

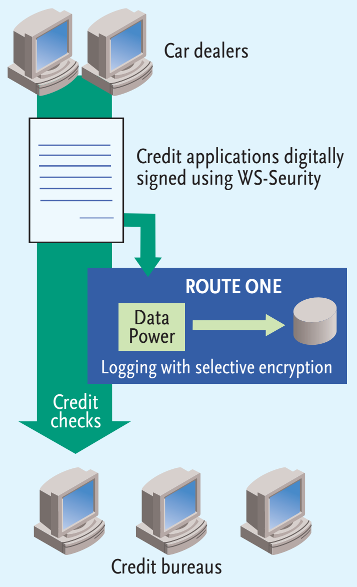
Another view holds that industry heavyweights, who have paid their dues when it comes to security, transactions, and reliable messaging, are indeed qualified to translate their experience in these matters into the language of XML. TN Subramaniam, director of technology at RouteOne, which makes software that streamlines credit management applications on behalf of car dealers, learned that lesson the hard way. At one point he began drafting his own spec for single sign-on, only to abandon it when he discovered SAML, which his joint-venture partners enthusiastically adopted because all their identity management vendors — including Netegrity and Oblix — were supporting it.

"What are the chances," Subramaniam asks, "that five architects meeting every other day will iron out all the possibilities, versus having a committee thinking it all through in great detail with all the vendors on board?"

It's tempting to interpret the tension between these two perspectives as a replay of the cathedral and the bazaar — or perhaps instead, WS-Heavy and WS-Lite. In that dichotomy, WS-Heavy would refer to the security, reliability, and scalability that WS-* claims to deliver, whereas WS-Lite would mean the speed, simplicity, and agility that attract labels such as REST, AJAX, and RSS. None of the enterprise architects we interviewed for this story has pledged allegiance to either of these camps, though. They're intensely pragmatic people who will do whatever it takes to get the job done, and it's instructive to learn how they are — and are not — making use of Web services standards.

RouteOne's Connections

This clearinghouse for car buyer credit checks uses WS-Security and SAML to secure consumer data.


Car dealers
Credit applications digitally signed using WS-Seurity
ROUTE ONE
Data Power
Logging with selective encryption
Credit checks
Credit bureaus

RouteOne: Securing Credit Checks

Although end-to-end SSL is often sufficient, RouteOne's Subramaniam has two reasons to prefer the more granular approach enabled by WS-Security. First, it's necessary to digitally sign the credit applications his application transmits, and to do so according to rules understood by service partners. WS-Security defines such rules, although admittedly, and unfortunately, too many of them. One method is to put the signed application into the body of the SOAP message; another is to use SOAP with attachments. In the end, there was no agreement among the service partners, so RouteOne uses both. That's frustrating, but Subramamian would rather have two rules than none.

The second reason touches on one of the deep principles that motivates the design of the WS-* stack: pervasive intermediation. RouteOne is required to maintain meticulous audit logs and would prefer not to have to encrypt all of them. So it's using DataPower's XML router/accelerator to selectively encrypt only sensitive items such as gross pay and Social Security number. Because it's a standards-based intermediary, the DataPower box can straightforwardly modify RouteOne's XML message traffic in this way, and it could be swapped out for another appliance that did the same thing.

When services communicate directly, as many if not most still do, there's no need to define the rules of engagement that enable service intermediation. Today's most visible exemplars of WS-Lite — Amazon and eBay — use Web services in a point-to-point way. In that mode there's not much difference between SOAP/WSDL APIs and REST APIs, so it's not surprising that developers who work with these platforms overwhelmingly prefer the REST flavor. But when you do need to flow your XML traffic through intermediaries, SOAP and WSDL suddenly make a lot more sense.

Subramaniam is a pragmatist, however. Plain XML over HTTP, sans WSDL, also plays a role in RouteOne's internal and external affairs. Because it's a no-brainer to put a servlet interface onto an internal legacy system and pull XML data through it, that strategy is used where appropriate. Some of RouteOne's external partners use the same approach, and because "they're making money hand over fist" doing so, Subramaniam can't mandate otherwise. Instead, RouteOne normalizes inbound traffic to SOAP and WSDL in order to enable its expected future use of BPEL (Business Process Execution Language) for service orchestration. Today, partners who don't present SOAP and WSDL interfaces are not competitively disadvantaged. But the tipping point may not be far off.

RouteOne depends on both SAML and WS-Security, and Subramaniam wishes he could use a standard form of reliable messaging, too. "If I don't send a message, we are losing money," he says. Drawing inspiration from ebXML (e-business XML) and JMS (Java Message Service), he specified — and is now using with partners — a scheme that guarantees orderly and reliable delivery of messages. But he'd rather it were otherwise and hopes that OASIS will succeed in merging the two proposals it is now hosting: WS-Reliability and WS-ReliableMessaging. This duplication is "really, really bad," Subramaniam says. "I wish we had a common spec so I could dump my stuff and just use it."

Corillian: Point-to-Point Simplicity

Many service-oriented systems don't require reliable messaging and, according to Scott Hanselman, chief architect at Corillian, his company's banking middleware falls into that category. Corillian's product, called Voyager, handles services touched indirectly by 25 percent of all users of online banking, Hanselman says. "But the only transaction they care about is the one at the host." So he's not worried about the merger of WS-Reliability and WS-ReliableMessaging. Although he does make use of WS-Security, he regards SSL as equally effective in most cases. That approach precludes routers and intermediaries, he admits, "but rarely do I use them, because nine times out of 10 we're doing point-to-point messaging."

He's also dismissive of UDDI, the much-maligned standard for publishing directories of Web services. What about the argument that services not found in the yellow pages won't be reused? Hanselman doesn't buy it. Finding services isn't really a problem for developers, he says. Using them easily and effectively is. Imagining a fictional average developer named Mort, Hanselman opines that SOA will be a nonstarter until we can shield Mort from XML angle brackets and X.509 certificates. To that end, he thinks the most important standard is WSDL because it's a tool-enabler.

Of course WSDL has earned its fair share of criticism, too. RouteOne's Subramaniam thinks that the "goofy" complexity of WSDL 1.1 made it a ball and chain that SOA has had to drag around, and he hopes that the "much cleaner" WSDL 2.0 will lighten the load.

Perhaps, Hanselman says, but "you can't unring the bell." Millions of Web services transactions ride on WSDL 1.1 and will for a long time to come. Using WSDL 1.1, Corillian was able to describe the objects, messages, and services at the core of Voyager and to bind those descriptions to internal machinery that doesn't speak XML. As the need arose, the company created alternate bindings that enable customers to see the engine through a Web services lens. If WSDL 1.1 was an 80 percent solution, Hanselman thinks, then WSDL 2.0 might be a 90 percent solution, but either can deliver crucial leverage.

Ohio State: Securing Vital Signs

The most widely adopted of the advanced Web services standards is clearly WS-Security. Beyond that it's hard to find practitioners who have worked with the more exotic beasts in the WS menagerie, but Furrukh Khan — who holds joint appointments in the colleges of engineering and medicine at the Ohio State University Medical Center and is broadly responsible for its medical IT — tells a fascinating story about his transition from basic to advanced Web services.

In this scenario, vital signs flowing from monitors are recorded in databases and are simultaneously delivered to smart clients that observe, replay, analyze, and annotate the streams of data. The streams must be delivered to a lot of clients reliably, securely, and in near real time.

A first implementation, based on Microsoft's WSE (Web Services Extensions), made use of WS-Policy, which hasn't yet found a home in a standards body but likely will soon. WS-Policy was used to declare the means of authentication to back-end databases — for example, to require X.509 certificates signed by a specified key — as well as the required payload signature and encryption.

The current implementation — based on the beta version of Microsoft's Indigo, a Windows implementation of a stack of advanced Web service protocols — uses WS-ReliableMessaging to ensure orderly and reliable delivery of messages. And it uses WS-SecureConversation to optimize that secure, reliable channel for high-volume traffic.

Khan explains that WS-Security alone, in concert with WS-Policy, could not sustain near-real-time traffic. The protocol, which required frequent exchanges of credentials with the identity management system, was too chatty. WS-SecureConversation, which enables caching of credentials, streamlines the protocol. That, coupled with a feature of Indigo's implementation of WS-ReliableMessaging that enables a router to broker a connection between two end points and then get out of the way, resulted in a massive scale-up.

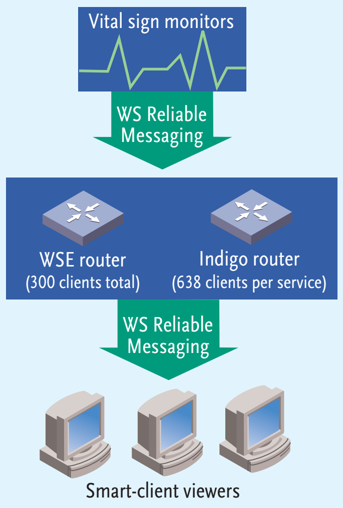
"Before, with WSE, each router limited us to 300 clients," Khan says. Indigo can support 638 clients per router, he adds, and with optimization, that many clients for each service running behind the router. "So if you keep on adding services, it scales linearly," he says. The system currently supports more than 1,000 clients, all observing vital signs simultaneously every 30 seconds.

Reflecting on the transition from WSE to Indigo, Khan echoes Scott Hanselman's point about shielding developers from XML. WSE handled the basic scenarios, he says, but beyond those, "we had to go into the schema and do all the angle brackets." Thanks to Indigo's higher level of abstraction, that problem vanished.

More broadly, Indigo made a harder problem — the appropriate use of Web services in concert with platform-native services and transports — tractable. "Behind each Web service there's an MSMQ [Microsoft Message Queue] and an enterprise service," Khan says. "In the Microsoft domain, enterprise services are completely different from Web services, MSMQ lives in its own world, and XML has its own toolset." Different team members had to be experts in different disciplines; no one person could master them all. From Khan's perspective, Indigo gives average developer "Mort" the leverage he needs.

Medical Monitoring

At Ohio State Medical Center, reliable messages carry real-time patient data, which are consumed by fat clients.



Providence: Enforcing Contracts

Providence Health Systems deploys what's becoming a typical two-tiered SOA to support its clinical and business applications and its physician and patient portals. A set of coarse-grained services, which map closely to business processes, are woven from another set of more elemental services. Although some advanced standards are in use, such as WS-Security, Providence doesn't deal with them directly. "We rely on our vendor's implementation of the security stuff," says Mike Reagin, vice president of development at Providence.

The vendor in this case is Infravio, whose Web services management system provides the framework within which Providence deploys and manages its services.

Infravio implements UDDI, but Reagin says that, with relatively few services in play, directory lookup isn't a big deal. Declaring and enforcing policies that control the use of those services, however, is a very big deal, as is monitoring service activity.

In Infravio's model, services are provisioned as producer/consumer pairs, each of which is governed by a contract. The master patient index, for example, is a common service used by both the physician and patient portals but in slightly different ways. The patient's health-plan member number, which appears in the patient portal, must be stripped from the physician portal. By creating separate WSDL interfaces for separate consumers, Infravio enables the common service to be reused rather than duplicated. This variation is achieved in a declarative way, rather than by writing code.

Providence's SOA deployment is, for now, largely internal. Services feed its outward-facing portals but are not yet directly exposed to partners. That day will come, Reagin feels sure, and when it does, he expects that his use of the core standards, SOAP and WSDL, will enable more advanced scenarios: orchestration, reliable messaging, policy-governed security, and auditing.

Which pieces of the WS-* stack will enable those scenarios? Reagin doesn't lose sleep over the question. When the time comes, he'll buy — rather than build — the needed infrastructure.

Pfizer: Trusting the Fabric

Security and reliable messaging are key requirements for the Pfizer Global Pharmaceuticals (PGP) group. The pharma giant's SOA deployment meets those requirements with the help of Blue Titan's Network Director, which manages PGP's Web services traffic across the enterprise.

On the security front, Blue Titan's "fabric" enforces a policy that routes requests through a DataPower intermediary for compliance auditing and through an Oblix system for authentication. Martin Brodbeck, PGP's application architecture director, sees WS-Security as the integration framework for these activities. Although he doesn't deal directly with related standards, such as WS-Policy or WS-Trust, Blue Titan does in fact support them.

It's worth noting that a number of standards said to be "vendor-driven" are primarily of interest to vendors. For example, another architect interviewed for this story was hands-on with WS-Security but unaware that WS-Trust plays a role in his implementation. Why? The WS-Trust protocol is spoken only between his security broker, VordelDirector, and his identity provider, Entrust. The messages exchanged between his company and its Web services partner have nothing to do with WS-Trust, says Mark O'Neill, CTO of Vordel.

"We and Entrust chose to use it because it's a spec that we don't have to work out ourselves," he says. The WS-Security protocol used by the service end points and the WS-Trust protocol used by infrastructure components are "solving completely different problems — it just so happens that both involve specs that begin with WS."

Along with security, reliable messaging is a key PGP concern. With various flavors of message-oriented middleware in play, along with multiple versions of some of these (such as JMS), the company values the Network Director RM's capability of hiding the differences. Although that product's support for WS-ReliableMessaging is not immediately relevant, PGP is evaluating Indigo, which natively supports the standard. "Blue Titan in concert with Indigo will make RM [reliable messaging] really, really easy to do," Brodbeck says.

To the short list of important standards such as WS-Security and WS-ReliableMessaging, Brodbeck adds RSS, the wildly popular format for Weblog syndication. That PGP would regard this variant of WS-Lite as strategic may surprise you, but if you think about how collaboration and knowledge management drive the top line in an organization such as Pfizer, it shouldn't. What PGP envisions, however, is not your garden-variety blogging software. "We have to recontextualize RSS for the enterprise," says Richard Lynn, PGP's vice president of global applications and architecture.

PGP's requirements include virtualizing RSS feeds so that they're independent of hard-coded addresses, aggregating them for specific business functions and securing them using the same kinds of declarative policies that govern existing Web services. According to Frank Martinez, founder and CEO of Blue Titan, a forthcoming release of Network Director will address these requirements, building on the product's capability of wrapping WS-Heavy infrastructure around WS-Lite protocols.

Heavy, Lite, or Just Right?

When you regard the WS-* stack as a whole, you have to conclude that the critics are right: It really is a monster. Taming it will require, in part, a unifying conceptual framework. That's a point that Gannon, Khan, and Subramaniam each make in different ways.

Gannon points to a series of blueprints and reference models published by OASIS. These documents aim to help architects understand how the various WS-* specs, which are designed as modular building blocks, combine to solve specific problems. For Ohio State's Khan, it's not just about blueprints. He needs a toolkit that tames the complexity and thinks Indigo will be that toolkit.

RouteOne's Subramaniam hopes that a recent initiative called JBI (Java Business Integration) will be a unifying force in the Java world. What's hard about Web services, he says, "is that you have to see the whole picture — WSDL, and then SOAP, and relevant parts of WS-Security, and BPEL." He's anxious for vendors such as SeeBeyond, which was recently bought by Sun Microsystems, and webMethods to embrace JBI. "When you can see how it all fits together in the big picture of JBI, a very nice infrastructure emerges," Subramaniam says.

Of course, toolkits and frameworks are double-edged swords. Even when wire protocols are standard and open, you can get locked in to proprietary abstractions layered on top of those protocols. That's why pragmatic architects and developers who don't yet need advanced WS-* features tend to focus on the basics: SOAP and WSDL.

"If you need some kind of envelope, why wouldn't you use SOAP?" Subramaniam asks. "And if you need to describe your interfaces precisely, why wouldn't you use WSDL?" Frank Grossman, co-founder of Mindreef, says that most of the customers who use his company's SOAPscope diagnostic suite have adopted this strategy, which he adroitly labels "WS-JustRight."

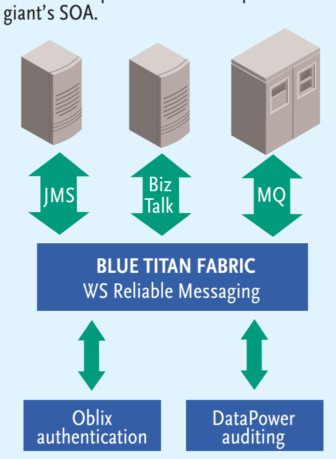
For Grossman and others, WS-JustRight means using SOAP and WSDL to strike a balance between formal contracts and agile interoperability, while laying a foundation for future use of more advanced SOA features. PGP's Brodbeck agrees that WSDL is the key enabler of reusable business transactions and processes. He also extends the definition of WS-JustRight, however, to include enterprise-enabled RSS as the key enabler of reusable content.

For many practitioners, WS-Just Right now includes aspects of WS-Security, too. For a few, it includes reliable messaging, transactions, routing, and policies related to these features. The definition will evolve over time, but the only one that really matters now is the one that's just right for you.

— Jon Udell

The Pfizer Fabric

Identity-based security and reliable messaging form the warp and woof of this pharma giant's SOA.



SOA Planning and Design: It's Still the Wild West

Remember methodologies and CASE tools? They were procedures and approaches for designing and building information systems, typically using structured and object-oriented design techniques. This was huge back in the day, but ultimately methods and CASE technology retreated back into the world of development.

Today, we are all doing things differently when it comes to architecture and design (or doing no architecture and design, based on what I'm seeing out there). You can trace a lack of planning back to architecture failures almost every time.

Along comes SOA. What SOA is, at its essence, is really good architecture in a standards-oriented wrapper. It's also complex distributed systems architecture that requires a ton of forethought and planning for architects and developers to get it right.

Moreover, SOA is more of an ideal than a simple architecture and development project. You're really never done, but you are heading in a unified direction — the ability to create an architecture that provides an infrastructure of agility for business, allowing them to respond to change in a timely and efficient manner. That's the reason we're investing here.

So, what's a SOA architect to do? How does one design and build a SOA? Where do you start? How do you know you're done?

Truth be told, we're not good at building SOAs yet, although you can point to some early project-level successes. SOA is more of a strategic notion, one that transcends projects, and a single instance in an enterprise does not do much good until the rest of the enterprise comes on board.

SOA planning and design has received almost no press when you consider the amount of articles out there about the technology. I've taken a run at a planning methodology with my "12 Steps to SOA," (www.infoworld.com/3247) written four years ago now, and there are others out there as well including approaches from ZapThink , and even vendors such as PolarLake . However, typically these types of publications take a back seat to the ESB vs. Fabric debate, or even arguments over who invented a buzzword. Not productive.

With the development of the SOA Reference Model and the SOA Blueprints from OASIS, we are beginning to focus more on planning and design than before. However, there is much more to be done to arm those moving towards SOA with the right procedures, checklists, repeatable patterns, and approaches to insure success. We need to get real about leveraging SOAs, and that takes a lot of planning and consideration.

Are We Willing to Share?

One of the things that I've been thinking about as we create standards like SOA Blueprints is whether or not enterprises are actually willing to share their solutions.

SOA Blueprints, as you may recall, is a standard for sharing SOA solution patterns, based on your SOA requirements. In other words, you match up your needs with common solutions that are known to work with those needs.

Can't argue that it's a good idea to see what else is working before you try it yourself. That's why we purchase how-to books at the hardware store.

One of my questions concerns how the Blueprints folks plan on gathering solution patterns. I'm not sure many enterprises will be willing to give them up, and I'm not sure the vendor community is an authority on what works and what does not.

Indeed, many enterprises I'm dealing with consider their SOA a strategic technology and won't reveal what's in their recipe for fear that their competition will discover their technical secret sauce. I found this out when attempting to gather case studies for conferences, articles, and books: You get the doors slammed in your face most of the time, and when you do get to see case studies, they are simplistic and uninteresting.

So, how do we learn from the work of others, and leverage a knowledgebase such as Blueprints? I think the answer is in common patterns gathered by those implementing SOAs, this includes consultants and other service organizations. Without naming names, determine and gather general solution patterns and document the requirements. Perhaps pay a fee for each solution entered that adds value to the knowledgebase.

Then again, perhaps the lack of willingness to share our secrets will make standards such as Blueprints unviable going forward. We don't want to kiss-and-tell, even when it comes to technology.

— *Dave Linthicum*