An Introduction to Service-oriented Architecture (SOA)

– CIO **March 19, 2007**

What is Service-Oriented Architecture (SOA)?

SOA is a confusing term because it describes two very different things. The first two words describe a software development methodology. The third word, architecture, is a picture of all the software assets of a company, much as an architectural drawing is a representation of all the pieces that together form a building. Therefore, service-oriented architecture is a strategy that proclaims the intention to build all the software assets in the company using the service-oriented programming methodology.

What is a service?

Services are software chunks, or components, constructed so that they can be easily linked with other software components. The idea behind these services is simple: Technology should be expressed in chunks that business people can understand rather than as an arcane application such as ERP or CRM.

At the core of the services concept is abstraction, the idea that you can assemble software code into a chunk meaningful enough that it can be shared and reused in many different areas of the company. For example, there is a lot of software code that goes into creating an automated task such as sending a query to a credit reporting website to find out if a customer qualifies for a loan. But if the programmers at a bank can abstract all that code to a higher level—that is, take all the code that was written to perform the credit rating check and package it into a single unit called "get credit rating"—the programmers can reuse that chunk the next time the bank decides to launch a new loan product that requires the same information rather than having to write the code from scratch.

Developers create the abstraction by building a complex wrapper around the bundled code. This wrapper is an interface that describes what the chunk does and how to connect to it. It's an old concept that dates back to the 1980s, when object-oriented programming first appeared; the only difference is that today, the ambition for the size and sophistication of these software objects is far more grand.

For example, at telecom company Verizon, the service called "get CSR" (get customer service record) is a complex jumble of software actions and data extractions that uses Verizon's integration infrastructure to access more than 25 systems in as many as four data centers across the country. Before building the "get CSR" service, Verizon

developers who needed that critical lump of data would have to build links to all 25 systems—adding their own links on top of the complex web of links already hanging off the popular systems. But with the "get CSR" service sitting in a central repository on Verizon's intranet, those developers can now use the simple object access protocol (SOAP) to build a single link to the carefully crafted interface that wraps around the service. Those 25 systems immediately line up and march, sending customer information to the new application and saving developers months, even years, of development time each time they use the service.

There are many different ways to connect services, such as custom programming links or integration software from vendors, but since 2001, a set of software communication mechanisms known as web services, which are built upon the ubiquitous World Wide Web, have become an increasingly popular method for linking software components together.

What's the difference between SOA and web services?

SOA is the overarching strategy for building software applications inside a company—think of an architectural blueprint—except that in this case, the architecture calls for all the pieces of software to be built using a particular software development methodology, known as service-oriented programming. Web services, meanwhile, are a set of standard communication mechanisms built upon the World Wide Web. Web services are a linking and communications methodology. SOA is an overall IT strategy.

How do I know whether I should adopt SOA as a strategy?

Because it is an architectural strategy, SOA involves much more than merely building software. Creating an architecture based on a portfolio of services asks that CIOs make a compelling case for enterprise architecture, a centralized development methodology and a centralized staff of project managers, architects and developers. It also requires a willing CEO and executive staff to pave the way for IT to dive in to the core business processes of the company. Understanding those processes and getting buy-in on enterprise sharing are the keystones of SOA-based business transformation.

Governance is critical. For services to be reused across the company, there must be a single, centralized software development methodology so that different areas of the business don't build the same service in different ways or use linkages that are incompatible. There must be a centralized warehouse, or repository, so that developers will know where to look for services—and so IT will know who is using them. The services must be well documented so that developers will know what they are for, how to link to them and the rules for using them. (For example, some companies charge fees to use the services and create performance agreements to make sure the services work well and don't put too much load on the corporate network.)

Most companies that have progressed along the path to SOA have created a centralized architecture group to choose processes that will be service enabled and to consult with

different areas of the company to build the specific services. The centralized group also creates a convenient mechanism for governance. If all service requests have to go through the architecture group, the service development methodologies, projects and performance agreements can be more easily managed.

The companies that have had the most success with SOA so far are those that always have success with technology: big companies with big IT budgets whose business is technology-based (think telecom and financial services). They also tend to have supportive, technologically sophisticated business leaders. For companies without these advantages, SOA may not be the panacea it's being made out to be.

For smaller companies, for companies that have made big bets on integrated application suites and for companies that already have solid application integration strategies in place, SOA isn't a "when," it's an "if." CIOs need to pursue an SOA strategy carefully because the service development and architecture planning pieces of SOA are distinct but not independent—they need to be considered and executed in parallel. Services built in isolation, without taking into account the architectural and business goals of the company, may have little potential for reuse (one of SOA's most important benefits) or may fail outright.

What are the advantages of SOA?

First, put the benefits of SOA in perspective. SOA is a scythe that slices through complexity and redundancy. If your company is not large or complex—i.e., more than two primary systems that require some level of integration—it's not likely that SOA will provide much benefit. Lost in all the hype about SOA today is the fact that the development methodology itself brings no real benefit—it's the effects that it has on a complex, redundant infrastructure that bring the rewards. Architects say there is more work involved in creating a good service-oriented application than there is in doing traditional application integration. (Surveys show consistently that SOA is being used for traditional application integration at most companies.) So there is actually an extra cost being generated by SOA development up front. For there to be a benefit from that work, therefore, it must eliminate work somewhere else, because the methodology in and of itself does not create business benefit. Before considering whether SOA has benefits, you first must determine whether there are redundant, poorly integrated applications that could be consolidated or eliminated as a result of adopting SOA. If that's the case, then there are some potential benefits.

To get the entire picture of benefits being sold with SOA, you have to look at it on two levels: first, the tactical advantages of service-oriented development and second, the advantages of SOA as an overall architecture strategy.

Advantages of service-oriented development:

1. Software reuse. If the bundle of code that constitutes a service is the right size and scope (a big if, say SOA veterans), then it can be reused the next time a development

team needs that particular function for a new application that it wants to build. For example, a telecom company may have four different divisions, each with its own system for processing orders. Those systems all perform certain similar functions, such as credit checks and customer record searches. But because each system is highly integrated, none of these redundant functions can be shared. Service-oriented development gathers the code necessary to create a version of "credit check" that can be shared by all four systems. The service may be a wholly new chunk of software, or it may be a composite application consisting of code from some or all of the systems. Regardless, the composite is wrapped in a complex interface that hides the complexity of the composite. The next time developers want to create an application that requires a credit check, the developers write a simple link to the composite application. They don't have to worry about linking with the individual systems—indeed, they don't need to know anything about how the code has been bundled or where it comes from. All they need to do is build a connection to it.

In a company that constantly builds new systems that rely on similar functionality—an insurance company with many different divisions, each with slightly different products, for example, or a company that is constantly acquiring new companies—the time saved in developing, testing and integrating that same bit of software functionality can add up.

But reuse isn't assured. If developers in other parts of the company don't know that the services exist, or don't trust that they are well built, or development methodologies differ around the company, the service may languish as a one-off. Companies that get reuse have developed governance mechanisms—such as centralized development teams, a single development methodology and service repositories—to increase the chances for reuse.

But sometimes the service just isn't designed properly. Perhaps it doesn't perform enough operations to be widely applicable across the company, or perhaps it tries to do too much. Maybe the developers didn't take into consideration all the different ways that others might want to use the service in applications. SOA veterans say that sizing services properly—also known as granularity—is as much art as science and that poor granularity can dramatically reduce the possibilities for reuse. Research company Gartner estimates that only 10 percent to 40 percent of services are reused.

2. Productivity increases. If developers reuse services, that means software projects can go faster and the same development team can work on more projects. Integration becomes a lot cheaper (at least 30 percent cheaper, according to estimates by Gartner) and faster, too, taking months off development cycles for new projects. Shadman Zafar, Verizon's senior vice president for architecture and e-services, says his catalog of services lets him skip forming a project team for the development of a phone-line ordering process, because the services necessary to compose the process were already in place. "With point-to-point integration, we would have had a central project team to write the overall integration, and local teams for each of the systems we needed to integrate with. With [the phone-line process], we had a single team that was focused almost entirely on end-to-end testing," he says. That saves time and resources and improves the quality of

new applications, because testing is no longer the last hurdle of an exhausting application development process; instead, it's the focus.

3. Increased agility. Even if services will not be reused, they can offer value if they make IT systems easier to modify. At ProFlowers.com, for example, there are no redundant applications or multiple business units clamoring for services. But splitting the flower ordering process into discrete services means each component can be isolated and changed as needed to handle the spikes in demand that occur around holidays, according to ProFlowers CIO Kevin Hall. When ProFlowers had a single, monolithic application handling the process, a single change in the process or a growth in transaction volume (on, say, Valentine's Day) meant tearing apart the entire system and rebuilding it.

In the new system, a server farm responds to spikes in activity during each phase of the ordering process by transferring storage capacity to the specific service that needs it most. The system is much more predictable now, and there have been no outages since the service-enabled process was rolled out beginning in 2002, according to Hall. "Because we can scale horizontally [more servers] and vertically [splitting up services], I don't have to buy all the hardware to serve every service at its peak load," he says. "You don't have to be able to eat the elephant in one bite anymore."

Advantages of an SOA strategy:

1. Better alignment with the business. SOA is the big picture of all the business processes and flows of a company. It means business people can visualize, for the first time, how their businesses are constructed in terms of technology. When IT projects are put in terms of business activities and processes rather than complex software applications, business people can better appreciate and support IT projects. "When I said we have 18 slightly different versions of 'credit check' buried inside different applications in different agencies," says Matt Miszewski, CIO for the state of Wisconsin, "the agency heads could understand why having all those different versions was a problem, and they could support creating a single version that everyone could use." The grand vision for SOA is that when IT fully service enables the major processes of a business, business people will someday be able to take control of modifying, mixing and matching the different services together into new process combinations on their own. But that vision is still many years away.

2. A better way to sell architecture to the business (and IT). Enterprise architecture has long been the concept that dared not speak its name. Some CIOs go to great lengths to avoid using the term with their business peers for fear of scaring, alienating or simply boring them to death. Enterprise architecture has always been a big, difficult and expensive undertaking, and its ROI has often been opaque to the business. Standardizing, mapping and controlling IT assets do not make the business obviously more flexible, capable or profitable. As a result, IT architecture efforts often fail or become completely IT-centric. SOA provides the value to the business that in the old enterprise architecture was rarely more than a vague promise. Reuse, improved productivity and agility in IT and a software infrastructure tuned to specific business processes are the lures to sell an enterprise architecture effort to the business. But remember that architecture is not for

everyone. Small companies or highly decentralized companies may not be able to justify a centralized staff of project managers, architects and developers.

How do I balance the need for architecture planning in SOA with the need to prove value to the business quickly?

Architectural planning is time-consuming. Service-oriented development, drawing upon well-known programming principles and widely available technology standards (such as SOAP, HTTP and so on), can happen a lot faster. But the two need to happen in parallel, say experts. "We do development projects as needed, and then on the side we have a longer multiyear project of mapping out the processes and building enterprise-level services," says Kurt Wissner, managing director of enterprise architecture and development for American Electric Power (AEP). "People need to see the benefit of SOA pretty quickly. That's why I like the project route, because otherwise you don't have anything tangible to sell to anyone about why you're doing this."

While it would help to have the architectural plan and the process mapping in place before building the services (to improve the chances for reuse), architecture planning has no short-term payback, which can be devastating. "I tried to boil the ocean at another company and I failed," recalls Wissner. "We did a big multimillion-dollar architecture plan that duplicated what we already had. It didn't provide much value over traditional point-to-point integration, and we had nothing to show for our efforts. If you start with the entire enterprise, there are too many risks you might fail."

By taking the enterprise planning in smaller chunks at AEP, Wissner can more easily recover from setbacks. "We've had hiccups but could take corrective action because the issue wasn't that big," he says. "If you break it into simpler pieces, it's more easily digestible."

How do I know which services will provide the most value for my investment?

When in doubt, start with processes that involve customers, directly affect revenue and address a specific pain point in the business. A 2006 survey by the Business Performance Management Institute found evolving customer needs and preferences to be the top driver in business process change or the introduction of new applications, followed by competitive threats and new revenue opportunities. (Cost savings was a distant fourth.) "Externally facing applications are the ones that provide the most business value, and they have a good set of change requirements that come up very often," says Daniel Sholler, vice president of research for Gartner. "If you can improve those applications by 10 percent, it's better than improving lower-level applications by 50 percent." Of course, adds Sholler, SOA may not provide more value than, say, a good packaged application. "But if it's something you would have to build yourself anyway, you need to do it service-oriented," he says.

How will SOA affect my IT group?

If you have a decentralized company, be prepared for a struggle. SOA drives centralization. Indeed, it demands it. "You have to have someone heading it up, and you have to have one individual or small team manage the architecture," says Mike Falls, senior system engineer for Fastenal, an industrial and construction supply company. "If each team is left to itself, they may each come up with different ways of building services. You need one group, one set of research and someone to make sure the development groups are sticking to the service development methodology."

As the service portfolio grows, the development process may begin to look like an assembly line. "It becomes a factory," says AEP's Wissner. "You have these different project teams that you funnel work through, and they can grow and shrink as required."

Once the SOA factory gets ramped up, expect to add more project managers, business analysts and architects as the productivity of the developers increases, says ProFlowers' Hall. "Two developers can now do the work of six," he says. "That means the architects and project managers are running to keep up with the output of the engineers. We are probably doing 50 percent more work than we did three years ago."

Those programmers need to understand object-oriented programming and distributed applications—and that means an investment in training. According to the *CIO/Computerworld survey*, only 25 percent of respondents have the staffs they need for SOA—49 percent said they are planning or have training programs in place for current staff to bring them up to speed.
