Planning a Service-Oriented Architecture

IBM®

an internet.com Developer eBook

Planning a Service-Oriented Architecture

Over the last four decades, software architectures have attempted to deal with increasing levels of software complexity. But the level of complexity continues to increase and traditional architectures seem to be reaching the limit of their ability to deal with the problem.

At the same time, the traditional needs of IT organisations persist, such as the need to respond quickly to new requirements of the business, the need to continually reduce the cost of IT to the business, and the ability to absorb and integrate new business partners and new customer sets, to name a few. As an industry, we have gone through multiple computing architectures designed to allow fully distributed processing; programmeming languages designed to run on any platform, greatly reducing implementation schedules; and a myriad of connectivity products designed to allow better and faster integration of applications. However, the complete solution continues to elude us.

Jupiterimages

Service Oriented Architecture (SOA) is seen as the next evolutionary step in software architecture to help IT organisations meet their ever more complex set of challenges. According to IBM, the SOA market nearly doubled in 2005 to more than £2 billion and could top £7 billion by 2009.

The W3C Web Services Architecture Working Group defines SOA as a form of distributed systems architecture that is typically characterised by the following properties:

• **Logical view:** The service is an abstracted, logical view of actual programmes, databases, business processes, and the like, defined in terms of what it does, typically carrying out a business-level operation.

• **Message orientation:** The service is formally defined in terms of the messages exchanged between provider agents and requester agents, and not the properties of the agents themselves. The internal structure of an

“Service Oriented Architecture (SOA) is seen as the next evolutionary step in software architecture to help IT organisations meet their ever more complex set of challenges.”

agent -- including features such as its implementation language, process structure, and even database structure -- are deliberately abstracted away in the SOA. By using the SOA discipline, one does not and should not need to know how an agent implementing a service is constructed. A key benefit of this concerns so-called legacy systems. By avoiding any knowledge of the internal structure of an agent, one can incorporate any software component or application that can be "wrapped" in message handling code that allows it to adhere to the formal service definition.

• **Description orientation:** A service is described by machine-processable meta data. The description supports the public nature of the SOA: Only those details that are exposed to the public and important for the use of the service should be included in the description. The semantics of a service should be documented, either directly or indirectly, by its description.

• **Granularity:** Services tend to use a small number of operations with relatively large and complex messages.

• **Network orientation:** Services tend to be oriented toward use over a network, though this is not an absolute requirement.

• **Platform neutral:** Messages are sent in a platform-neutral, standardised format delivered through the interfaces. XML is the most obvious format that meets this constraint.

A service-oriented architecture allows for software systems to be designed that provide services to other applications through published and discoverable interfaces, and where the services can be invoked over a network. When you implement a service-oriented architecture using Web services technologies, you create a new way of building applications within a more powerful and flexible programming model. Development and ownership costs as well as implementation risks are reduced. SOA is an architecture and a programming model, a way of thinking about building software. ■

Requirements for a Service-Oriented Architecture

1. First and foremost, leverage existing assets. Existing systems can rarely be thrown away, and often contain within them great value to the enterprise. Strategically, the objective is to build a new architecture that will yield all the value hoped for, but tactically, the existing systems must be integrated such that, over time, they can be componentised or replaced in manageable, incremental projects.

2. Support all required types or "styles" of integration. This includes:
- User Interaction: being able to provide a single, interactive user experience
- Application Connectivity: communications layer that underlies all of the architecture
- Process Integration: choreographs applications and services
- Information Integration: federates and moves the enterprise data
- Build to Integrate: builds and deploys new applications and services.

3. Allow for incremental implementations and migration of assets. This will enable one of the most critical aspects of developing the architecture: the ability to produce incremental ROI. Countless integration projects have failed due to their complexity, cost, and unworkable implementation schedules.

4. Include a development environment that will be built around a standard component framework, promote better reuse of modules and systems, allow legacy assets to be migrated to the framework, and allow for the timely implementation of new technologies.

5. Allow implementation of new computing models -- specifically, new portal-based client models, grid computing, and on-demand computing. ■

Not Just Web Services

The success of many Web services projects has shown that the technology does in fact exist whereby you can implement a true service-oriented architecture. It lets you take another step back and not just examine your application architecture, but the basic business problems you are trying to solve. From a business perspective, it's no longer a technology problem; it is a matter of developing an application architecture and framework within which business problems can be defined and solutions can be implemented in a coherent, repeatable way.

First, though, it must be understood that Web services and a service-oriented architecture are not the same thing. Web services are a collection of technologies -- including XML, SOAP, WSDL, and UDDI -- which let you build programming solutions for specific messaging and application integration problems. Over time, you can reasonably expect these technologies to mature, and eventually be replaced with better, more efficient, or more robust ones, but for the moment, they will do. They are, at the very least, a proof of concept that SOAs can finally be implemented. So what actually does constitute a service-oriented architecture?

SOA is just that, an architecture. It is more than any particular set of technologies, such as Web services; it transcends them, and, in a perfect world, is totally independent of them. Within a business environment, a pure architectural definition of a SOA might be something like "an application architecture within which all functions are defined as independent services with well-defined invokable interfaces which can be called in defined sequences to form business processes." Note what is being said here:

1. All functions are defined as services. This includes purely business functions, business transactions composed of lower-level functions, and system service functions.

2. All services are independent. They operate as "black boxes." External components neither know nor care how they perform their function, merely that they return the expected result.

3. In the most general sense, the interfaces are invokable; that is, at an architectural level, it is irrele-

With the launch of something it calls the Retail Integration Framework (RIF), IBM gave prospective customers and its competitors a preview of what it has in store for SOA-enablement in the coming year.

RIF isn't a product but rather an enterprise software architecture, or framework, designed to help fill in the gaps between packaged software applications and legacy systems to help speed the implementation of new customer-focused strategies and technology initiatives in an SOA environment.

This particular platform, happens to focus on the retail vertical. But expect IBM to serve up a steady diet of industry-specific and even process-specific announcements through the rest of this year.

"This is kind of the direction we're heading," Tom Rosamilia, general manager of IBM's application and integration middleware group, said in an interview with InternetNews.com. "RIF is one of a string of announcements where you'll see more customisation work by industry. We're taking SOA into the industry space to help speed the time to deployment. That's what it's all about."

RIF is based on open standards and features retail-specific components, templates and patterns geared to improve the integration of business processes such as new product rollouts, cross-channel or selling, point of sale (POS) integration, store-to-enterprise integration and retailing business intelligence. It's based on open standards that integrate with IBM's WebSphere middleware.

Because most large retailers are opting for packaged applications from the likes of Microsoft, Oracle and SAP rather than building their own proprietary and more customised systems, there's a greater opportunity to leverage fragmented applications and processes and reuse them again and again across the enterprise.

For example, if one application in the marketing department can get to customer or inventory data in one part of the organisation and quickly share it with the sales team in another division of the company without having to buy or write another application, a company can start getting some tangible return on its SOA investment. But connecting the two departments and, typically, disparate software systems in a smooth fashion without disrupting the day-to-day operations remains a bit tricky.

– InternetNews.com

vant whether they are local (within the system) or remote (external to the immediate system), what interconnect scheme or protocol is used to effect the invocation, or what infrastructure components are required to make the connection. The service may be within the same application, or in a different address space within an asymmetric multiprocessor, on a completely different system within the corporate intranet, or within an application in a partner's system used in a B2B configuration.

In all this, the interface is the key, and is the focus of the calling application. It defines the required parameters and the nature of the result; thus, it defines the nature of the service, not the technology used to implement it. It is the system's responsibility to effect and manage the invocation of the service, not the calling application. This allows two critical characteristics to be realised: first, that the services are truly independent, and second, that they can be managed. Management includes many functions, including:

1. Security: authorisation of the request, encryption and decryption as required, validation, and so forth.

2. Deployment: allowing the service to be redeployed (moved) around the network for performance, redundancy for availability, or other reasons.

3. Logging: for auditing, metering, and so on.

4. Dynamic rerouting: for fail over or load balancing

5. Maintenance: management of new versions of the service ■

SOAs to Lower Legacy Costs and Free Up Manpower

Why adopt service SOA in the first place? The main reasons are eminently practical and driven by economics because SOA adoption represents a clear path for IT organisations to reduce cost and to improve operational efficiency whilst bringing in agility and competitiveness for their companies.

Let's look at the fundamental problem today. Let's take as an example a typical Fortune 500 corporation. This company might gross £20.5 billion a year and have an IT organisation with a budget of about £563 million and about 6,000 employees.

Out of that budget, a figure of merit is the breakdown between dollars spent sustaining existing services ("legacy" costs) versus dollars spent in new services. The dollars in the second category are the ones associated with growth and business innovation.

Studies from Gartner and Aberdeen put the fraction dedicated to legacy anywhere between 70 and 80 per cent, depending on the company with the lower numbers associated with the more progressive companies. Because of competitive pressures, and in spite of the economic recovery that's going into the fifth year, typical IT budgets have remained flat or growing slowly, whilst the cost of legacy tends to grow with the company. Left unchecked, the cost of legacy will overwhelm the IT budget.

One strategy to lower the cost of legacy is through outsourcing/offshoring and the ensuing cost arbitrage. The arbitrage can be geographic and taking advantage of lower salary scales in some countries, or through division of labor, where an efficient, specialised firm delivers a service such as payroll at a lower cost than the in-sourced cost.

Offshoring entails a learning curve and cost bump where the transition negotiations take place and consultants from the service provider are brought in. There is an initial cost decrease after the transition due to staffing reductions.

Although lower initially, legacy cost follows a steeper slope because salaries in Bangalore are growing much faster than salaries in the United States and Europe. Eventually, a new stasis is reached when the cost of the outsourced service plus overhead reaches parity with the in-sourced cost.

And not every service can be outsourced. Outsourcing a company's core activities might lead to loss of intellectual assets and collective knowledge that will limit an

organisation's growth potential. Outsourcing could also impact customer satisfaction, quality-of-service and employee morale that might hurt the organisation in the long term.

Technology refreshes provide an alternative. These refreshes have a deflationary effect. A strategic emphasis to aggressively adopt emerging technologies will lead to reductions in capital outlays and cost of labor. Every refresh slides the cost line down.

These transitions are not without risk and must be managed carefully. However, the rewards can be very attractive. For instance, investing in server consolidation increases a system's capability to handle bigger workloads with minimal data centre new investment.

SOA & Legacy

SOA adoption brings more fundamental change, lowering the legacy curve and its growth. Moreover, SOA does not exclude outsourcing and technology adoption. In fact, it might provide a more rational framework for their application, with clean interfaces to mix and match in-house and outsourced composable services with feedback based on business outcomes.

The adoption of SOA becomes attractive if it can be shown that it leads to a structural and permanent reduction of the current 70% legacy cost by 10 to 20 per cent. Through elimination of redundancy and breaking silos, SOAs should bring increased operational simplicity allowing dialing cost increases to a sustainable rate.

The elimination of redundancy and a 15% increase in efficiency would be equivalent to hiring 1,000 staff without actually increasing headcount, accomplished through the use of standardised platforms, simplified operating environments.

The adoption of SOA won't be without pain. Many job descriptions will change. An overall strategy requires an EA governance structure, consolidation of projects into fewer and deeper activities, and more consumption of reusable components. This transition will take time.

The cost curve will continue running its course for a time, and eventually tapers off as an increasing proportion of applications are brought up under a SOA framework.

When the processes have been institutionalised, this 1,000 headcount truly represents the resource that gets freed up from legacy work to contribute to an organisation's growth. ■

Staffing for SOA Success

Just as SOA is changing the way business and IT interrelate, it is also about to transform the types of people who work in IT.

This is because SOAs basically transform the IT department from a transaction-oriented cost-centre to a change agent. SOAs do this by making business leaner and more agile whilst simultaneously cutting costs and wringing ever more value from existing infrastructure through the reuse of past investments.

But making this happen requires looking at IT and the business it serves in more holistic ways. And this, although changing from days past, is not exactly IT's long suit.

"To make SOA really successful it can't be about just building technology, it's got to be about solving business problems," says BEA Systems' Paul Patrick, who is vice president and chief architect of its SOA infrastructure product, AquaLogic.

From a technology perspective, SOA is comparatively straightforward. If you have a mature IT department with some depth of skills, then the technological portion of your SOA implementation - wrapping data in XML, setting up SOAP calls for Web services, setting up your registry and repository, etc. - should come pretty natural.

The hard part is seeing things from a big-picture perspective and, unfortunately for those who will have to

deal with it, the politics of moving your line-of-business users used to consuming dedicated IT services towards a model that, at its heart, is all about sharing.

"The most important skill, which is going to be new to IT professionals, is they're going to have to shift their thinking and their ability from being very technology- and product-focused to becoming much more (business) process-centric," says Marianne Hedin, a programme manager at IDC.

This is where new blood may be needed to get your SOA initiative off the ground and help make it successful.

Key to this effort is having a systems' architect on an SOA team who is specifically challenged with tackling the larger, enterprise-wide issues that will inevitably come up, including:

- Who is going to have priority if shared services get bottlenecked?
- Whose budget pays for new hardware and software if the services are going to be rolled out company-wide?
- How will the SOA affect the business as a whole?
- Who's responsible for re-organising IT into a service-delivery business?

"It's in many ways a micro view of the problems sitting at Homeland Security where you've got 17 distinct agencies all trying to act as if they are one big thing, but their culture has been, 'I've got my world, you've got yours. Why should we share?' That's the hidden problem," says BEA's Patrick.

The ideal solution, is to set up a team led by the CIO that includes a business analyst who understands both IT and the business processes it supports (not any easy person to find by all accounts); a system's architect that can turn business concepts into IT-speak and ride herd over the infrastructure in order to minimise unintended consequences; and representatives from each of the company's lines of business.

This last piece of the puzzle is particularly important to a successful SOA transition, claims Chris Warner, Software AG's director of Technical Marketing and a member of SAG's SOA competency centre. Without executive buy-in and sponsorship of the changes wrought by SOA, it will most likely, at least in part, fail.

"It's kind of important that you have an executive sponsor for any new initiative," he says. "In the case of SOA, the way that works best is if you can tie that initiative to something the executives already care about. In other words, not SOA for SOA's sake."

Other team members can come and go, such as programmers to explain different concepts, but the core members should be charged with smoothing the feathers sure to be ruffled by the wholesale changes SOA can bring and making sure that whatever individual processes are served up by your SOA do not implode the entire system.

"A lot of people think SOA and they just think just technology," says BEA's Patrick. "Our experience has been that if all you do is take a technology approach to this problem … you can do a really great job and quick hit a wall." ■

Managing the Complexity of SOAs for Success

Imagine you recently completed the first successful phase of your "Enterprise SOA Master Plan." Services are distributed throughout the enterprise, linking legacy applications with your ERP and CRM applications, and are presented to your business users via a new portal interface.

The users love the ease with which they can access all this information and conduct their daily tasks. Now that they know what is possible, they want more and you are ready to give them what they want. After all, you have just proven that SOA is more than a mere buzzword.

However, no one has yet told you there is a problem.

This initial, highly successful project has introduced a new level of complexity into your IT organisation. Monolithic and heterogeneous applications running in isolation for many years have suddenly become crucial

components in a more complex organism held together by the principles of service-oriented architectures (SOAs).

The IT architects have very detailed architectural diagrams to describe it all, the developers wrote thousands of lines of code to make it all work, and the integration team has configured a complex web of message infrastructure to ensure that every message is guaranteed to be delivered to the right application. What could potentially be amiss in this picture?

Let's pose a few questions:

- How many services are currently deployed within the organisation?
- Of these services, how many are currently up and running and how many are experiencing problems?
- How well does every service perform its given task? Are your end-users waiting longer to achieve a given task?
- How stable are these services? How are some of these newly created business processes and applications affected when a crucial service (or services) malfunctions?
- Can you conduct an impact analysis to measure the potential impact of changes to an existing service or set of services?

By using the principles of SOA you have been able to solve a complex problem and by doing so introduced additional complexity into your IT infrastructure. The challenge that you now have to face is how to manage that complexity.

SOA Governance Approach

SOA Governance is a discipline that, in an ideal world, should be established early on in the SOA lifecycle. Yet, it is never too late to implement good policies, procedures, and technology to assist with the management of a freshly implemented SOA project.

Consider governance a "best practice" whereby one transforms lessons learned into sound policies, guidelines, and procedures with the aim of establishing a mature SOA ecosystem.

Here are five guidelines considered essential to good governance:

1. Establish an SOA Competency Centre
This is a cross-functional team of key people that represent different disciplines within IT. By establishing such a group you facilitate communication between the various groups, ensuring that everyone is represented and that every aspect of the SOA architecture gets proper consideration.

Furthermore this group can be instrumental in establishing organisational standards, setting guidelines and creating blueprints. This core group can also disseminate knowledge to new groups that need to adopt the SOA approach.

2. Experience is a Good Teacher
Blueprints and best practices can be very good teaching mechanisms. They are a tangible way in which everyone involved in the SOA implementation can benefit from the work that has been done before.

Too many times development teams have to "reinvent the wheel" in order to solve a particular problem. Identify and document those solutions that work well and turn them into patterns that can be used repeatedly.

By making a particular approach repeatable you are able to reduce risk, shorten development time and also decrease the cost associated with the overall project. Likewise, it may be just as important to document those approaches that don't work well or which may have failed. By documenting these so-called anti-patterns one is able to prevent mistakes from being repeated.

3. Use an SOA Repository
Services are by definition reusable and self-describing, but don't be deceived by this oversimplified view.

If services are not properly documented and made accessible in a consistent manner, they could become just as unusable as some of those legacy applications.

Accomplishing such simple tasks as establishing the

exact nature of a service, obtaining the latest service description, or determining the owner of a particular service may become quite a daunting task if services management infrastructure is not put into place. This is where an SOA repository can play an important role. It can become the focal point where all services can be documented, searched, and accessed.

4. Monitor Quality of Service (QoS)
Bring predictability to your SOA by implementing service management facilities. What you need is the capability to maintain end-to-end visibility into your services infrastructure via the ability to define SLAs (service level agreements), measure services operation and response levels, analyse service usage and resource utilisation patterns, and proactively be alerted of potential and imminent service failures.

Stability is the hallmark of a mature system and through the effective management of your deployed services (and related infrastructure) you can bring a great measure of stability to your SOA.

5. Manage the SOA Lifecycle
Create a disciplined and well-managed SOA lifecycle. In all other areas of IT this kind of discipline is well established and maintained. SOA should be no different.

Implement tools and procedures that will help manage the services development, testing, quality control, deployment, and maintenance processes.

As part of the lifecycle, create a services approval process, ensuring that services are tested for standards compliance and services are put through a comprehensive vulnerability, interoperability, and regression testing process. Implementing these measures will assist in elevating the overall maturity and robustness of your SOA implementation.

Implementing an SOA entails more than just implementing a few services or using an ESB (enterprise service bus). SOAs may start out simple, but they inherently introduce complexity into the IT organisation. This is mainly due to the nature of the problems that we are attempting to solve.

Acknowledging the complex nature of SOA and taking the appropriate steps to manage this complexity is a crucial step toward the successful development, implementation, and maintenance of an SOA. The best advice is to plan for SOA success. Plan to be hugely successful and in preparation, consider how you will govern the next generation of business applications. ■

This content was adapted from EarthWeb's Developer.com and CIO Update Web sites. Contributors: Theo Beack, Allen Bernard, Enrique Castro-Leon, Arulazi Dhesiaseelan, Kishore Channabasavaiah, Kerrie Holley, Edward M. Tuggle, Jr.