SOA From a Management Perspective: Part One

Featured Author - Joe Strub - January 5, 2007

Part I | Part II

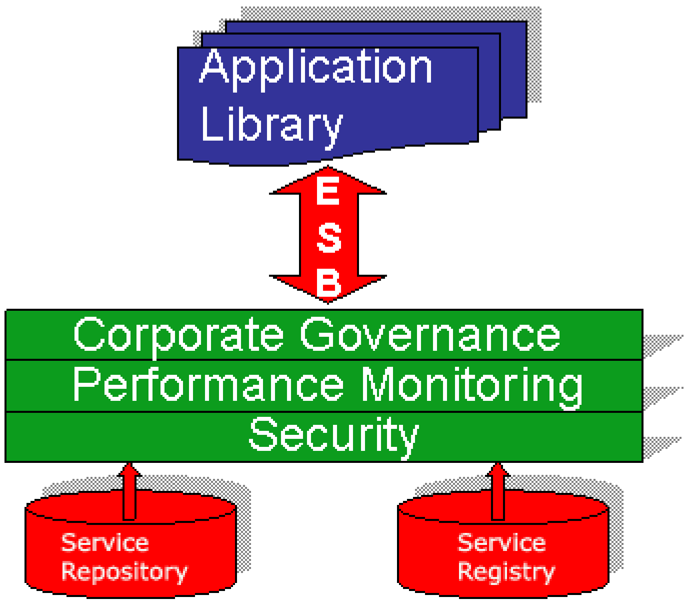
We have been hearing about *service-oriented architecture* (SOA) for some time. Now, major software players are starting to lay out their plans and strategies. Some are even willing to assign delivery dates. As a company, you cannot ignore SOA, since we are constantly told that, from a software perspective, it is the best thing since sliced bread or, to use an updated analogy, the TV remote control. Regardless of your views, *chief information officers* (CIOs) need to outline their plans and be prepared to respond to executive management's question: "What are we doing?" This note sounds the alert, raises the flag, or fires the warning flare as to why converting to SOA does not appear to be an easy transition and cannot be accomplished with smoke and mirrors. Accordingly, we will look at the basics of SOA; the rollout plans of major software vendors; the benefits of SOA; concerns about SOA; and why the implementation of SOA won't be easy.

This is Part One of a two-part research note. Part Two addresses concerns about SOA and how your organization can migrate to an SOA environment.

What is SOA?

SOA is a collection of services or groups of components that perform business processes such as credit verification, currency conversion, or inventory availability. SOA employs an architectural style for building applications by combining loosely coupled and interoperable services. By being loosely coupled, an application does not have to understand or even know the technical details of a service to call it. As a result, SOA attempts to deliver platform independence, and is not tied to a specific technology.

Examples of SOA can be found in our everyday lives. A common example is a DVD player. The player offers the service of playing a DVD. You can play multiple DVDs in the player. You can even play the same DVD in another player, but the sound quality may not be the same. SOA, however, should not be confused with *object-oriented programming* (OOP). Following our DVD example, in OOP the DVD would come with its own player, not to be separated. This diminishes one of the primary advantages of SOA, namely reusability. To understand the evolution of SOA, see research note *Architecture Evolution: From Mainframes to Service-oriented Architecture.*



Being loosely coupled also helps to screen some of the technical complexity of programming, a potentially big boost for productivity. For example, you do not need to know how a credit check is performed to complete a customer's order. You just need to know what information, such as customer ID and order amount, the credit checking service needs to return an approval or rejection. The process is similar to when televisions were built with individual electronic components and repair meant replacing a component and not the entire set.

With SOA comes new terms and concepts, or old concepts with a new lexicon, both of which mean some difficult decisions lie ahead. With the use of services you can expect a lot of messaging traffic. Accordingly, you will need technology to manage this traffic and its flow on the information highway. An *enterprise service bus* (ESB) facilitates the connection of legacy systems to services. It also transforms and routes traffic. ESBs are particularly effective for long-running processes, invoking multiple services such as purchasing, which can encompass item look-up, pricing, discount terms, and more. After being developed and tested, services must live somewhere. Typically, services are published in registries or directories while being stored in repositories. This combined infrastructure controls secured access, specifies the input parameters, and enforces run-time performance parameters. Relative to performance and after services enjoy wider and wider acceptance, reporting and alerts are needed to ensure that applications are taking full advantage of SOA. Other difficult choices such as development platform, integration with web services, and testing and debugging protocols remain, and must coexist with your current technology and network topology.

Major vendors are starting to lay out their visions for SOA. Based on its **NetWeaver** development and integration platform (see research note *Multipurpose SAP NetWeaver*), **SAP**'s approach is to deliver narrowly defined models or packages rather than release large-scale updates of closely interlinked components. By providing business process models, SAP provides the means of getting you up and running quickly, assuming that the models can fit within the constraints of your business.

Attempting to merge the software code resulting from its recent acquisition of **PeopleSoft**, **Oracle**'s **Project Fusion** endeavors to provide a more open environment. Accordingly, Oracle's approach is to provide tools to model your business processes. These tools include a business process development platform and middleware and database components, which are open to third party vendors. This *business process management* (BPM) approach can deliver a more open framework, resulting in components tailored to your environment. So, while SAP's approach could simplify and accelerate the overall process of implementing SOA, Oracle's plan may provide greater adaptability of the unique aspects that make a company successful.

Referring to the "real approach" to SOA, **Microsoft** advocates a more incremental method, using advancements in **.Net Framework**, **SharePoint**, **2007 Microsoft Office System**,

Exchange Server, and **Vista** (see research note *Subtle (or Not-so-subtle) Nuances of Microsoft .NET Enablement*). Unlike the enterprise infrastructure-centric approach, Microsoft touts a wave approach to deliver SOA interoperability gradually to the Microsoft Dynamics product lines. **Microsoft Dynamics**, formerly known as **Microsoft Business Solutions**, includes **Axapta**, **Great Plains**, **Navision**, and **Solomon**. In so doing, some features will be available now instead of waiting for the full rollout. Originally known as **Project Green**, Microsoft has committed to an initial phase called **Wave 1**, which is nearing completion. It is expected to achieve a common look and feel throughout Microsoft Dynamics. Obviously, this is where **Office 2007** and Vista will set the tone. Expect to see left-pane navigation bars for easy access, top-page trails for traversing back to a previous point of reference, and user ribbons, which will replace the traditional menus and toolbars with a set of tabs of common and most relevant commands. **Wave 2** is expected for release in 2008 or 2009 as product lines continue to move away from coding to model-based development using Visual Studio .Net tools and languages. Once this transition is complete, Microsoft can consider merging product lines. Microsoft shares two potential obstacles with Oracle. First is seamlessly integrating acquired software packages to present a consistent and familiar theme. Second is adopting a model building capability as opposed to delivering the models, potentially sacrificing delivery speed for the sake of flexibility. While Microsoft's architecture vision appears to be clearer due to our familiarity with and the release of Office and Vista, the starting point for merging product lines is vague, particularly when compared to SAP and Oracle.

Given SAP's increased interest in penetrating the *small and medium enterprise* (SME) market, its approach of providing process models ready for deployment makes a lot of sense. For companies with *information technology* (IT) resources, availability of Oracle's modeling tools allows enterprises to grow their own services, customized to their business requirements. However, Microsoft's similar approach is somewhat baffling given its predominance in the SME space.

Vendor plans are, at best, sketchy, as more vendors are jumping onto the SOA bandwagon in an attempt to generate new software revenue streams. After digesting the over-hyped Y2K phenomenon, many companies are adopting a reasonable wait-and-see attitude before taking another bite of the technology apple. Can you blame them?

What Are the Benefits of SOA?

Be assured that the benefits of SOA are significant and achievable, and deserve careful evaluation and analysis. Reviewing some of the more substantial benefits justifies this cautious stance.

Service Reusability

Nowadays programs are rarely written from scratch. Typically, you start with a familiar program that most emulates the functionality you are trying to re-create, or that provides structural consistency. Now take this philosophy and incorporate built-in services and routines. Service reuse is the real power of SOA. Companies are finding that development time is significantly reduced, program logic flow greatly simplified, and duplication of effort eliminated. A well-maintained, tested service can be reused across the entire enterprise. Recent surveys indicate that service reusability is the most often cited benefit delivered by SOA. This should not be surprising. The concept of reusability has been the nirvana that IT groups have been trying to attain since the first IF…THEN…ELSE statement was coded. Whether SOA will achieve this objective remains to be seen. Remember, IT can also stand for independent and temperamental.

Agility

Recall that one of the problems associated with contending with Y2K was interpreting the two-digit year designation. Finding every date routine in every program was considered a nightmare, warranting a complete replacement of legacy systems. Suppose that date handling was a called service that ensured correct formatting and calculation of periods of time. Y2K would not have been the worrisome beast that everyone thought it was. The analogy can be applied to more frequently changed business processes. Want to attract more customers by changing the pricing policy? Is bad debt increasing at an alarming rate, making you want to tighten credit checking procedures? If these processes were being supplied as centrally located services, accommodation of these business changes can be quick and immediate. If done correctly, SOA gives new meaning to agility and the concept of turning on a dime.

Corporate Governance and Security

Unfortunately, this next benefit bemoans the sad state of IT, but is necessary for SOA to be effective, namely "Tell them and they will comply." Corporate governance promotes compliance with policies and standards and ensures that everyone is marching to the same beat. Developers can be persuaded/encouraged/forced to use repository-based services if they ever want their applications to see the light of day. As a result, SOA can facilitate compliance with the US Sarbanes-Oxley (SOX) law and industry-specific regulations, thereby making it increasingly important to give sufficient notice to SOA governance. And, if personnel modeling business processes understand the need of using services, the need to provide secured access to these services is a natural follow-on. But this can be a double-edged sword. If you require the use of services, a communication channel must be established to let everyone know and find what is available for reuse. While registries and directories can help facilitate this communication, proactive methods must be added to supplement these passive tools.

Scalability and Manageability

There is some definite overlap with scalability and manageability relative to the previously mentioned benefits. However, these two benefits deserve special mention primarily because they are central to a well run and efficient IT organization. In an SOA environment, a change to a vendor-supplied *application programming interface* (API) means that the upgrade needs to be applied in one place only. Providing a central location for the services should make their management easier and more consistent. Furthermore, as services become popular in an organization, their rollout should be become more routine and straightforward.

About the Author

Joseph J. Strub has extensive experience as a senior project manager and consultant for the planning and execution of enterprise resource planning (ERP) projects for manufacturing and distribution systems, for large to medium companies in the retail, food and beverage, chemical, and consumer-packaged goods (CPG) process industries. He has developed marketing and communication programs for IT organizations and consulted on off-shore, outsourcing opportunities for multinational companies. Additionally, Strub was a consultant and information systems auditor with PricewaterhouseCoopers, and an applications development and support manager for several Fortune 100 companies. Currently, Strub is an independent consultant.

He can be contacted at JoeStrub@WriteTechnologyPlus.com.

Related Book

SOA From a Management Perspective: Part Two

Featured Author - Joseph J. Strub - January 8, 2007

Part I | Part II

Concerns About SOA

This is Part Two of a two-part note. Part One provides a basic understanding of SOA, the rollout plans for major software vendors, and the benefits of SOA.

Back in the day when procedures and subroutines were all the rage, there was a performance concern: Did the use of subroutines, with all of its branching back and forth and passing of data, degrade performance as opposed to duplicating the code in stream? The same concern with message traffic in *service-oriented architecture* (SOA) is being voiced by some *information technology* (IT) shops. Vendors downplay this concern, explaining that today's processing speeds more than compensate for service calls and use of generalized routines. The problem is that you need to compare apples with apples. Improved processor speeds will benefit both SOA and non-SOA environments. Assurances must be made that the time is takes to process a complete transaction in an SOA environment is not significantly slower than what companies are experiencing today. This is why previously mentioned performance alerts are critical to the success of SOA.

The prevailing IT culture may inhibit SOA. Traditionally, IT shops are measured by the lines of code generated. This is contrary to the reusability benefit of SOA. The paradigm needs to be shifted to reward rapid and agile implementations rather than the size and complexity of the programs.

The cost of implementing SOA will not be cheap. In addition to reengineering your existing architecture technology, SOA will require a significant investment in human capital so that business analysts can define finite business processes; systems analysts can convert these processes into specifications; and systems engineers and programmers can translate everything into functional and manageable services.

Lacking at this stage of its deployment, but quickly catching up, are third-party tools to assist in the monitoring and improving of performance in an SOA-defined environment. Early adopters of SOA must either construct their own resources or work jointly with vendors to test beta versions of tools. This is why, for example, pioneering companies are using everything from **Excel** spreadsheets to simple databases as repositories for services. By using in-house tools, companies have the opportunity to survey the technology landscape for the best solution for their infrastructure.

Although there are differing opinions on the subject, many consider Web services as a mandatory element of SOA. For those sharing this viewpoint, Web services must be implemented properly to create a solid SOA environment. While some companies may be taking a wait-and-see approach, now would be a good time to get your Web services house in order.

As SOA implementations start to proliferate across an enterprise, traditional, manual governance solutions, which include design walk-thru's, conference room pilots, and after-the-fact reporting, may be inadequate. Consequently, elements of the system life cycle approach will need to be reviewed to incorporate a more proactive governance strategy.

Staid thinking that your application lifecycle has served your organization well for many years may result in missing the SOA mark and achieving its expected benefits.

How Can You Get to SOA?

Now you're excited about SOA. You have bought into the benefits and can't wait to get started. Not so fast!

SOA is a tough sell to company executives. SOA helps your company be more flexible and agile. However, it is difficult to make a business case or develop cost savings based on flexibility and agility. Except in the rare cases where responsiveness can mean the winning or losing market share, justification for a change in technology is built on solving business problems or gaining a competitive advantage. Remember that Y2K brought about major systems upgrades not because of a more effective use of technology, but rather due to its threat of shutting down businesses.

As has been stated above, service reusability is a major drawing point of SOA. Making this concept a reality is easier said than done. We were unable to do it for objects and we couldn't make it happen for components. What makes us think that SOA will be any different? Oh, yea—corporate governance will do the trick. Even if corporate governance could enforce SOA policy, we are adding another layer of management and probably staff. Great—now we are responsible for increased labor expenses and overhead.

SOA can be explained as enterprise computing, reusable services, and a faster construction methodology. While SOA may be a straightforward, understandable concept, it requires a steep learning curve for companies wanting to embrace this technology and reap the expected benefits. Traditional developers, who take the big picture approach toward application integration, will need to break flows and processes into smaller bites if reusability is to flourish. Developing a core group of IT professionals to spearhead the effort will not only limit your exposure to minor setbacks, but will also create a cadre of SOA disciples to spread the word.

Another paradigm that needs to be shifted is the traditional image of the IT professional. The image of a "bits-and-bytes" hermit that lives somewhere in a technology cave and speaks in three letter acronyms won't cut it. If IT professionals are to effectively work side by side with their business counterparts to model finite processes, both groups must speak a common language. This language must be in the tongue of the end user. While you may say that your company has already overcome this hurdle, try corroborating this with the user community. Surely *enterprise service buses* (ESBs), repositories, and directories are important to the eventual success of SOA; they need not concern the user and will only muddy the water.

But the million dollar question is, "How do we implement SOA?" Some vendors would suggest a gradual deployment. But this is tantamount to straddling a boat with one foot on the boat (the SOA environment) and other on dry land (the current software infrastructure). As the boat drifts further and further away, your resources are stretched to their limits. You fall in, and your current and future states start to flounder. While this can be done, you had better be a good swimmer and have someone on staff proficient in CPR—that is, core program resuscitation.

Depending on the age of your existing applications, internal and external programming interfaces may be difficult to find with a single program, much less than in an enterprise's entire application library. To use a service, the current interfaces must be ripped out or neutralized. Think about it. Take the credit checking/updating service, for example. This could be used in standard order taking, returns processing, quote generation, onetime orders, promotional sales, discounts taken, and who knows where else. Now go through each of these programs or modules, trace the interface logic, and determine what changes have to be made. Not an easy task. While you may have access to the source code, you may not have the technical resources to play this game of technical hide'n'seek. While the software vendor can offer assistance, can you afford it?

While we are talking about technical resources, to take advantage of SOA's agility, the services may have to be modified or, dare we say, enhanced. Again, unless you have the technical resources to perform this type of modification, you may be at the mercy of the software vendor. If you argue the agility case for SOA, make sure that your company can support it or, at least, pay for it.

Unfortunately, SOA may be Y2KAOA—that is, Y2K All Over Again. SOA is not a case of migration; it is a case of rip out and tear down to the technology foundation, and replace on an enterprise-wide basis. While large enterprises may have the luxury of choices, *small and medium enterprises* (SMEs) certainly do not. To use an analogy, SOA is similar to the era when automatic transmission was first offered in cars. It made driving easier and less stressful. But you probably did not take your stick shift car to your local mechanic and have the manual transmission replaced. While your current car may not have been the latest and greatest in automotive technology, it was functional and served a valuable purpose. Similarly, your current enterprise software is most likely fulfilling its stated mission. Just because new technology is being touted by software vendors does not mean you are going to scrap what you have. No, you are going to wait until you can establish a business case for replacing the software such as faster processing to keep up with increased orders; better pricing algorithms reflective of your company's practices; or improved tools for financial analysis and decision-making. When you can make the business case, you will want to determine what software solutions are built on the SOA technology and incorporate this advantage into your evaluation and selection.

Summary

Some of you may be saying, "Can't this author make up his mind? Is he in favor of SOA or not?" As we have seen, there is little doubt that SOA could offer real, tangible benefits in terms of reusable services and faster application deployment. Whether we can achieve these benefits this time around remains to be seen. Regardless, SOA is not a technology that can be implemented in an incremental or piecemeal fashion. You can't just get your big toe wet; it requires a complete submersion. When you can make a convincing argument to your management to upgrade your enterprise portfolio, SOA may be mature enough to be considered a reasonable alternative. Right now *chief information officers* (CIOs) would best be served by watching from the sidelines and tracking the SOA developments while keeping their management informed. Like the first year model of a new car line, it can be unwise to implement the first version of software using SOA. Finally, what is particularly interesting is that few vendors are even offering an SOA-based solution, yet we are already talking about *SOA 2.0 and advanced SOA*.

About the Author

Joseph J. Strub has extensive experience as a senior project manager and consultant for the planning and execution of enterprise resource planning (ERP) projects for manufacturing and distribution systems, for large to medium companies in the retail, food and beverage, chemical, and consumer-packaged goods (CPG) process industries. He has developed marketing and communication programs for IT organizations and consulted on off-shore, outsourcing opportunities for multinational companies. Additionally, Strub was a consultant and information systems auditor with PricewaterhouseCoopers, and an applications development and support manager for several Fortune 100 companies. Currently, Strub is an independent consultant.

He can be contacted at JoeStrub@WriteTechnologyPlus.com.

Related Book