

IT Executive Exchange

Service-Oriented Architecture (SOA)

Executive Summary

The application of SOA is now becoming a reality in many firms. It holds out the promise of much greater business process flexibility and visibility, and increases in efficiency by reusing work. Cliff Donoughe of Accenture outlined the key architectural components of SOA, and summarized some key advantages that he has seen firms achieve using it. With the commitment of Oracle/Peoplesoft and SAP to re-architect their products as sets of services, we are poised to enter a new era of software development in which services, either internally developed or provided externally, will play a major role. Simple steps to get started were outlined: writing new programs as web services themselves, starting with technical services that can benefit the IT function as a means of learning, and creating Centers of Excellence to promote SOA use in the enterprise. Mastering SOA can take 6-18 months, depending on firm size, and may require substantial investments in hardware, software, and training. However, strong ROIs have been observed. The consensus of the group was that SOA is an important emerging core technology.

The IT Executive Exchange (ITEE) is a group of IT Executives and College of Business Administration professors at The University of Akron that meets about every six weeks to discuss pressing and leading edge IT issues faced by IT executives. The purpose of this forum is to have a healthy exchange of ideas that will be useful to all attendees. It is sponsored by the Center for Information Technologies and eBusiness (CITe) of The University of Akron's College of Business Administration. For previous topics and summaries, refer to <http://cite.uakron.edu>

This summary was prepared by Prof. William McHenry, College of Business Administration, The University of Akron.

Complete Summary of the Session

We were joined by Cliff Donoughe of Accenture. Cliff is a Technology Architecture specialist who has been working with Accenture since 1993, and working on architecture integration questions since 1996.

Donoughe began by explaining the various justifications for going to SOA (Service-Oriented Architecture). The goal is to reduce expenditures on software maintenance, operation, and development. Currently firms spend about 50% of their IT budgets to run existing systems, 20% on maintenance of old systems, and 30% on new systems. With SOA, it may be possible to reduce the portion spent on running existing systems from 50% to 35%, making it possible to allocate another 15% to new systems. It is doubtful that IT budgets will go up much in general in the near future—IT departments are constantly being asked to do more with less. Firms are often locked into existing architectures and legacy systems, but SOA provides a means of retaining the value of these systems without having to re-architect them.

SOA is not a new idea. The original attempts at integration (10-12 years ago) took place in the mainframe environment. But because they relied on proprietary technologies, they did not succeed. Then client-server computing came along. It was a great idea, but integration complexities made it very expensive. Developers found themselves building point-to-point integration solutions. While EAI (Enterprise Application Integration) tools became available and more then more robust, enterprises still struggled with point to point solutions. Severe strain could be placed on bottlenecks between multiple clients and single servers, leading to the development of clusters of back-end servers and transaction processing servers in the middle to balance loads. We also learned from earlier attempts at reusable software that services cannot be too small. Too many components leads to management nightmares. Ultimately wrapping a set of functions as a service and communicating with the service through messages became a leading means of grappling with the complexity of so many pieces interacting with each other. SOA attempts to be a complete set of standards and a philosophy of how to tie together multiple sets of functionality as services. One result is that these services can be reused within the enterprise. A second result is that services can be accessed from outside providers.

Examples from leading users:

- A financial firm rolled off products “much faster than they ever imagined” by taking functions from programs they already had, recasting them as services, and making them accessible on various new platforms such as the Web, mobile devices, voice, etc.
- A firm was able to start taking micro-payments by using a web service
- A communication firm used SOA to deploy a rules-based engine that allowed them to have a lot more flexibility with pricing policies. Changes used to take six weeks. Now, once they have proposed new rules and gotten them approved, they can be integrated, tested and deployed in a matter of days.
- Another firm used SOA to take applications that were in call centers and deploy them to the Web. They reduced call center traffic at first by 10%, now it has gone down 20%. They now have the flexibility to deploy the same applications across multiple channels.

- A financial firm was able to “divide up” legacy financial systems into smaller services that gave more flexibility
- An insurance carrier had a lot of legacy code for claims inquiry. Before SOA the APIs they had written for the mainframe were so complex that they had major headaches every time they wanted to add a new channel. They have exposed the APIs as a set of web services once, and now they can integrate any desired channel with little extra effort. Now they can use specific functions separately, such as rating a prospective customer or starting a policy. They can now orchestrate different flows for customers of different value.
- A tax office with offices in about a dozen states has numerous variations and requirements, including a voice response tax return submission function. SOA allowed them to reduce the complexity of deploying numerous closely-related functions across states.
- The U.S Air Force wanted to take what was being drawn on an “electronic” whiteboard, harvesting the individual portions of the plans and sending them to the right systems. They could orchestrate the order in which things were proposed, which sped up the process. This has been tested in the theater of war in Iraq.
- There are a lot of examples of retailers who have found it easier to roll out new locations using SOA, because the interface structure is the same.

A simple example of a web-based service is MapQuest. Numerous vendors wish to provide instructions on how to find their businesses. All they have to do is reach an agreement with MapQuest, plug in the interface on their web pages, and the consumer can readily get the required information. Given the prohibitive cost of creating the MapQuest geographical database, there is no other way to envision such a service being mounted, and it would only be cost effective for an individual when done in this fashion. Although this is an extreme case, major software vendors such as SAP and Oracle are beginning to reposition their ERP systems as sets of services. When this comes to fruition, this will be a major paradigm shift in programming.

Donoughe then explained the underlying technologies that comprise SOA. The Enterprise Service Bus (ESB) was described by another participant later as “load-balancing on steroids.” Its purpose is to provide the environment in which all of the messages between the services providers and service uses can be exchanged. We can think of it like a highly sophisticated central post office that provides the ability to receive, store, sort, prioritize, and send messages of different sizes. Furthermore, ESB software allows you to set this up, monitor its performance, and tune it for your environment. The ESB registers the subscribers to each service. It does not have to be implemented on one server, and in reality looks like a distributed system. So it is not a hub and spoke model as EAI systems tended to be.

The next level is BPO, or Business Process Orchestration. This is about binding together sets of services so that, from the user’s point of view, it looks like he or she is using a single, unified system. Some firms have been able to do this using workflow tools that they already had in place, while others have purchased software specifically for this step. The highest level is BAM, or Business Activity Monitoring. BAM makes it possible to keep track of the performance of each business process. One of the nice things about these software packages, Donoughe said, is

that they can incorporate some steps that are manual (i.e. not computerized). You can measure how long these steps are taking (e.g. how long does it take to fill an order).

Peering even further into the sordid underbelly of SOA, Donoughe reeled off a series of standards that have been developed or are under development to support this ideology of systems architecture. Among these are XML, SOAP (Simple Object Access Protocol, for communication between individual objects), WSDL (Web Services Description Language, for publishing information about objects), and UDDI (Universal Description and Discovery Interface, important for describing and finding external services). There are a lot of other standards that he showed very quickly on the screen, but did not talk about in detail. Donoughe emphasized that because of the existence of these standards, firms need not reinvent the wheel on any of this stuff. The field is becoming more mature. Major players such as IBM are solidly behind it.

Oracle/PeopleSoft and SAP have committed to re-creating their software as a set of services. They are progressing much faster now than might have been expected, which is engendering a lot of excitement in the industry. SAP is particularly important. Currently functions are divided up into transactions. Customization can be done using the ABAP programming language, but users would like to avoid customization like the plague. A different way of envisioning the use of SAP is to think of it as a set of services, and then use Business Process Orchestration to bind together those services that are needed to comprise the necessary functionality for a given firm. Absent this capability, it is still possible to reach into SAP using ABAP to create a web service internally that can then be accessed by the end users and bound up with other services as needed. If and when SAP provides this functionality as a general method, the possibilities will be much greater. SAP will be able to offer best practices, but at the same time ensure that enterprises can use those best practices in unique ways. One of our participants wondered how SAP will then price its services. It might no longer be necessary to buy nearly as many “seats” (licenses for unlimited use on a per-user basis). Instead one could buy a small number of seats for the power users and expose the rest of the users to only the functionality they needed through web services. Donoughe assured us that SAP will devise an economic model that will keep their revenues at least at current levels.

And so, we got a glimpse of what future architectures may look like. One of our participants noted that they have had to present certain data to their customers who are using Ariba to manage their supply chains. When this was only one customer, the complexity of creating a specific interface to the internal systems to expose this data to the Ariba-using customer was not too much work. A second customer started using Ariba, and then a third. Each of the implementations had to be slightly different, but there was considerable overlap. At this point it started to become clear that this firm would be better off creating a Web service that any of its Ariba-using customers could use to see this data.

Another participant noted that they were on the other side of such a relationship at his previous firm. They (call them Firm X) were serving as an intermediary (systems integrator) and needed to show their customers data about the availability of systems they were buying for their customers from IBM. Initially they had created a duplicate of this information on their own system, which required a considerable amount of programming to get the file formats correct. This could be hours out of date. Then IBM made a web service available to “reach in” to its

servers and provide this data. Firm X was able to slip this web service in to their web page, so that from their customers' point of view, the data was coming from them. In reality it was being pulled from the IBM systems in real time. It was emphasized that Firm X negotiated a service-level agreement for performance and availability with IBM about the availability of the data through the web service. This ensured that Firm X could give its customers the required level of service.

Getting started involves the following steps:

- 1) Mastering the ability to write programs so that they function as web services. This may be as simple as using JAVA and JMS (Java Messaging Service). JAVA objects then send and receive the needed data off a JMS queue. Other technologies such as .NET and J2EE can also be used to invoke services.
- 2) Start with a technical service that will be a great use internally and allow your IT group to master the technology without getting into end-user issues. An example is error-handling. All programs have to have it. Why reinvent the wheel with every program? While it is universally agreed that IS projects without business side sponsorship are doomed, this may be one exception. Functional services provide the biggest payoff, but are harder to implement at first. Leveraging Internet services can also be a means of getting started to learn the technology. An example of a function is a function that will create an order. Once this is a service, it can be called from an internal program, a portal, or the Internet.
- 3) It is hard to get companies to switch over to the mentality of reuse as opposed to also building anew. Donoughe said that some of his clients are using a Center of Excellence approach to spread the technology. The idea of having one central repository of all web services in the firm was rejected as being unrealistic and leading to bottleneck problems. The Center of Excellence works with the business units to ensure that components are reused and that newly built functionality is exposed as web services for everyone's use.

It is not necessary to buy ESB, BPO, or BAM tools to get started. Another example was given from the University of Akron, which is using an outside service to collect payments from students. CASHNET does all the collecting. When the students have paid in full, their payments are sent back to the university using a web service. This was fairly easy to set up.

Questions were raised about the impact of SOA on the audit model. For example, say that SOA was used to display to the user a consolidated view of four wage tables. At the data level, whatever audit controls were in place for those tables would still be used. At the systems access level, it would be necessary to audit the web service, since it was touching the data on its way to the end user. So if the web service involves something that would be auditable under Gramm-Leach Bliley, HIPPA, Sarbannes-Oxley, or another law or standard, then the provider of the service has to be audited if that provider is not your own firm. Furthermore, if that external provider has to be audited, your firm will bear the cost. So, those providers are now being audited and offering a form called the SAS-70, which is an assurance of this audit. Your firm can then include the SAS-70 with the audit of your firm to cover the portion of the services performed outside your firm. CASHNET offers a SAS-70 audit form.

We wondered how this might be realized if the web service providers are offshore. IBM has recently announced that it will consolidate its worldwide management of web services in its Indian Research Center in Bangalore, India.¹

What about the ROI of SOA? Donoughe noted that firms often struggle with ROI calculations when the reductions are not obvious (obvious reductions come in the form of reduced head count, for example). So how do you measure the impact of NOT having to create code? Gartner says that the reductions can be 20%. It is important to note up front that there is overhead involved in using this technique when creating programs to the SOA standards. The costs incurred only pay dividends if and when the components that have been created can be reused in subsequent projects. However, it can also be argued that creating programs as SOA objects from the start is becoming a best practice, and will yield benefits within the software development process even if the components are not reused. Donoughe also made it clear that there can be additional hardware costs to deal with the overhead of having programs communicate via messages. XML is known to be verbose. Additional costs can come for bandwidth and for processing. Hitting systems real-time can be a real resource hog, and may require more hardware. In his view this is a small price to pay for the flexibility that SOA brings.

One of our participants recommended that firms start using SOA when possible, all other things being equal. Little by little more and more of the functionality will operate this way, until a major transition is effected. Donoughe said that for medium-sized clients, it was taking 12-18 months to transition to SOA. For large clients that could invest more resources, it was taking 6-9 months. In the case of another participant, they found that they could deliver functionality in six months based on putting wrappers around legacy systems and using them as web services. The alternative would have been costly legacy system modifications or replacement that would have taken three years. Plus the wrapper that was created could then be used for other applications.

A possible pitfall with SOA is that the flexibility it provides will set user expectations too high. One of our participants recounted a situation in which they had created an advanced portion of part of a system for them which was highly customized to their needs. Replicating all this functionality in the main version of the system and keeping it as they moved forward would be difficult. Yes, SOA made it possible to have that much flexibility, but it seemed to be a double-edged sword. The clients tended to think that anything was possible. So expectations had to be managed carefully. Another challenge is ensuring that the web service APIs offered to customers are reasonably stable. One of our participant firms recounted this as an important issue.

Next Meeting: April 28, 1:30-3:00 (before general CITE Board meeting and reception with students)

Topic: Data Management, including issues of Data Warehousing, Business Intelligence, Data Quality, Storage and Security

¹ John Ribeiro, "IBM sets up SOA hub in India," *Network World*, <http://www.networkworld.com/news/2006/030706-ibmsoa.html>, 3/7/2006. "The Global Business Solutions Center will be in charge of enhancing IBM's portfolio of replicable components across 17 industries. IBM now has about 50 such components across those industries. The company plans to invest \$200 million a year to develop and market replicable components."