Global Knowledge®

Expert Reference Series of White Papers

How to Succeed at Service-Oriented Architecture

How to Succeed at Service-Oriented Architecture

Dr. Adrian M. Rossi, founder and CEO of ARC, ITIL®-v3-Certified Applications and Solutions Architect, Consultant, and Global Knowledge Instructor

Introduction

Around 1996, Gartner analysts Roy W. Schulte and Yefim V. Natis published the first series of reports around the newly introduced concept and terminology of Service-Oriented Architecture (SOA). It was not really clear back then what SOA really meant in practice for organizations, and after more than a decade, it seems it is still not clear to many of them what SOA is all about, its value proposition, and, more importantly, why they should care about it and adopt it in their organizations.

Even worse, while the rest of us were trying to find answers to these questions, in the meantime, vendors were doing their best (and still do so) to coopt the term and repackage it as a product that they could sell to unsuspecting organizations. The pitch was simple: purchase this product, and you will have an SOA! Quite a few organizations bought into this sales pitch from leading vendors and subsequently proceeded to fail miserably to adopt SOA, inevitably wasting millions of dollars and countless man-hours and creating severe disruptions within their enterprises, with little to no tangible results. In some extreme cases, the whole endeavor was abandoned to the point where even the mention of SOA to anyone in upper management could be a career decision—and not a good one.

What a shame.

SOA has a lot to offer when it is understood and its principles applied consistently and incrementally. In fact, it forms the foundation for cloud computing, which is rapidly growing in popularity (see the Forrester research article at ZDNet: http://www.zdnet.com/blog/btl/cloud-computing-market-241-billion-in-2020/47702).

As an SOA consultant and trainer for many years, I have seen many organizations struggle with SOA adoption and, in this article, I wish to share with readers what I believe to be the most important factors that impact whether you will succeed or fail at SOA. Let me begin by dispelling a few entrenched myths around SOA.

- SOA is not and never will be a magic bullet. It will NOT solve all your corporate IT problems in one fell swoop.
- SOA is not easy. Adopting SOA is a long-term commitment to better IT practices and system design and is not a quick fix.
- SOA is not a product. Vendors do not sell SOA; they sell products that support SOA development and speed up its adoption.
- SOA is not about web services. It is primarily about architecture and design. Web services are a practical result of this and an enabling technology, nothing more.

- SOA is not "an IT thing." It is NOT about technology, and even though the standards it promotes do simplify integration, that is not its main goal.
- SOA helps to address issues of years of IT system neglect or the accumulation of decades of technical debt. It can provide a new path and approach that can dramatically reduce IT system degradation in the future and salvage existing legacy systems to the maximum possible extent. However, this will not happen overnight.
- SOA requires a commitment from the whole organization. It can NOT be driven solely by the IT department. SOA should involve the business from the very beginning, because it is not purely an IT concern.

I do not want to discuss here the many reasons why organizations attempt SOA; this will be a topic for future articles. I will assume that the decision has been made for one or more good reasons (say, for example, reusability, flexibility, and/or agility) and that the organization is embarking on SOA either holistically throughout the organization or within some pocket of it (typically the IT department drives initial adoption but not always). So, what happens next?

Buy-In from Senior Management

A major goal of SOA is the evolution of the business and its processes, with potential widespread impact across the organization, so it is vital that members of the senior management team actively sponsor and promote SOA within the organization.

By the way, this is a show-stopper. Let me put this very simply: if you do not have support from upper management for your SOA efforts, and they do not share your long-term vision, you will not succeed, so don't even attempt it. Nothing more needs to be said.

Education First

On one occasion at a Fortune 500 company, while training their IT personnel in SOA, I asked the attendees whether they thought they were doing SOA in their organization. One of the team leads answered immediately, "Of course we are! We have at least 30 web services already in production, and we have been developing them over many years." I told him that my question was not whether they used web services but whether they were working within a SOA. Clearly, he was not able to understand the distinction. That's why education is near the top of my list.

In every case where I have stepped into an organization that claims to be "doing SOA," I have discovered a large degree of ignorance and many misconceptions around what SOA is all about, why it is needed, and how to go about doing it. It makes sense that the first step in adopting SOA is to educate the organization. Take note that I include a full vertical slice in the scope of this effort—from testers, developers, and programmers in IT, to business analysts and sales reps, all the way up to project managers and the executive team, including upper management and C-level executives. The educational campaign I am proposing involves representatives from both the technical and the business side of the organization. This is critical. Without a shared language and vision arising from a shared understanding of SOA, communication between technical and non-technical people will not be effective, and a dysfunctional relationship will result.

Of course, the type of education will need to be customized to each role; for example, developers might be interested in web service design and deployment, whereas a C-level executive wants to know how a service can impact the bottom line or open new markets. Formal SOA education can effectively bridge the business/IT gap and is the first step towards creating a productive environment where SOA can thrive.

Standards Next

A large part of what SOA is all about involves a strong commitment to the use of enterprise-wide standards. I use the term "standard" in its broadest sense—not only technical standards like SOAP, XML, or WS-BPEL, but also process and enterprise standards such as TOGAF, ITIL, or *PMBOK® Guide*. It is important to establish and agree upon the standards that everyone must adhere to within the organization when contributing to an SOA. These are not in any way set in stone; of course, as your organization matures, so possibly will your need to expand your use of standards to address more advanced deployments. However, establishing standards in the early stages of an SOA implementation ensures that a solid foundation is created for future activities.

In many cases, it will be necessary to formally train personnel to be able to understand and work with standards. For example, developers may need to ramp up on XML web services, REST, or BPEL. Architects may need to brush up on their UML and, additionally, learn about service-oriented analysis and design techniques. Project managers may need to learn to adapt their approach to budgeting and work management for the delivery of services and not just system components. And so on. Every role is in some way impacted by SOA, and some education, training, or mentoring will be needed to ensure that everybody knows what they need to do within this new paradigm of SOA, which standards they must adhere to, and how to do so.

Take It Incrementally

Start small. Do NOT attempt to introduce SOA globally through a big-bang effort. The amount of fear, uncertainty, and doubt (FUD) that will be generated among your employees will surely be severe enough to paralyze all forward motion, and your SOA journey will come to a screeching halt.

Do not delude yourself; without buy-in from your team, you will not succeed with SOA. Everyone needs to feel comfortable with the principles and policies around SOA, and this can only happen gradually as initial suspicion and doubt give way to trust and acceptance.

This can only happen if SOA is shown to be effective and useful, and the best way I have seen to demonstrate this is to start with a pilot project that showcases what SOA has to offer. Perhaps it is integration through standardization, or maybe the maintenance benefits of loose-coupling; whatever the focus, it should be something that makes the rest of the organization stand up and take notice.

The most effective pilot project will solve an existing issue within the organization, preferably something the business has been struggling with for some time. There is no better way to sell SOA than to prove it out in practice, tackling a real-world issue that executives can readily understand and then see for themselves how SOA can be brought to bear to resolve it.

Make It Iterative

Extend the success of your pilot SOA project by attempting a slightly bigger or more complex one. Evolve your approach using the lessons learned from the pilot project; keep what worked and change what did not work. SOA adoption is an organic, evolutionary process, and it is unique to every organization, since every organization has a unique combination of culture, business, and technology. When you have successfully applied SOA principles to this project, then look for a slightly bigger and more complex business area within the organization that could benefit from SOA. In this way, you build confidence in the approach while honing both your understanding and expertise in SOA.

It's All About Design

Many organizations believe that the path to full SOA adoption begins with the development of web services, typically as a way to layer a standards-based communication framework over an existing legacy application programming interface (API). They are not completely mistaken. This is known as the "bottom-up" approach and can be successful if it is handled correctly. More times than not, this approach is mismanaged with predictable results such as:

- Poorly designed and technology-dependent interfaces
- Services that are two "close to the machine" and fine-grained, exposing deep technical details that create a tightly coupled system
- Competing/duplicate services (two or more services that achieve the same business goal)
- Rogue services (those that were put into production without a mandate or by bypassing internal policies and procedures)

Generating a web service using modern integrated development environments (IDEs) is an almost trivial exercise. Take an existing class definition in the language of your choice, push a few buttons, and voila! You have yourself a Web Services Description Language (WSDL) and all the underlying code you need to be able to deploy a web service. So, is that all there is to it? "I have a web service so my SOA is complete?" Not by a long shot. There are potentially many issues with this generated web service, for example:

- Do the operations make sense from a business perspective?
- What data model is being used for the messages?
- How are the messages structured?
- Do the messages make use of a standardized message data model?
- Is this data model related to the business domain model, and does it make sense within the context of the business of the organization?
- Are there any technology dependencies in the operations?
- Does this create vendor lock-in, and is doing so desirable in this case? In all cases?
- How do we know that there aren't other services out there that do the same thing? And what happens if this service fails?
- Who should I contact to rectify the issue?
- How do I make changes to the service without disrupting current users of it?

The list goes on. Not only are there potential design issues to grapple with since no thought was put into design and machines make terrible architects, but there are even more questions around how we manage the service lifecycle and quality of service (QoS) issues. Is the service secure? Will data I pass to it be safe from interception? Am I forced to use a particular vendor's products or technology? And so forth. These issues are of a different type: they are related to service governance, discussed next.

Governance Is the Key

Governance is policy. Management is governance in action; in other words, it's the enforcement of policy and procedures. Governance gives us control over the many artifacts that are generated when adopting an SOA, and this control is paramount. Without it we have chaos. Sometimes this chaos takes time to emerge, and organizations have the illusion of control, at least for a while.

Typically, the first time governance is even discussed is around controlling the service lifecycle. Usually by the time senior management is aware of the issue, it is well advanced and more difficult to correct. The typical scenario is one where the organization's IT group has decided to adopt SOA without telling the rest of the organization. Developers begin to crank out web services in large quantities, mimicking existing APIs from their current systems. Little thought is given to the nature of the interface operations or the data model they use. Conflicting data abstractions and overlapping functionality exist between services, but no one notices yet. Soon, these web services find their way into production, and users begin to rely on them, but no process has been put in place to handle user complaints or issues. The developer of the service is certainly not interested in listening to complaints or critique from customers, whether internal or external. In fact, in the meantime, he has released version 2.0 of the service with a completely revamped set of operations. Once deployed, current users realize that they are no longer able to use the old version (1.0), and they have no idea how to use version 2, whose API was not documented anywhere. Customer frustration increases.

Around this time, noise begins to trickle up to senior management, and they start to ask questions. By this time, there are several dozen in use, all poorly documented, with no governance policies, no description of use, no KPIs or other metrics defined, and no idea how many users each can support. The services simply have no visibility or control. The end result is obvious: there is NO EASY WAY TO KNOW how to fix anything that is going wrong. Even if we knew, it would be difficult to make any changes without impacting other services or the service consumers. In fact, we have created the same tangled web of tightly coupled software that we were trying to get rid of in the first place! (Ironic, really.)

This is the point when organizations realize they have a problem, and they have two possible paths ahead of them. 1) Blame SOA, pull the plug, and revert all web services to old APIs, or 2) throw them away and go back to the old ways. More often than not, the second option occurs, and many millions of dollars are wasted in the process.

The second option is to realize that governance is needed to control the service lifecycle (as a start) and then to begin to hammer out policies and procedures that must be followed in order to develop, publish, and consume services. This is where standards are often introduced and service registries created. Sometimes this even spurs the training that was mentioned previously, although training should have been done first thing. Sometimes, a disaster is needed to motivate management to do what needs to be done.

Retroactively introducing governance is more painful than doing so from the start, but it's better than not at all. Governance is a critical success factor. Without its three Ps (policy, process, and procedure) around EVERYTHING you do within an SOA—looking beyond web services to service orchestration, business process management and modeling, workflow and automation, etc.—SOA failure is probable.

Current policies need not be thrown out in favor of new SOA-based ones. Often, existing policies, procedures, and processes simply need to be tweaked to cater to this new service paradigm and its associated artifacts. It is not a reorganization. It is simply extending what the organization already does into the SOA way of thinking, and it is imminently doable.

Summary

SOA is all about architecture—after all, it's right there in the acronym—yet most organizations think it is about turning existing software components into web services. When you adopt SOA, remember that it is all about design and governing that design. It's about how you design your service interfaces, your services, your data model, and your business processes. It's about how you keep track of your services, how you control the design, definition, deployment, and distribution of your services and their artifacts, how you define a service contract and service level agreement for your service consumers, how to secure your services, and how to react when things go wrong with them.

If you stay true to the principles of service orientation and ensure that everyone in your organization understands them, then you have a great chance of success with SOA. And, when you succeed with SOA, your organization will be more adaptable and more agile, and the quality of your services will be much higher. Most importantly, all these benefits lead to more competitive efficiency and, in turn, to higher profits. In the end, that's what makes it all worthwhile.

Learn More

To learn more about how you can improve productivity, enhance efficiency, and sharpen your competitive edge, Global Knowledge suggests the following courses:

- Designing SOA Solutions Using IBM SOA Foundation (WS008G)
- Getting Started with SOA (WS007G)
- Developing Applications for IBM WebSphere Enterprise Service Bus V7.5 (WB753G)
- IBM WebSphere Service Registry and Repository V8.0 for Administrators (VS811G)

Visit **www.globalknowledge.com** or call **1-800-COURSES (1-800-268-7737)** to speak with a Global Knowledge training advisor.

About the Author

Dr. Rossi has been an IBM contractor since 2003, and as an IBM Certified Instructor for WebSphere, SOA, and JEE, he is able to provide expert advice on IBM technologies. Dr. Rossi can help your organization with guidance and strategic direction on the use of SOA and BPM methodologies, techniques, processes, standards, and best practices. As an active software developer working with these products, Dr. Rossi has up-to-date, practical experience that he can share with clients in order to help them avoid common pitfalls and increase their productivity.