Implementing a Successful Service-Oriented Architecture (SOA) Pilot Program

Actional's Best Practices for Selecting and Deploying Initial Projects, and Preparing to Migrate to an Enterprise SOA

Actional

Forward by MomentumSI

MOMENTUM Si

Executive Summary

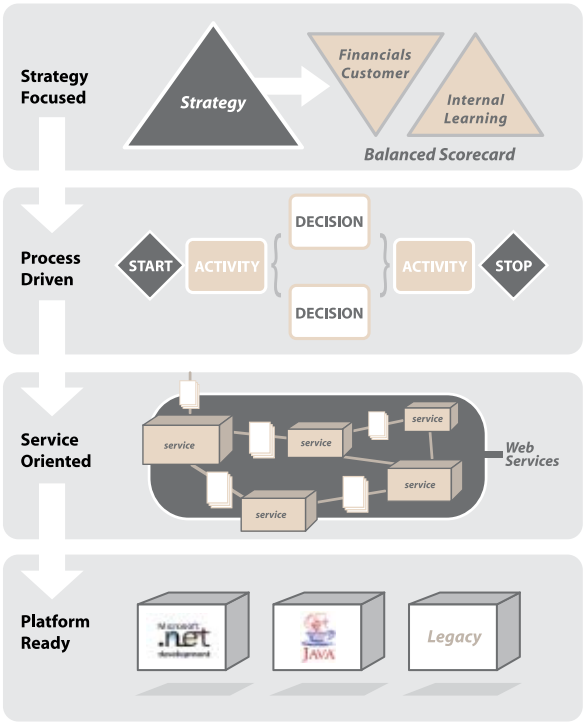
By consolidating and reusing common services, Service-Oriented Architectures (SOAs) offer many benefits to today's organizations. This white paper will guide those pursuing SOA towards proven best practices to increase the likelihood of garnering the benefits of an SOA, faster and easier. The paper will then present Actional's best practices for selecting and implementing a successful SOA pilot and then how to utilize the experience and lessons learned to migrate to an enterprise-class SOA.

Table of Contents

Forward

Day by day, company by company, IT organization by IT organization, today's enterprise is busy architecting for business solution agility and the alignment of key assets around the emerging Service Oriented Architecture umbrella. The ability to embrace SOA leads to the ability to rapidly capitalize on future IT investments and leverage existing technologies both inside and outside your organization. Many organizations will struggle as they seek to identify, implement and build on their first SOA forays into the new environment. This is unfortunate, as a number of best practices and lessons learned can be apply from the past and the growing SOA experience from technology providers and practitioners. As you work to extend IT capabilities utilizing SOA, a key for many of our enterprise customers is a pilot program which enables the greater understanding and methodological deployment of Web Services and service oriented architectures around a structured initiative.

The Business Driven Architecture



Success is predicated on the fundamentals and insight collected within the following pages. Implementing a Successful Service-Oriented Architecture (SOA) Pilot Program clearly identifies the axioms and best practices that underlie achieving SOA objectives, regardless of size or scope. The Pilot Program offers an expandable approach to SOA, and is a strong foundation to the larger blueprint of a Business Driven Architecture. You will find within the context of the following pages a concise understanding of the SOA reality and its application. The white paper provides a detailed understanding of the elements that comprise SOA, as well as techniques and practices for creating organization wide software integration solutions using SOA concepts.

Each organization will engage SOA differently and extract the greatest benefits based on their unique needs. Information and tools are the enablers that allow you to determine the best path.

The SOA corridor for success and failure will be paved by the development of an overarching strategy and its deployment across the corporate domains of people, process, technology and organizational requirements. The following information should prove highly useful as you navigate towards our SOA goals and objectives. I invite you to read and apply the well developed roadmap contained in this white paper.

Alex Rosen
Principal, SOA Practice
MomentumSI

Introduction

IT agility is one of the main concerns for today's complex enterprises. Even the most well planned IT infrastructures face significant challenges introduced by mergers, acquisitions, and rapid growth of the enterprise. To increase efficiency, companies must phase out functional silos that incorporate overlapping and duplicate processes and add complexity to their IT infrastructure.

Service-oriented architecture (SOA) provides the framework to re-architect IT infrastructure, eliminate redundancy and accelerate project delivery via consolidation and reuse of services (often referred to as Web services). SOAs can adapt quickly to changing business needs, deliver new IT projects faster at lower costs, and reduce ongoing IT administrative and infrastructure expenditures.

In a SOA, applications are not built as standalone monolithic silos. Instead, business-relevant services (e.g. customer, ordering, inventory, etc.) are either built new or layered on top of existing applications. Then, project specific application logic is built on top of the services in a separate architectural tier. This approach is what leads to the benefits of SOA:

- By building services, they can be leveraged across multiple projects – reducing redundancy and avoiding the need to rebuild functionality in new projects. This helps deliver projects faster and lowers ongoing costs.
- By decoupling project specific application logic from the services, it enables flexibility as this logic can be easily changed, updated, or even replaced to address new business needs – combining the underlying services in new ways without having to change or rewrite them. This leads to business agility as well as lower cost over time.

The benefits are compelling but SOA also changes the dynamics of IT by introducing interdependencies across projects, applications, and even IT teams. Historically, allowing project teams to be autonomous drove accountability to the project teams and managed risk by limiting the external interactions. In contrast, the benefits of SOA in many ways are derived from external interactions (using services built by other project teams). These two worlds must be brought together without falling into the traps that exist at either extreme: A brittle, unreliable, and difficult to maintain spaghetti of application interconnections which can occur when project teams are left to build and use services without the right controls in place; or, at the other extreme, a central organization whose job is to connect applications together which becomes a bottleneck for project teams, reduces accountability and increases the risks for project success.

Because of the changes required and unique pitfalls of SOA, before moving to this form of architecture, careful pre-planning is essential. Your move to SOA is migratory, leveraging existing applications that deliver value while building new applications and services. This generates a complex heterogeneous environment that avoids the "rip and replace" approach and focuses on building new services only when a new opportunity arises or existing solutions do not deliver the necessary results.

Critical Pre-SOA initiative questions include:

1. How do we phase in and successfully manage the move to SOA within the current IT environment?
2. How do we leverage the existing IT infrastructure and investments in the SOA?
3. What procedures need to be established or modified?
4. Where will policy and security reside?
5. How do we explain the benefits of SOA and provide financial justification to the business stakeholders?
6. How does the move from silos to SOA impact our ability to detect and resolve problems?
7. Who owns shared applications or services and how do we allocate costs and budgets for these services?
8. What do we need to do to ensure our SOA scales from pilot to production?

With thorough up-front planning, a transition to SOA can be achieved efficiently and provide tangible benefits to the enterprise. The following sections will offer guidance in answering the questions listed above.

What Are the Overall Goals of the SOA Initiative?

It is important that you clearly understand and can articulate your organization's unique goals that will drive the move to an enterprise-wide SOA. Every enterprise SOA will be driven by unique business and technical goals – as well as by short-term and long-term goals. Understanding these up front will help you choose the right projects and help ensure the success of early implementations.

Critical questions to answer include:

1. What are your primary reasons for using SOA? (To reduce costs? Achieve better flexibility? Enable faster delivery? Improve customer satisfaction?)
2. What is driving your near-term use of SOA? (Connecting your core applications? Integrating with your partners? Providing a single view for customers and/or users? Getting real time business metric? Regulatory Compliance?)
3. What is the long-term potential for SOA in your organization? (Faster product introductions? Flexible outsourcing? Business process flexibility? Stricter governance? Other?)

SOA goals may include:

- Operational Efficiencies
  –Reducing infrastructure and management costs
  –Improving end-to-end process visibility, security, and/or control
  –Streamlining root cause analysis and automating triage

- Business Efficiencies
  –Accelerating time-to-market of critical business applications and processes
  –Enabling easier compliance with policies (security, business, regulatory) from all functional areas
  –Maintaining business continuity across the enterprise through real-time response to security, performance or availability breaches
  –Seeing business metrics in real time
- Development Efficiencies
  – Increasing service reuse and reducing development costs
  – Reducing time to deployment of new services
- Integration Efficiencies
  –Reducing integration cost and complexity through the use of industry standards
  –Seamlessly interoperating with, and leveraging the capabilities of, key third-party infrastructure solutions such as identity management systems, directory services, enterprise management systems

By documenting precise goals for an enterprise SOA, teams involved in the initiative can more effectively choose the right projects, manage the projects (e.g., avoid "scope creep"), make sure the overall results are not derailed and manage the expectations of surrounding stakeholders.

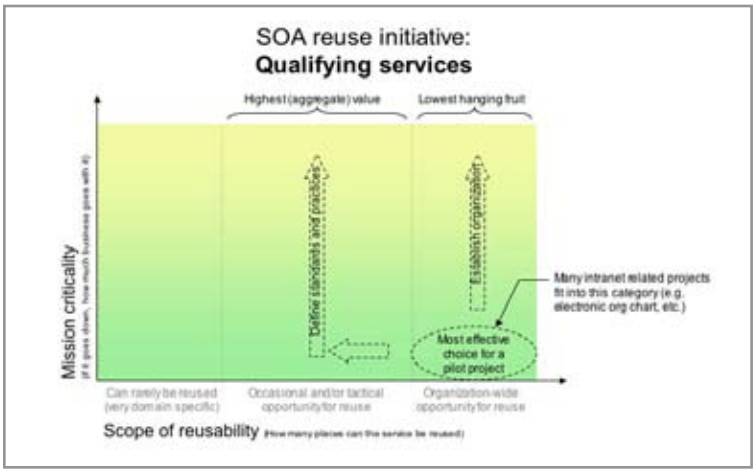
Starting with a Pilot

Rome wasn't built in a day, and neither is SOA. It is best built incrementally, project by project. By starting with an explicit SOA pilot project your organization can learn from its successes (and failures) to increase the success of their overall SOA initiative. The next section of this paper will discuss the issues you need to examine when choosing and launching a successful SOA pilot project.

Best Practices for a Successful SOA Pilot

Through many successful enterprise-class customer engagements across multiple vertical markets, Actional has compiled a set of best practices to help enterprises select and implement effective SOA pilots.

Key steps to a successful SOA pilot include:



Step 1: Identify the Goals an Initial SOA Pilot

The pilot will provide valuable insight into SOA infrastructure that will be extremely useful as you expand to building an enterprise-wide SOA. This is a great time to experiment and learn from trial and error. Goals for this initial project should be clearly articulated, and may include:

- Developing a service that can be reused by multiple departments
- Consolidating duplicate applications into one server

- Providing a showcase of successful SOA implementation to clearly illustrate the benefits of reuse and consolidation
- Learning valuable lessons that can be leveraged in the larger SOA initiative
- Understanding the tasks involved in getting services into production as well as the day-to-day tasks required to manage SOA once it is production

Step 2: Create a Cross-Functional SOA Team

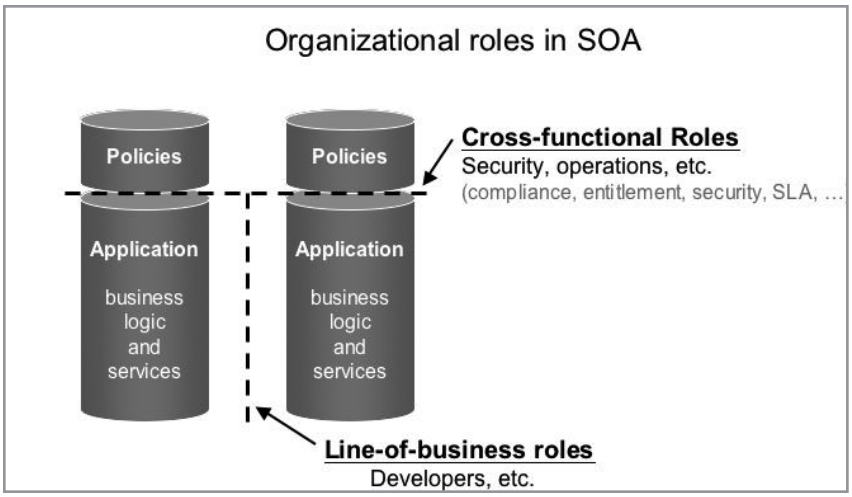
Successful SOA pilots involve the cooperation of cross-functional departments including line of business, development, operations, security and more. While these stakeholders may not be directly involved with a typical pilot on a daily basis, it is critical that they experience both the pain points and benefits of SOA through team participation. Regular meetings and communication will expose any concerns or territorial aspects that could inhibit SOA production.

Choose the team members wisely and consider their ability to participate outside of their daily responsibilities. Outline time commitments for both meeting participation and communication review and ask each team member to agree to the commitment up front. Additionally, create and share a communication strategy that includes a reasonable number of updates sent to each team member. Then it is important to stick to that schedule as it lends credibility to the pilot and elicits the best feedback and participation.

Moving towards an SOA will most certainly dictate a change in the way applications are developed and deployed. Most believe the team and organizational change is the toughest challenge in migrating to SOA as many see this change as the destruction of the silo-based organizations that exist today. Therefore, the ability to pick the right cross-functional SOA team may be the most critical aspect of the pilot.

Step 3: Determine the Appropriate Pilot

In order to accurately and successfully exhibit the benefits of SOA to those who may be skeptical, you must first choose the appropriate pilot. It must clearly demonstrate the promise of SOA without causing material harm to any aspects of the business – that is, a service with the best risk/reward ratio. Early pilot success lends credibility and therefore leads to production SOA.



Consider the following questions when determining an appropriate pilot:

1. Create a new service or leverage an existing one?
   New services can be easier to create initially but wrapping a Web service around an existing legacy system may provide measurable benefits in less time

2. Select a high or low visibility pilot?
   - Low visibility pilots:
     – Are less critical if problems arise during implementation
     – Allows triage without constant scrutiny
   - High visibility pilots:
     – Higher visibility pilots come with multiple opinions and the associated political complications
     – Benefits are delivered earlier and to more departments
     – Determine stakeholder requirements and whether or not a visible success is necessary

3. Choose a revenue-based or back-end application?
   The most common pilots are often related to user-facing systems such as portals and Web sites. For example, a portal that provides a single view for customers or employees based on data gathered from one or more different systems. These are often good choices because the end results are tangible and can provide a hands-on experience with the benefits of SOA. In comparison, back-end integration, such as synchronizing data between systems, can be valuable but it is difficult to clearly demonstrate the benefits.

4. Choose a customer-facing or internal application?
   There are many pros and cons with choosing either a customer-facing or internal-only application. Internal services, such as one in Human Resources used to enable employees to update their personal information via a corporate intranet portal, can protect customers from potential issues but feedback may not be relevant to wider scale use of SOA. In comparison, a financial institution could use a currency exchange rate service for a limited volume but highly visible pilot.

5. Choose a transaction-oriented or query-oriented service?
   Query-oriented services are used primarily for making existing data available for other uses such as retrieving lab results in a healthcare environment. In contrast, a transaction-oriented service is one that creates new information records, or triggers a new business process such as the DMV registering a new automobile.

   Transaction-oriented services are inherently more risky as problems can lead to data loss. As a result, many organizations start with query-oriented services to unlock existing data assets. Later, they extend these with transaction-oriented operations, as many services require both. For example, in telecommunications, a self-service Web site should be able to provide both self-service provisioning (transaction-oriented operations) and information on service requests, billing history, and more (query-oriented operations).

Step 4: Quantify Pilot Results

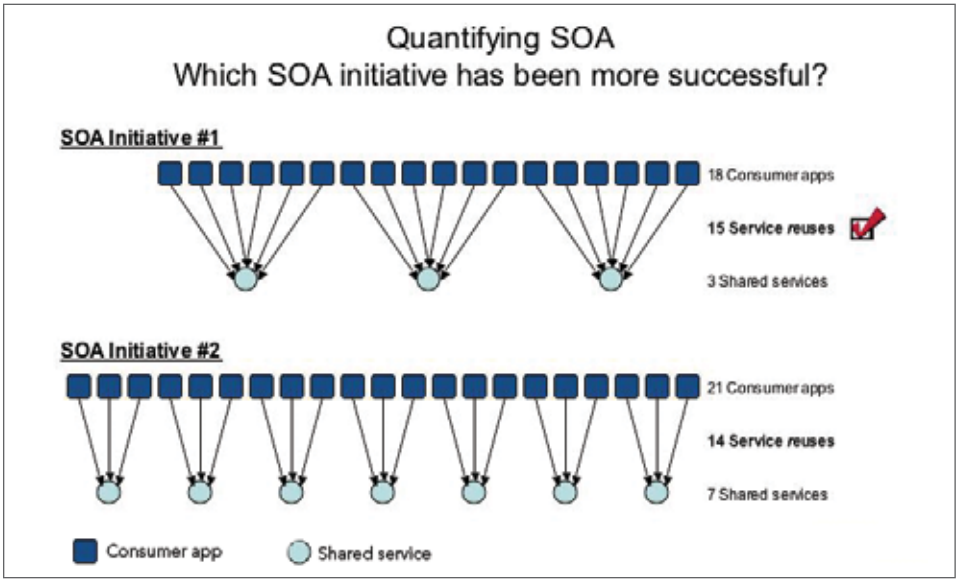
The most important deliverable for a SOA pilot is quantified results. Senior management often requires ROI calculations as well as tangible proof of your pilots' success. Create a method for ongoing data capture, particularly if a pilot is conducted over a large period of time, so the data is both accurate and readily available at the completion of the project. Gathering and compiling these figures is the key to budget justification for the next phase of any SOA initiative.

SOA can be measured by several factors:

Reuse of Shared Services

The number of instances a shared service is reused is an effective SOA measurement for ROI. Each reuse results in cost avoidance or reduction of building, maintaining, and operating a single-purpose service. In order to properly calculate ROI of shared services, some additional base metrics are needed:

- Costs to build/operate/maintain a shared service. This is the cost of having a shared service.
- Cost to build/operate/maintain a single-purpose service. This should be similar to that of a shared service if your SOA infrastructure is effective.
- Cost to use an existing service built by someone else. This is the cost incurred by reusing a service (the alternative would have been to build a single-purpose custom service). Controlling this metric is critical to SOA success. The reuse of a service is effectively an integration – and so your SOA needs to be structured to control the costs of integration.



- Shared Service Consumers
  The number of shared service consumers (relative to total consumers) measures the breadth of SOA adoption. That is, it measures how well the "cultural shift" of SOA has permeated the organization. This is not directly correlated with SOA business benefit, but is an important metric none the less.
- Web Services Adoption
  The number of web services created is not actually a measurement of SOA. Instead, it primarily reflects the breadth of adoption of the underlying technology. In many cases, this metric can actually be used as a negative indicator of SOA failure. That is, if a large number of services exist, but few are reused, this may be an indication that your SOA initiative needs some revision.
- Business responsiveness
  Compare the time it takes to change or add a feature to a non SOA based applications with a similar features change to a service.

In addition to the above methods of measurement, every project will have its own unique business justifications and associated measurements. For example, exposing customer information via a service in order to create a self-service customer portal may be used to significantly reduce call center operations costs. These benefits will, of course, differ for each project. Even if you define a formal SOA pilot project, don't do this for the sake of moving to SOA – the project should provide value to the business.

Anatomy of a Successful Pilot Program

In summary, the keys to a successful SOA pilot include:

- Analyze pilot requirements and scope them for early success
- Create a cross-functional team
- Choose the right pilot
- Set realistic goals
- Understand deployment considerations
- Carefully quantify the pilot from start to finish

One important pitfall to avoid in your SOA pilot implementation is limiting it to a test lab. Labs will provide your pilot implementation team with a safe place to learn the unique issues and benefits associated with SOA in each enterprise. However, SOA has significant impact on every stage and transition in an application's lifecycle (development, testing, staging, production, and especially subsequent versioning). It's important to gain real-world experience with the issues at these different stages of the lifecycle. These are among the most common causes of perceived failure of an SOA initiative. Unfortunately, this experience cannot be gained in a test lab.

Migrating to an Enterprise SOA

After bringing a pilot successfully into production, it is time to examine the issues involved in migrating the enterprise to an overall SOA framework.

Your initial pilot revealed a lot of useful information that can be leveraged throughout the migration to an enterprise SOA. Carefully analyze what went right with the pilot program, and more importantly, examine what went wrong and how the issues were overcome. Keep the information available, share it with your team and refer to it frequently.

It is also important to investigate any situations or issues that weren't part of your initial SOA project, including the use of more complex security requirements, meeting regulatory requirements, etc. to understand where your remaining gaps in experience exist.

Beyond your pilot projects, there are five remaining keys to a successful SOA initiative:

1. Defining Common Standards

On your first SOA project, your choice of infrastructure and technologies may not have been a critical success factor. However, as your initiative moves from project level to enterprise level, sharing common standards across projects becomes a critical success factor – if your services don't "speak the same language", then reusing services will be difficult and costly. This can lead to failure of your overall SOA initiative.

The first level of defining standards is deciding on the application interoperability standards – how will services talk together. This list typically includes XML, SOAP, HTTP, and UDDI. Beyond these base standards, you need to consider your security strategy. Note that, as much as possible, you should be choosing standards not products. Over time you will have many different applications and services talking together. Some will be built in-house on your current application platforms, some will be built in-house on future application platforms (you may, for example, choose to switch your application server vendor at some point in the future), and still others will be part of commercial, off-the-shelf applications. In many cases, the consumer application will not be built on the same technology platform as the service provider application. You need to make sure that your choices for service communication don't constrain your ability to add new applications to your SOA "network". As such, choosing open standards that are supported by a broad range of vendors is critical to the long term success of your SOA.

Note that there are two dramatically different types of open standards: API standards (such as JMS, JDBC, and ODBC) and interoperability standards (such as TCP/IP, HTTP, and XML). While API standards are important, they still have several key constraints: they are platform specific (for example, JMS only applies to Java not .NET) and they are vendor specific (an application that supports MQ Series cannot talk directly to an application that supports TIBCO even though both applications may be written using JMS). You would need yet another layer of infrastructure (e.g. messaging adapters) to try to get them to talk together. In contrast, with interoperability standards, the consumer and service provider applications, even when written on dramatically different platforms, can talk to one another provided both support the same interoperability standards. Be careful to choose interoperability standards that have the widest adoption to avoid constraining your choice of platforms.

Once you've chosen the common application interoperability standards for your organization, the next set of standards you need to consider involve security and management. A unified security architecture is critical as you will have many cross-application interactions and if each application has a different security model, it can lead to security holes and increased costs for managing and maintaining your SOA environment. Similarly, a unified management strategy will enable you to get a big picture view of your SOA across projects – a view from the perspective of the business processes it supports, not the silos that make it up. This will allow you to more effectively control the SOA, ensuring reliability and robustness. Remember, you can no longer rely on the security and management built into any one platform. For the same reason you need common standards for interoperability, SOAs are heterogeneous by definition but management and security must span the SOA.

2. Governing Project Teams

The team that built the pilot project was hand picked and heavily involved in understanding the requirements of the SOA initiative. After a successful pilot, you then need to roll-out the SOA initiative more broadly to a larger audience of developers and project teams. There is a lot of new information for these teams to learn, and many pitfalls of which they may not be aware. For example, services built by these teams may use a non-standard approach to security and they may not be WS-I profile compliant (the WS-I defines "profiles" which clarify standards ambiguity to ensure interoperability across platforms), or they may be technical as opposed to business-centric. These types of issues can become barriers to the overall SOA initiative as they make it more difficult to leverage the existing services.

There are a number of approaches to addressing this:

- **Training.** First and foremost, training teams on using the new tools and technologies is essential. Training on best practices and pitfalls is also essential. For example, training developers that relying on the use of time zones in "date" data types can impact interoperability.
- **Infrastructure.** Your choices of infrastructure software should be able to offload the development teams from having to make decisions that impact interoperability. For example, the infrastructure used for projects may offload the developers from needing to code security, auditing, management, etc. into their application. Not only does this make it easier and faster for them to roll out services, but also enforces corporate standards automatically.
- **Tools.** Tools that developers can use while building their services can aid them in performing automated checks for issues. For example, there are tools available that can help developers confirm that their service definitions (WSDLs) are WS-I compliant as they build them.
- **Checkpoints.** As a final stage, your process should define checkpoints where you can validate that services are appropriate. These checkpoints should be placed at a point in the process where they cannot be easily bypassed by someone in a hurry. For example, the process of publishing a service in a registry may be controlled by requiring approval – this becomes a natural checkpoint. At these checkpoints you can perform automated assessment for common issues (such as services not being WS-I compliant or not following best practices about schema design) as well as manual reviews for checks that cannot be effectively automated (e.g. use of corporate approved schemas where appropriate, or service granularity). Of course, catching an issue at this stage is costly – it may require that the service be significantly rewritten or redesigned, introducing cost and delays to project delivery. As such, this should only be relied on as a last resort – your processes should be designed to address as many issues as possible up front (using training, infrastructure, and tools) to avoid wasted effort.

Many of the points raised about checkpoints and infrastructure have given rise to the popular subject of governance. Today a lot of the focus on governance is around how to architect, design, develop and deploy services. But this is only half of the governance issue. The other half occurs in the runtime environment and ensures that services are actually operating and driving the business as intended. These two parts of governance need to be addressed together understanding which parts of governance live in pre production, those that exist in runtime and those that span both.

3. Separating Responsibilities

The use of infrastructure that offloads project teams from having to build-in functionality for security, auditing, management, etc. into their projects provides a natural ability to separate the roles and responsibilities in your SOA. You can have cross-functional teams responsible for security, compliance, or operations. However, while taking these responsibilities out of the hands of project teams should shorten project delivery and produce a better overall resulting project, it can also introduce new barriers. The key challenge occurs when these cross-functional roles require domain-specific knowledge to complete their tasks. For example:

- A security team member might not know the appropriate authorization rules for order creation versus order cancellation.
- A compliance team member may be tasked with enforcing privacy regulations, but not know what schema elements of an XML document require such privacy (for example, an XML element named "socno" might be a social security number, which would require encryption in certain regulatory domains, or it could just as easily represent something innocuous like socket number).
- A deployment team member may know that rerouting across data centers should be used to assure service levels for important customers but that certain operations cannot be rerouted to avoid data replication and consistency problems.
- An operations team member might be tasked with monitoring SLAs, differentiated by customer class, but may not know how to determine in what customer class the requester falls.

One approach to addressing these challenges is to require that these cross-functional teams "loan" members to project teams in order to match appropriate skills to each project. This, however, often creates bottlenecks for project delivery – the project team needs to "wait in line" to get a person allocated to their project, and they team becomes responsible for providing the domain-specific education to the cross-functional team member – increasing the risks for project delivery. An alternate approach is to clearly define key domain-specific concepts (e.g. "customer", "order", "personal identity") across the organization. This then allows:

- The cross-functional teams to focus on defining their rules abstractly, independent of any individual service, using the concepts (e.g. "all personal identities must be encrypted").
- The project teams, separately, map their unique data to the concepts (e.g. "This services' XML element <socno> is a personal identity" or "this operation can be rerouted").

By providing this separation, cross-functional teams can define their policies once, yet have them apply globally to all services. Project teams can then be freed from bottlenecks that require cross-functional team members to re-implement the policies for them. Your choice of SOA infrastructure can dramatically alter your ability to effectively separate these roles. Be cautious to determine what types of domain-specific information are critical in your environment, and evaluate how well the different options enable a clear separation of roles and responsibilities.

4. Leveraging Existing Applications

It's highly likely that the bulk of existing applications do not support XML and web services – whether these applications were written in-house or packaged. While you may choose to decommission some of these applications, many others will remain in service. As such, it is necessary to formulate a strategy to leverage and service-enable your existing application assets.

There are a number of different approaches to this. One approach is to hand code web service wrappers around the existing application when needed. The advantage to this approach is that you can use this layer to re-factor the service to be more business focused. The disadvantage is that it is can be time consuming and difficult to make a scalable, reliable, and robust hand-coded service wrapper.

The alternative is to use packaged integration products to service-enable the existing application. This may be an EAI or ESB solution or an adapter which directly converts the existing application into Web services in a single step. The advantage of these solutions is they can produce a more robust result. The disadvantage is that they don't make it easy to re-factor the service – the existing application appears to be web service enabled, but the interfaces are not necessarily the right level of business granularity. For example, the web service might have separate operations for creating "blank" orders and then populating the order with line items – in contrast a well designed ordering service would allow you to create a complete order with a single operation. This is a prime example of the difference between a technical service (designed the way it is for purely technological reasons) and a business service (designed to map to logical operations that the business performs).

The best approach is often to combine the techniques: use an adapter, EAI, or ESB to directly service-enable the existing application as technical services then write a business service layer around the technical services using your choice of platforms.

5. Ensuring Executive Visibility

In a SOA, no individual project addresses a complete business need – by leveraging existing services to support a new project, all of these services are essential to the business function. As such, it is important to provide executives with a view of the overall environment organized by business function, rather than by service or project. In the past, with each IT silo serving a unique business function, dashboards and other decision support tools could be tied directly to an individual silo – in fact, they could be implemented by the same project team that built the application. In contrast, business-relevant dashboards and decision support tools in SOA need to gather and combine information from across the interconnected services that now make up one business function. This necessitates a different approach – typically involving teams and tools organized around the business functions, rather than tied to any one specific silo'd service or project.

Leveraging Existing Experience

It is always beneficial to leverage the expertise of those who have completed numerous successful SOA implementations. The tips and tricks learned from actual SOA implementation experience are extremely valuable. This is even more important for pilot projects as the success (or failure) of a pilot project often determines whether the initiative will be allowed to continue.

Because SOA is a relatively new concept, you may have to dive into analyst research or reach out to peers and vendors in the market to find appropriate material. If budget permits, engage with an experienced professional services organization for guidance with technology that will assist in making your SOA initiative successful.

Summary

Service-Oriented Architectures can provide tremendous benefits to today's organizations. SOA services can enable your organization to adapt more quickly to changing business needs, deliver IT projects at lower expense, and reduce the ongoing costs of your IT infrastructure.

Through years of experience and customer engagements, Actional has the expertise to guide organizations through pilots and into SOA production with industry proven technology and services to develop, deploy, measure and manage successful SOA projects.

800 W. El Camino Real, Suite 120
Mountain View, CA 94040
Phone: 650-210-0700
Email: info@actional.com
Web: www.actional.com