

The SOA Magazine

Feature Article



Ten Strategies for Overcoming the Technological Impact of SOA Governance

by Robert Schneider

Published: November 17, 2008 (SOA Magazine Issue XXIII: October-November 2008)

[Download this article as a PDF document.](#)

Abstract: As many enterprises currently carrying service-oriented initiatives have learned the hard way, SOA governance isn't a "nice-to-have" option. Instead, it's vital to the long-term health and effectiveness of the undertaking. What makes governance particularly challenging is the sheer number of interacting software assets found in a typical service-oriented architecture. Forward-thinking organizations have turned to SOA governance automation to help manage this daunting task. This article offers a series of simple tips and tactics to bolster the likelihood of being successful when selecting and implementing SOA governance technology. It also points out the interplay between governance solutions and the major phases of a service-oriented initiative.

Introduction

For many organizations, embarking upon an SOA initiative involves a significant investment in staffing, training, hardware, and software. Unfortunately, far too many of these enterprises overlook the importance of protecting their investment by deploying a well-planned, robust SOA governance infrastructure.

In this article, I point out a few best practices that you can employ to help bolster the chances of being successful when bringing SOA governance technology into your organization. I begin by attempting to explain the paradox of how so many organizations are managing to implement their SOA without investing in governance technology, as well as the unpleasant surprises that await those types of entities. With that out of the way, I point out the relationship between the major phases of the service lifecycle and associated governance technology. The remainder of the article is dedicated to itemizing 10 tactics that should improve the likelihood of a happy SOA governance outcome.

SOA Governance Today

The majority of IT organizations are muddling through either without any formalized SOA governance technologies, or with a collection of home-grown solutions. In addition to the ever-popular word-of-mouth approach, examples of internally-developed governance applications include:

- Wikis
- Spreadsheets
- Emails

Given that governance is so important, how is it possible that many of these enterprises would describe these manual tools as meeting their needs? Reasons for this perceived satisfaction include:

- No overarching SOA vision
- A relatively small number of deployed Web services
- Small design teams

- Re-use and re-composition aren't a priority

As it turns out, this peace-of-mind is likely to evaporate upon the first serious governance-related crisis, which often leads to a panicked search for technology to halt the conflagration.

Governance and the Service Lifecycle

Before delving into the governance tips, it's worth noting that from a governance perspective, the service lifecycle can be broken down into three major phases:

1. Design-time
2. Testing and Quality Assurance
3. Run-time

Each phase introduces unique governance process and technology requirements. It's imperative that the selected governance solution be capable of addressing the needs of all service lifecycle phases.

During design-time, solid governance technology can help with:

- Meta data management
- Service discovery
- Service composition and modeling
- Disseminating organizational policies

During testing and quality assurance, governance can be of assistance in:

- Service unit validation & composition interaction
- Policy adoption
- Security compliance
- Service performance prediction

Finally, governance has a major role to play at run-time, including:

- Service level agreements
- Version control
- Error reporting and management
- Performance monitoring

Ten Governance Technology Tips

Strategy 1: *"Include governance technology as part of your overall SOA roadmap."*

Scope: All service lifecycle phases

Many organizations fall into the trap of deferring any governance arrangements until after services are being employed to run a significant part of the business. Waiting to make this decision often means that you'll need to incur additional effort, cost, and overhead once you've selected a governance infrastructure. Retrofitting always takes longer than expected, and siphons off valuable resources. These added burdens can jeopardize the entire SOA initiative. For all of these reasons, it's crucial to avoid the temptation to wait until you have "enough" services before thinking about governance. Since many SOA initiatives rely on the vision of an architectural committee, governance topics should definitely have a place on the agenda of these working groups.

Strategy 2: *"Make sure your governance platform is agnostic with regard to service development technologies."*

Scope: Design-time

When speaking to a collection of software developers drawn from different organizations, one sure way to trigger a donnybrook is to initiate a debate on the relative merits of .NET vs. Java. These religious wars can be amusing to observe, but they can also wreak havoc on an enterprise's governance initiative. Consequently, when choosing a governance platform, it's important that the selected technology work, at a minimum, with services developed in Java and .NET. If your governance platform only supports one style of development technologies, you'll end up living with multiple governance software installations.

The next big decision when selecting a governance platform is to decide between an open source solution vs. a proprietary product.

Open source benefits include:

- Reduced likelihood of vendor “lock-in”.
- Compliance with the “open source only” policy for infrastructure software regulation found in many enterprises.
- A reduced financial outlay, which makes it more likely that the organization will elect to implement this kind of governance software.

Proprietary solutions benefits include:

- Solid integration with design, development, and management tools.
- Simpler “one-stop shopping” that simplifies things, while yielding a better “out of the box” experience.
- Growing trend of software vendors delivering their solutions as open source.

Strategy 3: *"Make sure your governance platform is able to support a full range of service deployment technologies."*

Scope: Design-time

Although many people make the mistake of viewing Web services as the only way to implement a Service Oriented Architecture, there are other technologies that can get the job done just as well. These include Java objects, CORBA, and other service implementation approaches. With this reality in mind, it's vital that your governance platform be able to recognize and work with a broad range of services, regardless of the chosen implementation strategy. As a bonus, the governance solution should be as non-intrusive as possible, since there's a good chance that it won't be possible to alter the pre-existing services present in your organization.

If you fail to select a governance solution that meets these criteria, odds are that you'll either only govern a portion of your enterprise, or will need to deploy multiple governance platforms. Neither of these outcomes is desirable; partial governance is not much better than no governance at all. Many administrators and developers have wasted copious amounts of time going down the wrong paths when trying to overcome a governance-related problem when a comprehensive picture of all software resources would have pinpointed the issue.

Strategy 4: *"Recognize the importance of testing as part of your overall SOA governance responsibility."*

Scope: Testing & Quality Assurance

It's an unfortunate fact that quality assurance often takes a back seat to its more glamorous software development sibling. While this may be tolerable when building traditional siloed applications, it doesn't work in an organization that's undertaking a Service Oriented initiative. Given that services form the foundation for multiple applications, it's vital that testing receive the proper amount of time and attention. The overall scope of quality assurance expands as well: it's essential that your testing go beyond individual services to include complex compositions of multiple services. Composition testing often requires significant performance-driven regression testing. Since run-time composition usage can be difficult to decipher, it may be necessary to employ scoping or other monitoring technologies to determine true service interaction. Finally, since testing and governance are so closely related, it's wise to give serious consideration to integrating your chosen testing software into your overall governance environment.

Figure 1 shows an example of service testing software being utilized in a governance-related capacity. In this case, we're measuring service test coverage to ensure that our quality assurance efforts are as far-reaching as possible.

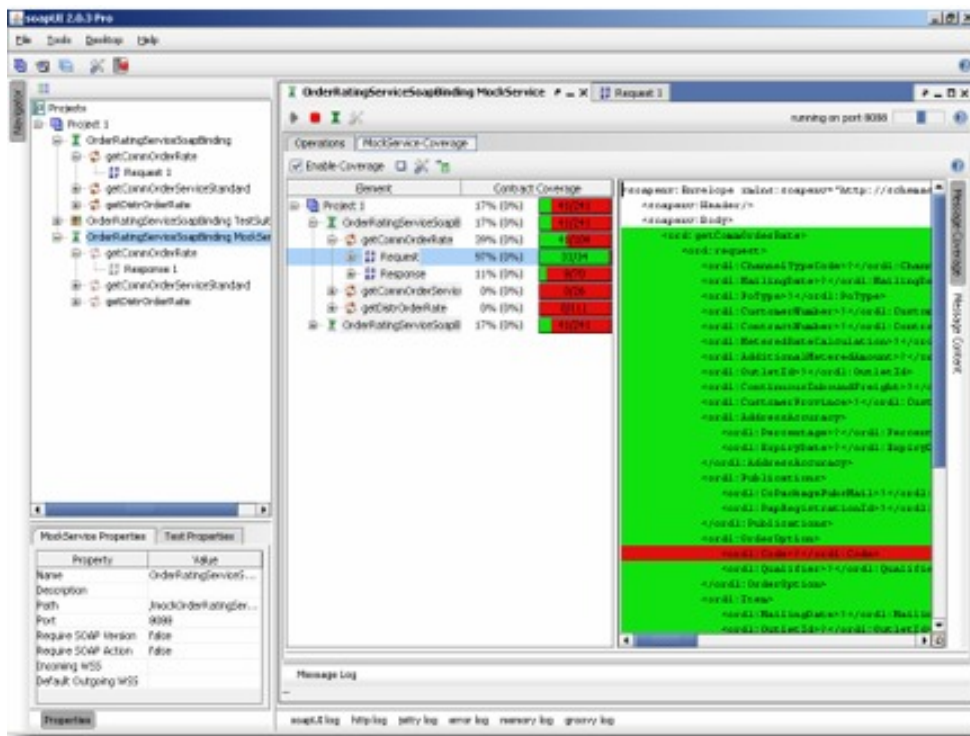


Figure 1

Strategy 5: "Collect important governance-related metrics and review them regularly."
Scope: Run-time

Modern governance platforms can capture enormous amounts of operational data. The sheer quantity of this information can be overwhelming. Unfortunately, when faced with a statistical flood of biblical proportions it's natural for administrators to simply ignore the data, unless there's an immediate problem that needs rectification. The lesson here is that gathering metrics isn't enough – you need to review and possibly take action on them. It's common for these metrics to contain intelligence that can be used to predict future issues. This means that astute governance experts may be able to prevent these anomalies before they even occur. To help enable this predictive, pro-active problem solving, make sure that you gather and review the information provided by your governance package. While many governance solutions are not known for the quality of their graphical capabilities, there are several best-of-breed business intelligence products that you can employ to make sense of the mountains of data that you're likely to generate.

Figure 2 illustrates how governance software can be used at runtime to monitor service level agreements (SLA). Since these contracts often carry a financial penalty for breaches of their terms, it's incumbent upon administrators to pro-actively monitor service performance.

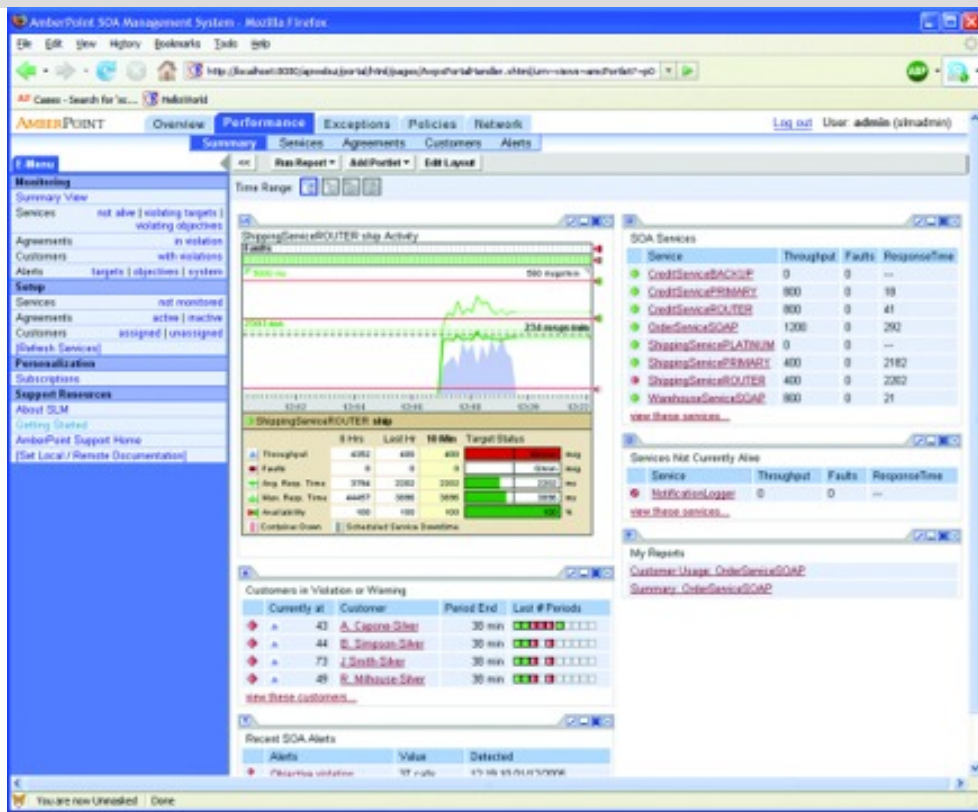


Figure 2

Strategy 6: "Track activity through multiple IT resource layers."

Scope: Testing & Quality Assurance/Run-time

While a service-oriented initiative is aimed at centralizing business logic into a collection of reusable, composable services, it's important not to forget that these services often simply place a facade over existing resources such as databases, application servers, objects, and so on. The result is an architecture made up of many "moving parts". With all these potential points-of-failure, it's natural that issues become more difficult to resolve – simply identifying the source of the problem can be an overwhelming burden. In many cases, the issue isn't the fault of the service, but the underlying resource.

However, users don't care where the problems initiate: they only want them solved (or prevented!). If only to address this reality, it's vital that your governance platform be able to monitor and interact with assets other than traditional Web services. By providing administrators with a comprehensive overview of all technology resources, a well-designed governance solution can help prevent problems from occurring, as well as assist in rapidly resolving any issues that do take place.

Strategy 7: "Break down the barriers between repositories and registries."

Scope: Run-time

There's a great deal of confusion between these two types of product; different definitions and delineations of responsibility abound. However, regardless of the exact description of these products, both have a role to play in an effective SOA implementation, including the design and run-time phases. The following describes how each product is typically utilized in these two critical phases.

Service registries answer these design-time questions:

- Where is the service?
- What is its purpose? (generally in brief)

Service registries answer these run-time questions:

- What is the service's version?

- Where is the service's contract?
- What policies are in effect for the service?

Service repositories answer these design-time questions:

- What is its purpose? (generally in more detail)
- What are the versions (including code) of the service?

Service repositories answer these run-time questions:

- Who's been using the service?
- What kind of responsiveness is the service providing?
- What's gone wrong with the service?

Finally, it's important to note that significant numbers of vendors are actively combining registries and repositories. Whether or not you deploy an integrated solution, it's wise to make sure that whatever investments you make are capable of addressing the responsibilities I just described.

Strategy 8: "When selecting a governance technology product, write a well-planned, formal RFP."

Scope: All phases

Although evaluating and selecting a solution from all of the available governance technologies can be a daunting task, there are time-tested patterns that you can leverage when making this momentous decision. To begin, it's vital that you know what you need: there is no substitute for homework and preparation. Follow the same discipline and processes that you did when selecting a database, application server, or other key infrastructure technology. Resist the temptation to employ a boilerplate RFP; make sure it reflects your organization's needs.

If you're not comfortable writing these kinds of documents, give serious consideration to hiring a vendor-agnostic external consultancy to write one for you. However, remember that if you're using outside help to design and/or implement your SOA, try to keep the RFP creation process separate from the technology vendor; it's easy to imagine for a serious conflict of interest. Next, to get vendors to take your RFP seriously (and respond accordingly), focus on quality, not quantity. Finally, once you've narrowed down your potential governance suppliers to a short list, it's common to ask these prospective vendors to participate in a pilot project or proof-of-concept.

Strategy 9: "Avoid governance tools that require code modifications."

Scope: Design-time and Run-time

In order to be effective, certain governance products necessitate code alterations such as incorporating special headers, configuration files, or linking with proprietary libraries. This requires complete developer compliance, which is difficult to obtain even under the most favorable conditions. If even one service is deployed without these added components, you won't be able to get a true picture of your environment, which can lead to erroneous decisions based on incomplete, inaccurate data. In addition to complicating the lives of your developers and reducing the likelihood of effective governance, these kinds of proprietary extensions can also seriously damage your chances of being vendor-agnostic. For all of these reasons, it's crucial that any governance solutions that you deploy do their work as non-intrusively as possible.

Strategy 10: "Make sure that the governance tool fits into your existing IT governance landscape."

Scope: Run-time

For most organizations, undertaking a service oriented initiative introduces a host of new toolsets, processes, and methodologies. Governance is just one of the added considerations, but it's important to note that this topic is not new for the majority of these enterprises. In fact, IT-focused governance technology has existed for many years: some of the best-known products include Tivoli, OpenView, and Unicenter. These packages are frequently used to monitor the health of servers, client computers, networks, and software applications. IT organizations have extensive knowledge of these solutions and rely on them to keep operations running smoothly.

As services become a larger part of your computing infrastructure, it's unwise to force your IT organization to learn and maintain completely different toolsets: ideally, your SOA governance tools should cleanly integrate with other IT

management platforms. Excessive complexity and training requirements lessen the chance that governance software will be used.

Conclusion

It's important to realize that while certain already-deployed SOA initiatives may appear to be successful in the absence of formalized governance methodologies and supporting technologies, it's quite likely that sooner or later events will transpire that challenge this perception.

For those enterprises that are now commencing their own SOA undertaking, it's wise to factor in governance considerations from day one, rather than waiting for a crisis to arise. Selecting necessary governance technology will be much more successful if the organization commits to using a formalized, vendor-agnostic RFP process.

No matter what governance technology is chosen, it's imperative that it be able to recognize and manage services constructed on any of the popular development platforms. In fact, many real-world services will be built upon traditional application components: the governance solution should be able to adapt to this reality. Given the close relationship between quality assurance and SOA governance, governance technology should be able to interact with relevant QA solutions, as well as team with mainline IT governance platforms.

By following a few, relatively simple SOA governance technology guidelines, the odds are definitely on the side of the progressive IT organization as it seeks to protect the substantial investments in people, process, and technology mandated by a service-oriented initiative.

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL



[Home](#) [Past Issues](#) [Contributors](#) [What is SOA?](#) [SOA Glossary](#) [SOA School](#) [SOA Books](#) [About](#) [Legal](#)

Copyright © 2006-2008
SOA Systems Inc.