

The SOA Magazine

Feature Article



What Every Developer Should Know About SOA Governance

by Robert Schneider

Published: January 19, 2009 (SOA Magazine Issue XXV: January 2009)

[Download PDF](#)

[Digg This](#) • [De.licio.us](#) • [Slashdot](#) • [Technorati](#) • [StumbleUpon](#) • [Google Bookmark](#)

Abstract: Governance is often the Achilles Heel of a service-oriented initiative, especially when it comes to how software developers treat their responsibilities in this important matter. Unfortunately, far too many enterprises witness an adversarial relationship between the developer and SOA's vital governance requirements. This article explores how easily things can go awry when software developers resist participating in governance. Happily, by investing in processes, people, and technology, the organization can mitigate the risk of this undesirable outcome. With these foundations in place, software developers can focus on applying proper governance techniques to the key deliverables created in any service-oriented project.

Introduction

Creating software for use in a service-oriented world requires a new perspective on design techniques and development technologies. Unfortunately, in the race to understand and implement these new methodologies and tools, important considerations such as governance frequently get overlooked. These types of oversights can end up as a significant reason for the oft-experienced gap between the promise of SOA (increased ROI on software investments, augmented organizational agility, and diminished IT maintenance burdens) and the reality of most SOA initiatives: a proliferation of duplicate services, unclear policies, and frustration.

This article aims to help the developer understand the importance of solid governance practices as part of the software development lifecycle. To begin, I provide a frank assessment of the penetration of service-oriented initiatives, followed by an examination of the uneasy state of relations between developers and the governance process in many organizations that actually have embarked on one of these undertakings. With these two concepts out of the way, I move on to an exploration of the kinds of groundwork that an enterprise must construct in order for its developers to have a fighting chance at satisfying governance requirements. Next up is a brief review of how governance affects the service-oriented lifecycle, followed by the relationship between governance and key project deliverables. I close the article with a special examination on governance and the software development process.

Developers and Governance Today

While SOA continues marching ahead, with increasing numbers of organizations using this approach for designing and developing mission-critical applications, it's important to realize that even in the best of cases, most of the IT budget is still consumed by traditional expenditures. These endless expenses include integration of existing solutions, developing one-off siloed applications, and maintenance. In addition, the relentless demands of tight timeframes continue unabated, regardless of architecture. As we'll soon see, the ramifications of these realities are felt far-and-wide, including within the seemingly isolated subject of SOA governance, particularly from the perspective of the developer.

Upon further inspection, it turns out that a relatively small number of organizations are utterly committed to a full service-oriented undertaking, which can be thought of as having the following distinct characteristics that

separate it from earlier architectural philosophies:

- **Business-driven:** Unlike many architectural initiatives, which are driven by and focused on technology, the rationale and justification behind a true SOA solution is to provide demonstrable business value, especially with regard to increasing ROI and organizational agility while reducing IT burdens.
- **Vendor-neutral:** This type of architecture is designed to be as independent as possible of proprietary, vendor-specific capabilities. This lets the enterprise remain as flexible as possible, selecting best-of-breed technologies without fear of vendor lock-in. Governance has an important and never-ending role to play in preserving this aspect of a well-designed SOA initiative.
- **Enterprise-centric:** Wherever possible, the software assets developed as part of a service-oriented effort are meant to be of use across the entire enterprise. This can have a dramatic impact on ROI, but it comes with a heavy governance responsibility.
- **Composition-centric:** One of the best measures of the success of an SOA initiative is the degree to which its services are composed into new solutions. While not every service is a good candidate for repeated composition, the goal is to foster composition wherever reasonable. Once again, effective governance is often the deciding factor that determines if this milestone will be achieved.

IT leadership is rightfully skeptical of any new design philosophy, and remains focused on the bottom line. To mitigate risk and control cost, many SOA initiatives begin with a proof-of-concept or pilot project. It's only after meaningful ROI has been demonstrated that these enterprises truly get on board the SOA bandwagon. While this is a wise approach, this viewpoint also has the tendency to introduce hidden governance shortcuts and side effects that impact developers.

A significant percentage of enterprises that have full-blown SOA engagements underway shortchange the vital topic of governance. For those that do make this important subject a priority, developer reaction has been decidedly mixed. The following are just a few of the complaints voiced by developers in these types of organizations:

- Unhappiness with what can be perceived as a "counter-agile" delivery strategy mandated by top-down SOA initiatives.
- Reluctance to face the overhead imposed by governance.
- Frustration at the loss of creativity and control typically mandated by effective governance.
- Disappointment with insufficient organizational support.
- Cynicism at mixed messages coming from management.

It's vital (especially for IT management) to note that these are legitimate grievances. Unfortunately, if these issues remain unaddressed, the path of least resistance for the developer is to simply ignore their governance responsibilities, either passively or actively. This abdication results in:

- Service proliferation
- Performance issues
- Confusion and duplication of effort
- Exponentially more difficult governance tasks



Figure 1: illustrates a well planned service inventory deteriorating into a chaotic mix of overlapping, duplicated services, destined to turn any architect's hair (or what remains of it!) gray.

These developer-driven issues are not academic: they negatively impact all layers of the organization, not just IT. These complications include:

- Reduced ROI
- Diminished organizational agility

- Increased IT burden
- Degraded service to customers
- A perception that SOA "isn't worth it!"

In fact, troubles related to governance have been the trigger that has halted a significant number of SOA initiatives. However, despite all of the damage that developers and their unhappy relationship with governance causes, it's not fair to place all of the blame squarely on this constituency. Unfortunately, far too many enterprises attribute any SOA disappointments as being exclusively caused by recalcitrant developers sabotaging well-intentioned governance efforts. In fact, I would argue that developer non-adherence to governance is a symptom of deeper problems. Mitigating these underlying issues would go a long way towards boosting developer governance compliance, which is what's covered next.

Organizational Prerequisites for Effective Governance

Developers can't be expected to implement a solid governance methodology and infrastructure on their own. The organization must lay the foundation, in the following ways:

- Develop and broadcast an overall SOA roadmap, including the business reasons for undertaking this initiative.
- Make investments in technology and methodology.
- Communicate a commitment to governance as part of the SOA implementation plan.
- Honest recognition and acceptance by management of the added costs and time impacts of governance.
- A commitment to training and support for the development team.

In terms of technology, it's unrealistic (and unfair) for an IT organization to expect developers to adhere to governance guidelines without any supporting infrastructure. Modern governance technology lets you:

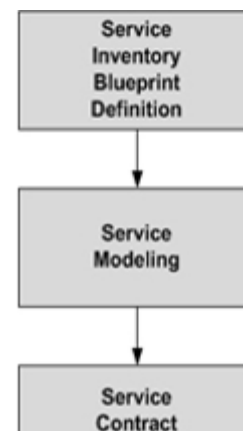
- Predict hotspots, version incompatibilities, business policy inconsistencies, and so on.
- Notify administrators when trouble arises (or is likely to arise).
- Measure the impact of a change to a given service.
- Track re-use, govern your inventory, and so on.

A governance software investment doesn't need to "break the bank". In fact, high-quality open source software is available for governance initiatives. It's easy to "try before you buy"; pilot projects and proofs-of-concept are great for this.

Considering how effective it can be, it's surprising that more enterprises don't make an investment in a Center of Excellence (COE). This provides a controlled, safe environment for analysts, architects, developers, and anyone else involved in the SOA initiative to learn and experiment. A COE should contain a realistic mixture of hardware and software (including relevant packaged applications and development environments). It's essential that you factor in the realities of multiple domains and/or cross-departmental concerns. A consultancy can help, but IT should stay involved, and in charge of the design and running of the COE. Finally, don't forget to include governance software as part of the COE!

Incorporating Governance into the Service Lifecycle

The most effective SOA initiatives introduce a thorough and all-encompassing software design, development and management process into an organization. Governance is an essential part of this lifecycle. The most successful enterprises propagate these procedures in the following ways:



- Communicate the reason for the SOA initiative, especially with regard to business justifications and expectations.
- Point out that this initiative necessitates a new style of development lifecycle.
- Train the team on the lifecycle, and their places in it.
- Allocate enough time for analysis - this has a major impact on governance.
- Recognize the need for new roles and responsibilities.

Figure 2: illustrates an example of a widely adopted lifecycle.

Each phase builds upon what's been accomplished previously. As we'll soon see, governance isn't listed as its own, separate facet in the lifecycle because it has a role to play in each of these phases. Finally, note that different organizations frequently use their own terminology for their own customized lifecycles.

Regardless of the site-specific vocabulary for a SOA undertaking, a wide range of new job skills, titles, and responsibilities come into play. Many of these roles directly or indirectly impact governance. Figure 3 illustrates just how many new responsibilities are likely to be encountered.

Of particular interest to developers is the new type of team member known as a governance specialist (shown on the far right of figure 3). These individuals are experts in governance processes, patterns, and technologies. This role is generally required during post-testing stages, at which point the governance of services, compositions, and entire inventory architectures comes to the forefront. However, as illustrated, governance specialists can also be required during any of the pre-deployment stages in order to provide guidance as to how modeling, design, or development-time decisions can impact future governance considerations.

Now that we've examined what the organization must do (in terms of prerequisites, methodology, and staffing) to employ successful governance, it's time to turn our attention to how governance interacts with some of the most vital deliverables found in a service-oriented project.

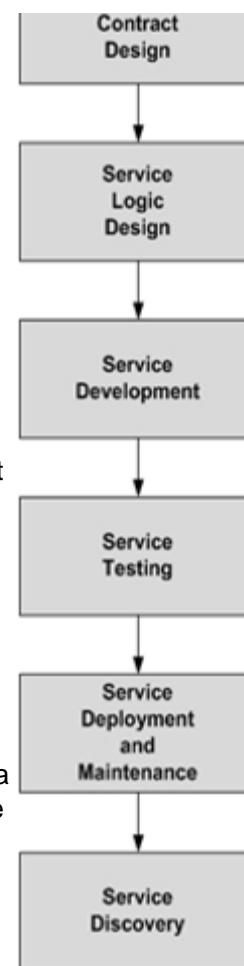


Figure 2

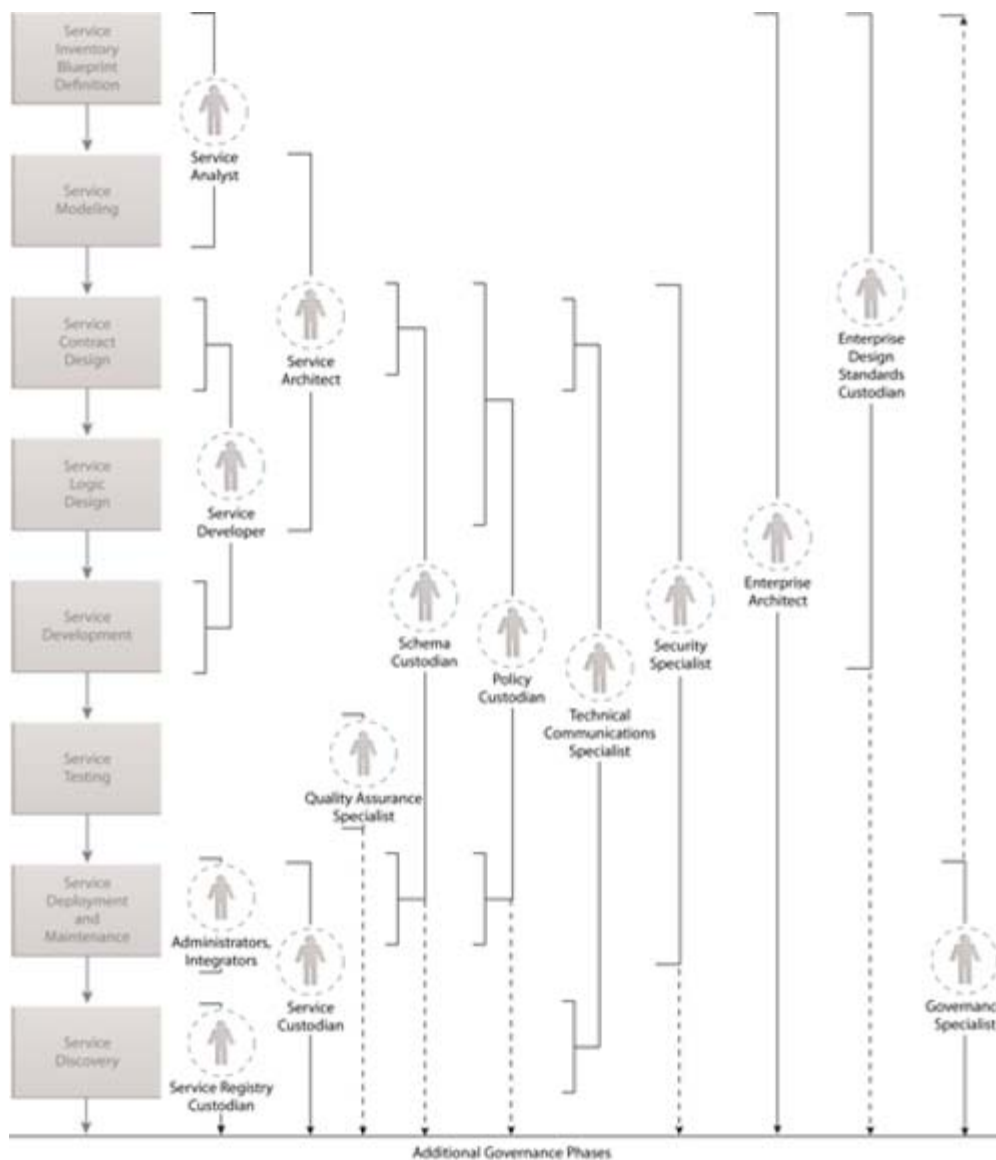


Figure 3

Shaping Key Deliverables with Governance in Mind

From the perspective of a software developer, it's natural that application logic represents the alpha and omega of SOA governance considerations. However, this isn't an accurate assessment of reality. In fact, application logic is just one of several key deliverables that have a close relationship with governance, including:

- Schema
- Contracts
- Policies
- Compositions

In the next portion of the article, we'll explore how governance interacts with each of the deliverables listed above, saving application logic for last.

Schema and Governance

Accurate, well-governed schema are an essential foundation for SOA, as well as for Web services even if a full service-oriented initiative is not in the cards. Schema impacts governance as follows:

- Schema extensions and overrides can complicate governance efforts.
- XML Schema alterations can have the same resonance as major relational database schema

modifications.

- When XML Schema is generated from a database schema, the database platform may provide a certain amount of governance infrastructure.
- However, automatically generated XML Schema can get out of sync with the database should alterations be made to the underlying table structures.

Since schema is so vital to effective governance efforts, there are strong arguments in favor of applying a centralization pattern. This approach, as described on the SOA Design Patterns website (http://www.soapatterns.org/schema_centralization.asp), lets you design and implement schemas independently from the service capabilities that utilize them to represent the structure and typing of message content.

Contracts and Governance

Contracts describe what the service will offer, and what it expects from consumers. These objects require significant governance analysis, planning, and maintenance. To begin, it's essential that architects and developers understand the impact of non backwards-compatible changes. Be particularly wary of renaming/removing operations; additive changes aren't quite as dangerous. Take advantage of Web service testing software to assist with WSDL refactoring. Finally, where applicable, consider applying patterns to support multiple concurrent versions of a contract, as described here (http://www.soapatterns.org/canonical_versioning.asp).

Policies and Governance

Policies let service designers and developers provide a wide variety of guidelines regarding service behavior. Special governance concerns include:

- The hierarchical nature of policies introduces the possibility of conflict.
- Policies are often assembled out of multiple assertions; there may be inconsistencies and other issues arising from this diverse assemblage.
- Policies can be attached to different portions of the Web service contract.
- Operator composition introduces complexity.
- It can be very difficult to get a single, authoritative view of policy landscape.

For all of these reasons, it might be wise to reduce governance risk by centralizing policies rather than maintaining them in individual services. The Policy Centralization pattern, shown in figure 4 and described in more detail on the SOA Design Patterns website, can be very helpful.

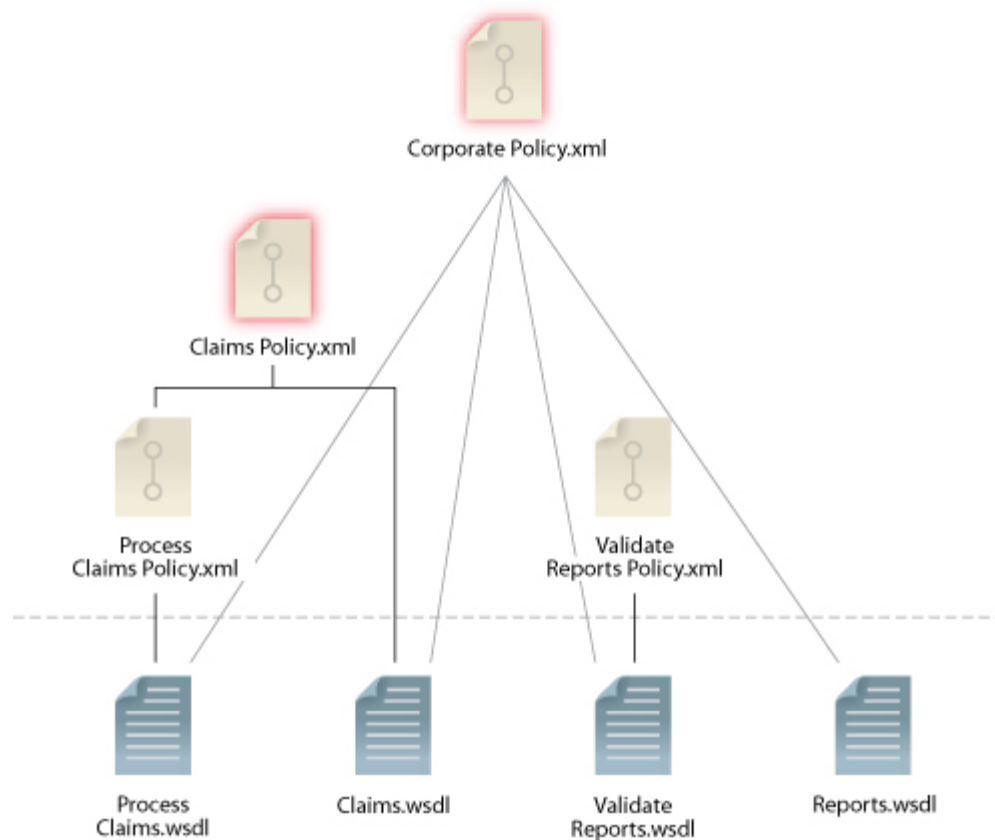


Figure 4

Policy assertions that apply to multiple services can be abstracted into separate policy definition documents or service agents that are part of an inventory-wide policy enforcement framework. As with many of the centralization patterns, this reduces the number of moving parts that need to be governed, but increases the potential for far-reaching havoc should governance failures occur.

Compositions and Governance

Recall from earlier that compositions are one of the primary reasons for undertaking an SOA initiative. Special governance challenges for these deliverables include:

- Compositions are made up of many moving parts. Downtime to any single part shuts down the entire composition. Faulty governance increases the likelihood of downtime.
- You may not even own all of the services that are part of your composition, making comprehensive governance difficult, if not impossible.
- You may not be able to predict all of the compositions that will be utilized.
- Compositions often make use of 3rd party technologies (such as Enterprise Service Buses (ESB), Message queues, and so on). These all must be governed as well.

These potential headaches highlight the need for solid testing and governance technology, combined with an effective, well-planned methodology.

Application Logic and Governance

It would be nice to report that software developers eagerly and earnestly embrace governance. Unfortunately, this is rarely the case. Cultural and other political rationales are often the hidden reasons behind resistance to governance efforts. To begin, standards are much more important in an SOA initiative, which by necessity places limits on developer creativity. This is especially true for developers whose primary experience has heretofore been writing siloed applications. In a world where SOA has been implemented, developers find themselves spending much less time constructing entire applications from scratch, and much more time assembling compositions of already-existing services into new solutions.

However, in spite of these constraints on ingenuity, most developers sincerely want their SOA initiative to succeed. As described earlier, it's imperative that the enterprise take the time to set up and document a well-

planned set of processes that reach all the way into software development and beyond; too many organizations expend all their efforts on service analysis and modeling, thereby neglecting to give proper thought to these subsequent responsibilities.

Thorough, easily accessed service profiles can go a long way towards helping developers gain a comprehensive view of all the services at their disposal. Software developers who help create and maintain these service profiles should be rewarded for their contributions. At the same time, organizations should consider placing access control restrictions on sensitive design documents, source code, and so on: developers should be encouraged to use the service profile and service contract as their primary means of learning about what a service offers.

Some organizations adopt a carrot-and-stick approach to coaxing developers to follow governance guidelines. Here are just a few of the available options, all of which are greatly helped by creating a flexible compensation plan that is "SOA-aware":

- Recognize that "number of lines of code written" isn't (necessarily) a valid productivity metric anymore.
- Reward developers for reusing others' work.
- Conversely, reward developers for writing reusable services when no applicable services exist.
- Penalize developers who unnecessarily create new services (or otherwise violate governance standards).
- Conversely, don't punish developers for the overhead (and inevitable schedule impacts) mandated by adherence to solid SOA design and governance methodologies.
- Enforce governance compliance for outsourced development teams. Nothing irritates internal staff like seeing one set of behavioral expectations enforced for employees and a different (usually more lenient) standard for consultants.
- Maintain appropriate staffing levels in related areas (e.g. architecture, testing/QA, and so on) to facilitate developer productivity.
- Recognize the need for new skill-sets (e.g. service and policy custodians, technical communications specialists, etc.) in support of SOA and related governance.
- Educate development management personnel about the likelihood of cross-departmental support requirements.

Conclusion

Many organizations embarking on their service-oriented journey make the mistake of ignoring the key relationship between their software developers and the governance responsibilities inherent in any SOA initiative. These enterprises often learn the painful lesson that developers' participation and support of governance is not optional. Failed governance is often the underlying cause of management disappointment with SOA.

Fortunately, with the proper mixture of people, processes, and technology, a progressive organization has a good chance of obtaining the vital support of its developers. When software developers are on board, governance has a chance to succeed. When governance is effective, it's much more likely that the service-oriented initiative will deliver on its promise.

The figures in this article were copied with permission from books in the Prentice Hall Service-Oriented Computing Series from Thomas Erl (Copyright 2004-2009, Prentice Hall, www.soabooks.com).