zapthink white paper

GOVERNANCE, QUALITY, & MANAGEMENT

THE FOUNDATIONS OF SOA

ZapThink, LLC • 108 Woodlawn Road • Baltimore, MD 21210 • info@zapthink.com • www.zapthink.com

GOVERNANCE, QUALITY, & MANAGEMENT

THE FOUNDATIONS OF SOA

July 2008

Analyst: Jason Bloomberg

Abstract

Service-Oriented Architecture (SOA) is an approach to organizing IT resources to better meet the changing needs of the business. While many organizations are somewhere on their SOA roadmaps, many such organizations face challenges when planning the underlying infrastructure that will support their SOA implementation. One reason for this challenge is that there are three core infrastructure areas that are jointly essential to the success of separate, but overlapping SOA effort: governance, quality, and management.

Governance means creating, communicating, and enforcing the policies that apply to the behavior of IT and its users. Quality is a measure of how well working systems meet the needs of the business. Management focuses on how well those systems meet performance, security, and other non-functional requirements for working software. In the context of SOA, however, these three sets of capabilities begin to merge.

To provide the business agility benefit that is the core business motivation for many SOA initiatives, governance, quality, and management need not only apply to the design time and run time phases that traditional software projects exhibit. In addition, SOA requires these capabilities apply to the change time phase as well, where organizations reconfigure and recompose Services to meet changing business needs. As a result, the governance, quality, and management challenges that SOA presents go beyond traditional IT concerns.

I. Executive Summary

SOA is a set of best practices for organizing and managing IT resources as flexible, business-oriented Services.

Service-Oriented Architecture (SOA) has become the predominant architectural movement in IT organizations around the world, and yet, there remains extensive confusion over the nature of SOA and how best to implement it. Despite what numerous platform and integration vendors espouse, SOA is not something you buy—it is something you do. Fundamentally, SOA is a set of best practices for organizing and managing IT resources as flexible, business-oriented Services.

Services abstract the underlying complexity of the IT environment, providing greater power and flexibility to the business. SOA has the power to increase competitiveness in the face of today's ever-changing business environment, and once businesses realize the transformative power of this critically important business concept, they will be in the position to deal with ongoing, often unpredictable, change.

Today, the majority of enterprises and government organizations are somewhere on their SOA roadmap. While business agility is the most strategic business benefit of SOA, many organizations leverage SOA to reduce the cost of integration, increase asset reuse, and improve customer visibility and overall business transparency, as they build the business case for agility.

While effective SOA roadmaps begin with the business problem, which leads to the architectural design, governance framework, and an iterative plan, the infrastructure organizations need to implement SOA also appears early in each SOA roadmap. The infrastructure challenge for SOA, however, rarely centers on middleware—after all, most large organizations already have plenty of middleware. On the contrary, the core infrastructure challenges of SOA are in the areas of management, governance, and quality across the SOA lifecycle.

The Infrastructure Challenge of SOA

Loose coupling is the core architectural principle for building Services in the context of SOA, because it goes to the heart of the business agility benefit.

At the core of any SOA implementation is the Service abstraction. Services represent IT capabilities and data to the business so that they can compose those Services into Service-Oriented Business Applications (SOBAs), which are composite applications that implement business processes. To support the flexibility that Services require, the SOA infrastructure must enable such Services to be loosely coupled, which means that Service providers (the software that provides the Services) and Service consumers (the software that consumes the Services) can be independently created and modified without breaking the other.

Loose coupling is the core architectural principle for building Services in the context of SOA, because it goes to the heart of the business agility benefit of SOA. When architected properly, loosely coupled Services are composable, reusable, and dynamic. It becomes possible, therefore, to leverage such Services

in diverse, often unpredictable ways to meet continually changing business requirements.

There's no question that today's enterprise IT environments are enormously complex, and it is that complexity more than any other cause that leads to the inflexibility that the business wishes to address. And yet, SOA does not actually eliminate complexity—it *abstracts* the complexity, providing a flexible, simplified set of Services to the business that overlays the unavoidable technical complexity. At the core, IT has always dealt with underlying complexity, through the power of abstraction. In the world of IT, abstraction is a way to simplify the complexities of the technology with simple, yet powerful representations. Beneath the abstraction layer, there remains the complexity that IT deals with today. But due to the power of loose coupling, the Services available to the business provide whatever value the business requires from them.

To successfully share Services, however, it is essential that organizations be able to find them and trust them.

Much of SOA's value proposition also depends upon the successful reuse, or sharing of Services. To successfully share Services, however, it is essential that organizations be able to find them and trust them. Trust requires assurance that they meet business requirements, including both functional and performance requirements. Additionally, organizations must manage Services and SOBAs while they are in production, tracking performance and availability, diagnosing and correcting problems as they occur, reporting on SLAs, and all other aspects of the behavior of the Services.

Governance, quality, and management: the foundations of SOA infrastructure

As with any abstraction, however, there is no magic. To build such powerful services requires sophisticated governance, management, and an overall focus on quality. After all, while it's simple to talk about loose coupling, such loose coupling depends upon high quality, well-managed and governed SOA infrastructure. In fact, companies will not be able to break down the IT/business disconnect until they solve this underlying governance/quality/management (GQM) convergence problem in the context of a heterogeneous environment.

Perhaps one of the greatest challenges to dealing with this problem is the fact that no single-platform approach is up to the task of providing governance, quality, and management in a heterogeneous environment. The single-platform perspective is to move capabilities onto the platform; SOA, however, supports the notion that heterogeneity is here to stay. Software remains where it is, and it is up to the infrastructure to provide the necessary GQM capabilities in this inherently diverse environment.

Yet while the marketplace recognizes each of these critical areas of SOA governance, quality, and management, only recently have we realized that each of these distinct market segments in their own right are really different aspects of same problem: making the challenge of loose coupling a reality. In combination, SOA governance, quality, and management are all part of the same spectrum of capabilities that can make the perceived difficulty of loose coupling in a continuously changing IT and business environment a reality.

II. SOA Governance: Aspect of IT Governance

To understand the full spectrum of capabilities that GQM offers, it's important to understand each capability separately. First, the definition of governance is creating, communicating, and enforcing policies. Governance is the key to balancing executive control with employee and customer empowerment across the enterprise, or in the case of IT governance, across IT. At its most basic,

governance requires establishing and enforcing how a group of people agrees to work together. Specifically, governance is the establishment of chains of responsibility, measurement approaches to gauge the effectiveness of governance activities, policies that guide the organization, control mechanisms to ensure compliance with those policies, and communication among all parties. Governance delineates who is responsible for making decisions, what decisions the organization should make, and policies for making those decisions in a consistent manner.

IT governance applies governance to an IT organization, including its people, processes and information. SOA governance is a subset of IT governance that puts key IT governance decisions within the context of the Service lifecycle. SOA governance addresses aspects of this lifecycle such as planning, publishing, discovery, versioning, management, and security of Services. One of the primary goals of SOA governance, therefore, is the effective management of this lifecycle.

Governance is an essential part of any SOA implementation.

Governance is an essential part of any SOA implementation, because it ensures that the organization applies and enforces the policies that apply to the Services that the organization creates as part of its SOA initiative. But more importantly, organizations can leverage SOA best practices to represent policies broadly in such a way that the organization can achieve better policy management, flexibility, and visibility into policy compliance across the enterprise.

Governance is especially important for SOA for several reasons. In particular, since Service consumers and providers are loosely coupled, different people in different departments or organizations are generally responsible for developing and managing them. As a result, IT organizations require extra coordination to operate their SOA implementation successfully. For many SOA implementations to be successful, multiple applications must share common Services, which requires coordination. Such coordination is essentially a governance issue.

SOA Governance in Practice

One of the more important roles for SOA governance is guidance: it guides the establishment of policies for the design, development, configuration, and composition of Services and how those Services will change over time. Governance also serves to establish agreements between providers and consumers of Services, coordinating the information that tells the people who operate the consumers what they can expect and the individuals responsible for the providers what they're supposed to provide.

SOA governance helps address many design time SOA issues: What Services are available? Who can use them, and how should they use them? How reliable are the Services? How long will there be support for the Services? When and how might the Services change? What if two consumers want the same Service to work differently?

SOA governance also serves a critical role in the reuse of Services. It helps to answer such questions as: Who's going to pay to develop shared Services? Will development and Service composition teams actually reuse them? How will everyone agree on the functionality for a reusable Service? Who's responsible for a shared Service if there's a problem with it? Who gets to decide how to version a shared Service, and what are the policies for such versioning?

Note that most of the questions above concern human behavior more so than the behavior of systems. In fact, governance is more of a political problem than a technology problem. Governance mostly focuses on ensuring that everyone is working together and that separate efforts are in alignment. Governance does

not determine the results of decisions, but it does guide what decisions an organization must make and who should make them.

Governance introduces challenges into an organization as well. After all, nobody likes to be governed! Furthermore, SOA governance can become a scapegoat for SOA problems. As a result, one of the key challenges for SOA governance is using it judiciously to make SOA work better without letting concerns about governance overwhelm other parts of the initiative.

SOA governance features

It's important to note that SOA governance is a broad set of coordinated practices, both human-centric as well as automated. Furthermore, SOA governance applies to the full Service lifecycle: design time, run time, and change time. The design time phase includes planning, designing, developing, testing, publishing, discovering, and deploying Service implementations. During run time, the phases include using, managing and supporting live Service implementations. Change time includes the configuring, composing, versioning, deprecating, and retiring Services phases.

During design time, the most fundamental aspect of SOA governance is overseeing the creation of Services. The Service creation team must identify, describe, scope, and design the Services. Because Services are abstracted interfaces to a wide range of functionality and data in the organization, it's rarely obvious from a developer perspective what should be a Service. Remember, the architecture should guide the creation of Services, and governance is the mechanism for conveying the architect's intent to the Service creation team.

Another key aspect of design time SOA governance is the use of a SOA registry/repository. It's important for the Service creation team to know when to look in the registry/repository for existing Service artifacts, when they should reuse an existing Service instead of building a new one, and then when to publish a new Service into the registry/repository. All of these activities fall under the scope of SOA governance, and in many cases, the registry/repository itself helps to manage and automate such activities.

Governance also applies to the usage of Services during run time. It is important for the IT organization to have policies in place that govern the consumption of Services, including which consumers should access which Services, as well as when or how those consumers access available Services. Such policies typically span several Services, and go beyond the information in each Service contract. In many cases, security requirements are part of such Service usage policies.

Perhaps the most important role for SOA governance, however, is during change time. After all, SOA success depends upon change time governance more so than governance of the other phases. Most of the activities that take place during change time involve the configuration and composition of Services—changes at the metadata level that do not involve any changes to the underlying code. It is important, therefore, for the organization to have policies that apply to such configuration and composition activities. Without such governance, organizations can run into issues with their Services that have nothing to do with the functionality of the Service implementations themselves. Taking a code-centric perspective on Services, therefore, will entirely miss this critical area of governance.

Most of the activities that take place during change time involve the configuration and composition of Services—changes at the metadata level that do not involve any changes to the underlying code.

Governing Service change and ownership

Service versioning and deprecation are also critical areas for SOA governance, and organizations who don't plan ahead for these activities often find

themselves scrapping their SOA efforts and starting over. Remember, the point of SOA is to build for change, so building Services with change in mind is a critical SOA best practice. After all, the Services reflect the business, so as the business changes the Services must change accordingly.

The challenge, of course, is that of loose coupling: How do you change Services without disrupting the consumers of those Services? There is no one right answer, of course, since the behavior of Services and Service consumers depends upon the various, changing requirements that apply to them. Instead, it is important for the architecture team to establish policies for versioning and deprecation that apply to both consumers and providers, and then leverage SOA governance to communicate and enforce those policies.

Well-governed Service versioning meets the goals of loose coupling. It enables users satisfied with an existing Service to continue using it, if the policy allows, yet allows the Service to evolve to meet the needs of users with new requirements. In other cases, the policy requires the consumer to update itself when a Service version changes. After all, even with Service versioning, a consumer cannot count on a particular version of a Service to be available and supported forever. In essence, when a consumer starts using a Service, that usage creates a dependency on that Service, and it's essential to manage that dependency.

Service ownership is another important area of SOA governance. When multiple SOBAs leverage a Service, who is responsible for that Service? Traditional IT shops depend on a staff reporting structure and organize their finances around business operations. For Services that one department creates controls, and uses, there is little problem. However, the Services and SOBAs in the context of SOA often don't follow an enterprise's hierarchical reporting and financial structure, creating both gaps and overlaps in the associated IT responsibilities.

As a result, organizations require increased levels of cooperation among multiple departments to share the burden of developing and supporting common Services. To facilitate this cooperation, many such organizations establish a cross-organizational standing committee that, in effect, owns certain shared Services and manages them. Such a group of shared Services that have a common business context is a Service domain. By leveraging governance to provide Service domains with committees with established responsibilities and budgets, organizations can resolve the political issues surrounding shared Services.

III. SOA Quality: Across the Full Service Lifecycle

Quality means far more than simply reducing defects. Fundamentally, quality means building something that meets the requirements of its users, now and into the future.

While governance concerns creating, communicating and enforcing the policies that are important to the organization, quality focuses on whether systems meet the requirements set out for them. Quality means far more than simply reducing defects. Fundamentally, quality means building something that meets the requirements of its users, now and into the future. Being defect-free is a necessary, but by no means sufficient criterion for a quality product. Software quality is no different. While many software quality assurance efforts focus on eliminating bugs, the bug-hunting process is only the starting point for software quality.

In an ideal world, quality assurance (QA) personnel would simply take the requirements document, use it to build a test plan, and run tests against that plan. Once the project passes all the tests, it's ready to go live. But in the real world, requirements continue to evolve, both during projects as well as once the

projects are complete. And there's nothing worse than evolving requirements for throwing a wrench in the most carefully laid QA plans.

Furthermore, quality extends beyond the functional (what the system is supposed to do) to the non-functional (how the system is supposed to behave). For example, the performance and security requirements for a system are every bit as important as its functionality. In the context of SOA, managing such non-functional requirements for Services is both a part of the quality challenge as well as the SOA management challenge.

The importance of full-lifecycle quality

Environments of continually changing business requirements, of course, are the perfect breeding ground for SOA. SOA leverages a metadata-driven Service abstraction to provide greater power and flexibility to business users, with the clear purpose of enabling IT to respond to changing requirements in an agile manner. This core agility benefit of SOA collapses like a house of cards, however, if the Services or the applications that consume and compose them are of poor quality or behave in an unpredictable manner.

Not all organizations require such full-lifecycle quality practices, especially when they have no particular requirement for agility. The defining moment for stronger quality practices arises when there is either a high rate of change in the underlying systems, or a high degree of complexity and change in the business rules or requirements.

The SOA story touches upon many challenges like these—including loose coupling, trust, governance, and management, to name a few. But in the final analysis, SOA quality becomes the primary organizing principle for actually getting this stuff to work—what does it actually take to enable IT to meet the changing needs of the business. SOA quality becomes the thread that enables the business to trust technology to provide the agility it requires to meet requirements now and into the future.

SOA quality during change time

Perhaps the greatest SOA quality challenge, however, involves maintaining quality throughout the Service lifecycle, especially once the SOA implementation is in place. The problem is, the more mature the SOA implementation is—that is, the better the Service abstraction maintains an agile separation between business users and the underlying IT capabilities—the more impractical traditional QA approaches are likely to be. In many of today's IT shops, there are separate, identical QA and production environments. QA personnel can load any new or changed code into the QA environment, and test it to their heart's content before giving it the thumbs up for promotion of changes to the production environment.

In a mature SOA environment, however, it's practically impossible to maintain a useful duplicate of the running system, because Services, configurations, and associated metadata continually change. As a result, maintaining a parallel QA environment rapidly becomes an exercise in futility. Furthermore, it's clear that SOA quality is a collaborative effort that involves many different participants across the organization.

The solution to this quality conundrum is to test new and changed Services and Service configurations in the live, production environment. The only way to ensure that all aspects of the new configuration continues to meet the requirements set out for it is to run test messages through production Services. Now, saying you should test in production is tantamount to proposing rewiring

your house with the power on—it's possible, but you have to be especially careful, know what you're doing, and plan ahead. In the case of SOA, planning ahead means that Services (as well as Service consumers) must be able to support a testing mode as a matter of policy.

There are a few important notes about running the QA process through a production system. First, the QA process is policy-driven. As a result, the testing process is itself Service-oriented, which goes a long way to satisfying the agility requirement of SOA. Second, it's resilient. Even if a fully tested Service still fails in production, the QA process responds in a way that minimizes the business impact, and thus maximizes the loose coupling of the Services. But most importantly, SOA quality requires planning ahead. Architects must plan for test modes and deprecation policy support as an essential part of designing Services.

The meta-requirement of agility

If a QA team treats a SOA project as though it were a traditional software project, its quality will likely suffer, because traditional projects don't have the business requirement of agility—what ZapThink calls the "meta-requirement" for SOA, namely the requirement that the implementation of the architecture must be able to satisfy future requirements even as they continually evolve. A core challenge of SOA, after all, is building for change. If you had good reason to believe today's requirements were permanent, you probably wouldn't bother with the expense and complexity of SOA. Testing for today's requirements without testing for this meta-requirement leaves an enormous hole in the QA process.

If you had good reason to believe today's requirements were permanent, you probably wouldn't bother with the expense and complexity of SOA.

However, it doesn't make sense to expect that the IT organization would be capable of building systems that could deal with entirely arbitrary change, since such a requirement would be prohibitively expensive to satisfy. Instead, each organization will have to decide for itself precisely how much they can invest to properly scope the level of flexibility they require their SOA implementation to have. While deciding on your agility meta-requirement is an essential part of your SOA planning, even more important for the long-term success of your SOA initiative is in ensuring that your SOA implementation conforms to that meta-requirement over time. In other words, guaranteeing that your SOA implementation meets the needs of the business over time is a core measure of quality, and as such, SOA quality assurance must go beyond traditional software quality management and address the meta-requirement of agility.

As the enterprise architecture team sits down to plan the SOA initiative, they should be asking questions about the specific levels of agility the organization requires from the SOA implementation. In particular, they should discover the answers to the following questions:

- What are the business requirements for Service reuse? Is the business looking to achieve a certain cost savings or other metric for reuse?
- What are the requirements for loose coupling? Break this question down into multiple levels: semantic, data, message protocol, wire protocol, etc.
- What is the user context for the implementation? For example, how many users will the Services need to support? Are they all internal users, or are you allowing users from outside your organization? What users will be allowed to make configuration changes, and under what circumstances?
- What are the metapolicies? In other words, what policies are in place that govern how the organization should be able to create and enforce policies?

It's important to note first of all that the end product of each of these questions should be a set of policies. Secondly, it's also significant that questions like these don't focus on requirements for specific functionality or behavior of the software you're building as in a traditional project. On the contrary, these questions all dwell on issues of agility—just how agile your organization wants to be, but possibly even more importantly, in what areas is it OK to be less than fully agile. Properly scoping the constraints on the SOA initiative that your organization accepts can save substantial money and time, and can also lead to framing the discussion of change-time quality policies.

IV. SOA Management: Beyond Web Services

In order to realize the benefits of SOA, companies must transition their systems from existing inflexible architectures to SOA in a manner that does not impede the ongoing necessary functionality of the technology. Yet, the act of rearchitecting is not sufficient by itself to guarantee that the resulting business Services will meet the needs of the business. The enterprise also must have management infrastructure in place that can support the monitoring of Services performance as they move the Services into production as well as once the Services are available for public consumption. In fact, Service performance is only one example of the wide variety of characteristics the enterprise must monitor.

Companies must implement a SOA management infrastructure that can establish and maintain the connections between the software on the one hand and the Services on the other.

In order to encapsulate the underlying software components and systems with loosely-coupled Service interfaces and then compose these fine-grained Services into coarse-grained business Services and SOBAs, companies must implement a SOA management infrastructure that can establish and maintain the connections between the software on the one hand and the Services on the other. It's also important for IT operations managers to understand that the increased flexibility SOA affords will necessitate new metrics for measuring the effectiveness of IT operations.

In the IT world, flexibility means an increased rate of change. As a result, IT requires a strategy that mitigates this faster pace by applying the principles of loose coupling, separation of concerns, and centralized management to their IT environments. Furthermore, IT operations must be able to leverage SOA to improve their IT management overall. This tight interrelationship between architecture and operations is one of the organizational changes that SOA introduces in the enterprise. Enterprise Architects and IT operations personnel must now communicate with one another, and establish a basis for ongoing problem resolution.

Definition of SOA Management

SOA management provides the link between design time governance processes and traditional operations management systems. SOA management focuses on run time management capabilities in the SOA context, such as performance management, problem isolation and run time policy enforcement. In essence, SOA management controls the operational aspects of the Services within the organization.

SOA management starts with Service management, including application service level management, transaction workflow management, business performance management, and Services and applications security capabilities, as well as root cause analysis. However, SOA management goes well beyond management of Services. Specifically, SOA management also requires an increased ability to manage both business and IT processes, including business process monitoring

change and release management, configuration management, availability management, security management, and role management. Furthermore, SOA management also provides for the management of the supporting IT environment, including middleware management, systems, storage and network management, virtualized IT infrastructure management, and security for users, data and infrastructure.

SOA management also provides end-to-end SOA policy management, which includes the automatic validation of policy conformance at design time and the distribution and enforcement of policies at run time across various enforcement points. SOA management also facilitates the creation and communication of service level agreements (SLAs) between Service consumers and providers.

SOA monitoring is also an important SOA management capability. It's important for organizations to receive alerts on performance issues for Services in order to resolve performance and availability issues as they occur, if not before. SOA management also typically enables the aggregation of performance data across Service implementations and their operations, and also typically offers Service dependency mapping and diagnostics into the performance and operation of composite applications that are composed of Services.

SOA Management vs. IT Service Management

Early in the development of Web Services-based SOA, enterprises focused on simply managing the service interfaces themselves—in particular, Web Services interfaces. It soon became clear, however, that managing Web Services and managing the overall SOA implementation were different, although overlapping problem areas. Loosely coupled Services, after all, are simply interfaces; effective management of such interfaces necessarily requires management of the underlying infrastructure. As a result, in many ways, SOA management has more in common with IT Service Management than it does with Web Services management.

IT Service Management (ITSM) is a process-based practice that aligns the delivery of IT capabilities ("services" with a lower-case "s") with needs of the business. ITSM has changed the way many IT shops manage their capabilities from managing IT as stacks of individual components to focusing on the delivery of end-to-end services following a set of best practices collected into the IT Infrastructure Library (ITIL). ITSM differentiates how IT delivers systems management functions. By aligning with ITIL, it's possible to implement a comprehensive and integrated overall systems management solution in a Service-oriented environment.

SOA management, in fact, presents many ITSM challenges. In order to provide for the loose coupling at the service interface, as well as for IT resources, it's important to manage and prioritize quality of service over a range of dynamic resources. SOA management must also deal with the increased complexity of deploying interdependent services. Deploying such services rapidly increases the complexity and cost of management.

The loose coupling of Services, in fact, makes it harder to determine and isolate problems when they crop up. Keeping track of the relationships and dependencies among Services is as important as keeping track of the Services themselves. While loose coupling provides flexibility, governance helps to ensure the reusability of Services, which is also a challenge for traditional systems management. Higher flexibility leads to higher rates of change, while loose coupling suggests more things to monitor and more relationships to track.

IT operations must also deal with version control and change management of Services, as well as managing the proliferation of overlapping Services. Customers need good change management, release management and availability management processes, regardless of the status of their SOA initiatives. Within SOA environments, however, the Services layer introduces a higher capacity to change implementations. One of the key values of loose coupling is that helps organizations change the underlying implementations of the Services more gracefully than before.

SOA also introduces the metadata-driven composition of Services. Managing these virtualized flows requires the control and tracking of resource utilization for IT financial management, as well as the identification and coordination of problem resolution across organizational boundaries. Managing distributed Services also includes managing access control for Services, applications and data, and provisioning of the associated identities, possibly requiring the federation of security credentials.

V. The SOA Governance/Quality/Management Triangle

Organizations that are on the path to SOA adoption know full well that the technical challenges of building and exposing Services alone are less significant than the hurdles of building loosely coupled, abstracted business Services that they are able to compose into SOBAs that implement continually changing business processes. While SOA abstracts the complexity associated with heterogeneous, point-to-point integration and tightly-coupled application logic, it introduces a different kind of complexity: the management of distributed, loosely coupled, and dynamically composable Services.

To respond to this new form of complexity are a range of SOA infrastructure solutions: management solutions that isolate failure and provide mechanisms for abstracting end-point differences, quality solutions that provide mechanisms for assuring the propagation of changes in environments of significant change, and governance approaches that provide oversight into the development of Service-oriented systems, mitigation of change and version management issues, and enforcement of policies core to the operation of the business as a whole.

As this paper has illustrated, these critical areas of SOA governance, quality, and management are not truly distinct solutions in their own right, but rather they are different aspects of same problem: making loose coupling a reality. In combination, SOA governance, quality, and management form a triangle of capabilities that can make the perceived difficulty of loose coupling in a continually changing IT and business environment a reality. The relationships among SOA governance, quality, and management are illustrated in the triangle below:

The SOA Governance/Quality/Management Triangle

Source: ZapThink LLC

It's important to note that in the figure above, there are specific interrelationships between each element of the triangle. To fully understand the GQM triangle, therefore, its important to explore these relationships.

The connection between SOA governance and quality

First, it's important to understand the relationship SOA governance and quality. The challenge of maintaining continuous quality in a continually changing system is an aspect of maintaining effective governance, and as such, the combination of SOA governance and quality tooling and approaches serve to make that problem more manageable. SOA quality tools provide feedback to governance systems by indicating how changing policies impact the overall system, and likewise, governance systems and approaches feed into the quality lifecycle by providing continually changing constraints that impact Services at design and run time. This quality-governance feedback loop furthers the realization of loose coupling by making sure that any design time change has no impact on running systems, and that Service consumers do not need to hardcode governance rules.

Organizations must manage and govern change effectively so that no change has a chaotic effect on the complex environment of Services.

Furthermore, organizations must manage and govern change effectively so that no change has a chaotic effect on the complex environment of Services. This challenge means not only catching and preventing failure through management approaches at run time, but also staging and testing those changes through run time quality and testing approaches. In addition, design time governance also requires that companies enforce Service development practices before uncontrolled Services permeate the IT environment. Effective integration between governance and quality approaches makes that enforcement possible by providing the visibility into deviations from established policy and methods for limiting the spread of Services that are non-compliant.

The connection between SOA governance and management

Likewise, there is a connection between SOA governance and management that facilitates loose coupling. SOA management offerings can address aspects of policy enforcement, rules-based routing and decision making, and exception handling. As such, SOA management tools can enforce policies at run time that governance tooling manages at design time. While SOA governance tooling such as registries and repositories serve as systems of record to manage Service

metadata, SOA management approaches ensure that Services in production comply with the policies in effect. SOA management tooling can also detect and prevent the occurrence of rogue Services and inspect Service interaction, further enabling the value of SOA governance approaches. In addition, run time SOA management helps with change-time governance issues by enforcing governed and approved changes in the distributed environment while minimizing quality and performance issues.

Furthermore, effective SOA governance requires effective management to provide the visibility run time systems require to feed back into the governance process. This management-governance feedback loop not only helps companies govern their overall SOA efforts, but also provides effective control and management without overly constricting the agility of the architecture. This feedback loop helps to guarantee loose coupling by making sure that business requirements for change don't have an adverse impact on the behavior of the whole system, by decoupling the logic of the business from the implementation as it exists at that point.

The connection between quality and management

The final aspect of this combined set of capabilities is the connection between SOA quality and SOA management. It makes very little difference if one Service works in isolation if some aspect of how it participates in Service compositions, or how some change to metadata impacts the performance of the system as a whole. In essence, unit testing an individual Service implementation is wholly inadequate to determining whether the Service actually performs in a composite environment of metadata-controlled Services.

The only way to effectively ensure SOA quality is to do so continuously, measuring the quality of all related meta-data, composition logic, policy, and underlying schemas continuously in production.

The only way to effectively ensure SOA quality is to do so continuously, measuring the quality not only of a discrete, atomic Service, but also all related metadata, composition logic, policy, and underlying schemas continuously in production. However, within the context of SOA the architecture *is* the business, and since the business is continuously changing, a QA model that requires duplication of the environment in order to ensure quality will be enormously expensive, impossible to manage, and ineffective.

Furthermore, there is a management-quality feedback loop that exists between tools and approaches to management that provide visibility when systems are approaching an undesired state and mechanisms that allow incremental testing and quality management of the system as a whole. This feedback helps to guarantee loose coupling by making sure that any Service-related changes don't break things, which is a fundamental requirement of loose coupling.

While many existing SOA management tools on the market handle run time SOA management, it is not the intention of these tools to handle quality management in the face of ongoing requirements change—at *change time*. Properly implemented, SOA enables organizations to effect requirements changes via declarative reconfiguration of Service metadata. Therefore, change time quality management focuses on metadata, and how well those metadata satisfy the requirements that apply during change time. Such requirements fall into two categories: ongoing, day-to-day requirements that reconfiguration can address, and the broader meta-requirement of agility.

The intractability of change-time metadata quality assurance results from the open-ended nature of reconfiguration in SOA. If architects fail to fully plan for this problem ahead of time, the sorts of changes users might introduce over time could be entirely unpredictable and unmanageable. Fundamentally, SOA empowers users, which in the absence of adequate governance can lead to anarchy. The important point to keep in mind is that change time metadata

quality assurance should be handled as a matter of policy. Apply Service-oriented principles to change time quality assurance in order to create, communicate, and enforce policies for metadata-based changes. Then you can treat those policies like other policies that your SOA governance infrastructure deals with.

VI. The ZapThink Take: The Emergence of the SOA GQM Suite

Considering change time quality assurance to be a matter of policy enforcement completes the GQM picture. It is imperative that companies place significant effort on this governance/quality/management aspect of SOA infrastructure, as these capabilities are critical to achieving the business benefits of SOA. As a result, companies should focus on the kind of SOA-specific enablement technology that is central to the goals of SOA, a combination of capabilities we might call the SOA GQM suite. The suite might not necessarily be a single-vendor product solution, even though it seems the market might actually be converging in this manner. Regardless of whether it is a single-vendor solution or a collection of best-of-breed applications, the SOA GQM suite is the collection of tools and capabilities that facilitate all three feedback loops that this paper discusses.

In many cases, the SOA registry/ repository is at the center of the GQM suite.

This complete feedback loop that links the three loops above is already becoming a reality in successful SOA projects. Right now, companies should focus on addressing each part of the GQM suite by making sure that they are implementing all of the feedback loops. In many cases, the SOA registry/repository is at the center of the GQM suite, because it acts as a central repository for Service assets for design time, run time, as well as change time. But whether through home-grown solutions, best-of-breed vendor solutions, or packaged offerings that aim to provide the whole suite of offerings, companies must properly address their GQM needs if they ever hope to realize the benefits of SOA.

Copyright, Trademark Notice, and Statement of Opinion

About ZapThink, LLC

ZapThink is an Enterprise Architecture (EA) strategy advisory firm. As a recognized authority and master of Service-Oriented Architecture (SOA) and EA, ZapThink provides its audience of IT practitioners, consultants, and technology vendors with practical advice, guidance, education, and mentorship solutions that assist companies in leveraging SOA to meet their business needs and presenting viable SOA solutions to the market. We provide this audience a clear roadmap for standards-based, loosely coupled distributed computing – a vision of IT meeting the needs of the agile business.

ZapThink provides IT practitioners strategic insight and practical guidance for addressing critical agility and change management issues leveraging the latest EA and SOA best practices. ZapThink helps these customers put EA and SOA into practice in a rational, well-paced, and best practices-driven manner and helps to validate or recover architecture initiatives that may be heading down an unknown or incorrect path. ZapThink assists with solution vendor, technology, and consultant selection based on in-depth, objective evaluation of the capabilities, strengths, and applicability of the solutions to meet customer needs as they relate to EA initiatives and as they map against emerging best practices. ZapThink enhances its customer's skills by providing education, credentialing, and training to EAs to develop their skills as architects.

ZapThink helps to augment consulting firms' EA offerings and intellectual property by providing guidance on emerging best practices and access to information that supports those practices. ZapThink provides frameworks for product-based consulting based on ZapThink insight and research, such as SOA Implementation Roadmap guidance, Governance Framework development, and SOA Assessments, and provides a means to endorse and validate consulting firm offerings. ZapThink also accelerates consulting firms' efforts to attract, retain, and enhance the skills of EA and SOA talent by providing education and skills development

For solutions vendors, ZapThink provides retained advisory for guidance on product strategy, as well as marketing, visibility, and third-party endorsement benefits through its marketing activities, lead generation activities, and subscription services. ZapThink enables vendors to leverage ZapThink knowledge to transform their offerings in a cost-effective manner.

ZapThink's Managing Partners are widely regarded as the "go to advisors" and leading experts on SOA, EA, and Enterprise 2.0 by vendors, end-users, and the press. Respected for their candid, insightful opinions, they are in great demand as speakers, and have presented at conferences and industry events around the world. They are among the most quoted experts in the IT industry.

ZapThink was founded in October 2000 and is headquartered in Baltimore, Maryland. Its customers include Global 1000 firms and government organizations, as well as many emerging businesses. Its Managing Partners have worked at such firms as IDC, marchFIRST, and ChannelWave, and have sat on the working group committees for standards bodies such as RosettaNet, UDDI, and ebXML.

Call, email, or visit the ZapThink Web site to learn more about how ZapThink can help you to better understand how SOA and Enterprise 2.0 will impact your business or organization.

ZAPTHINK CONTACT:
ZapThink, LLC
108 Woodlawn Road
Baltimore, MD 21210
Phone: +1 (781) 207 0203
Fax: +1 (815) 301 3171
info@zapthink.com

ZapThink, LLC • 108 Woodlawn Road • Baltimore, MD 21210 • info@zapthink.com • www.zapthink.com