

# The SOA Magazine

## Feature Article



### Quality Assurance for SOA Through Process Cadence

by Wayne Ariola

Published: May 3, 2007 (SOA Magazine Issue VII: May 2007, Copyright © 2007)

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

*Abstract: These days, most organizations moving ahead with SOA projects are aware of the fact that traditional design and development approaches will need to undergo some significant changes. Often overlooked, however, are the processes in place for ensuring the ultimate quality of delivered service-oriented solutions. This article explores the concept of "process cadence" as it applies to the modernization of quality assurance in support of SOA.*

#### Introduction: The Status of Quality Assurance

Having been in this industry for the past 20 years, the most significant "change" I've noticed is the speed at which "change" occurs: it is constantly accelerating. Today, IT organizations trying to keep pace are turning to service-oriented architectures with the aspirations of having an agile IT infrastructure capable of accommodating ever-changing business demands.

SOA offers companies an efficient and robust way to lower integration costs, increase business agility, and consolidate applications by reusing valuable business components. It further promises simplified access to IT services and the capability to share data with business partners, customers, and information systems with unparalleled efficiencies.

However, many organizations fail to fully realize these expected benefits simply because the quality processes they employ are inadequate. Quality processes are behind the times because they are not in step with the flexibility and agility that an SOA infrastructure can deliver.

Businesses traditionally have a linear process for quality where the quality assurance (QA) group waits anxiously at the end of the development process to test the application in a staged environment. SOA has two distinct impacts on this approach. First, the serialized nature of this process erodes the flexibility and agility benefits that the organization is trying to achieve. Second, given the distributed nature of a service-oriented solution, it can become very challenging to stage a QA environment capable of addressing the complexities of composed services.

In this article, I explore why reaping all of SOA's benefits requires a "process cadence" that brings your organization's quality process in alignment with the world of service-oriented computing. As part of this exploration I will introduce preliminary steps for establishing a technical infrastructure capable of supporting QA requirements for SOA projects.

#### Process Cadence

Most organizations turn to SOA in hopes of achieving benefits such as:

- *Business Effectiveness* (agility and responsiveness to market/competitive dynamics, greater process efficiencies, deployment of resources based on business needs, closing the gap between business and IT)
- *Cost Efficiency* (reduced maintenance and integration costs, reduced skills and effort to support business change, reduced application redundancy, improved payback times)

- *Decreased Risk* (higher level of IT quality, higher level of application and process consistency, incremental deployment)

Yet, if you consider traditional QA processes in which quality assurance is comprised of a set of tasks that occur in a serial fashion subsequent to development, it becomes clear that many of these desired SOA benefits are placed at risk. These risks are compounded by four very distinct QA challenges:

1. Given the distributed and complex nature of SOA, it can be virtually impossible to “stage” an SOA environment.
2. Given the technical nature of the message layer, many QA staffers are not capable of constructing the necessary tests.
3. Given the general definition of roles and responsibilities within QA and development, there is minimal collaboration on “quality assets.”
4. Given the high level of integration, more business domain expertise is required to achieve robust tests.

The key to overcoming such challenges is achieving a process cadence that initiates the quality process as soon as services are defined. A collaborative and building blocks-based approach to quality is particularly well-suited for handling the complexity that comes with larger SOA implementations. If businesses want to achieve agility through rapid incremental deliverables, then organizations cannot wait for a serialized quality process.

### Shared Business Services vs. Application Services

Ultimately, there are two fundamental approaches (Figure 1) towards realizing SOA: the top-down approach that results in the creation of shared business-centric services and the bottom-up approach that tends to produce more application-centric services. Each of these approaches has distinct benefits and challenges.

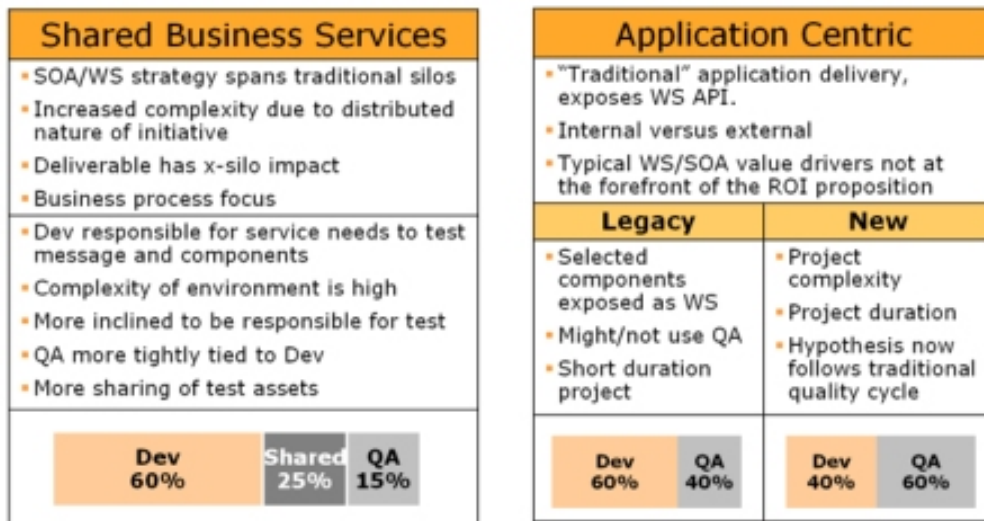


Figure 1: A summary of shared business services and application services.

For example, given the business process view taken by the top-down approach, the QA department can be faced with testing a process that crosses multiple application silos, multiple departments, and perhaps even two or more external business units. From a QA point of view there are several common impacts, including:

- Increased complexity related to establishing a suitable testing environment.
- Inefficient manual testing processes due to the fact that increased message logic is difficult to test using traditional user interfaces.
- Expert knowledge of the business domain that is required to truly test the environment.

Project teams proceeding with the bottom-up approach are in for some challenges as well, including:

- A distinct level of knowledge of the technical architecture is required to exercise the service.

- Given the scope of the exercise, it might not be efficient for QA to stage an environment for testing at all.
- As with the top-down process, the testing of logic embedded within messages remains challenging.

### **Quality Assurance of Service-Oriented Solutions**

Regardless of which delivery approach is chosen, development teams must be responsible for testing service APIs. QA professionals must then be able to leverage those tests in order to exercise (test) the services within a broader set of business scenarios. This is similar to quality processes in the embedded systems industry.

Like embedded systems, service-oriented solutions are comprised of an interrelated set of components (services) that interoperate on an assumed (or fixed) set of standards. Furthermore, SOA implementations can span multiple tiers that may be evolved, versioned, or changed at varying intervals. There can be multiple independent parties involved in the creation of services, and some services may even be delivered by external entities.

When having to take these considerations into account, the testing of a service-oriented solution will require an approach that can handle these potential complexities. In larger, more sophisticated systems, quality measures must permeate every aspect of solution design in order to guarantee acceptable end results. Therefore, a “building blocks” approach is often considered the logical choice.

Each service will have to be tested and exposed independently to verify service “goodness.” Because the level of interoperability between well-designed services typically is high, the level of certainty in the quality and functionality of these building blocks must also be high. Operations are intimately tied to the implementation layer represented as code. Therefore, testing requires a healthy understanding of the architecture because not only is the organization responsible for exercising the service interfaces, but now the underlying components and implementation layers must also be subjected to quality assurance.

Traditional QA processes will almost always need to change in support of SOA initiatives. The roles of development and QA teams might remain the same, but their respective responsibilities will generally shift. The major difference in today’s service-oriented world is the fact that the level of abstraction moves up. Whereas in the past the QA team might have been accustomed to testing a contained application, they will now need to set up a staging environment wherein they have the freedom to perform a variety of tests across various application boundaries.

### **A Step-by-Step Approach to Process Cadence**

Following is a set of steps that improve the QA process in order to realize process cadence:

1. Provide Visibility
2. Supply an Infrastructure for Reuse
3. Promote Bottom-Up Quality
4. Leverage the Infrastructure for Top-Down Quality
5. Assist the Management of Design Complexity
6. Concentrate on Quality Process Improvement

These six steps are described in more detail in the following sections.

#### *1. Provide Visibility*

Similar to the certification programs that promote used cars into “certified pre-owned vehicles,” inspection points for business services should be established before starting an SOA project. Such a process must be objective, reliable, repeatable, and clearly defined in order to promote trust in business services that are to be reused either internally or externally.

When exposing business information internally, or by sharing data with partners externally, the goal is generally to demonstrate that each part of a system is reliable. Visibility will ultimately promote trust and trust will promote reuse of business assets. The caveat here is that the measurements are objective, automated, and available.

Internally, the organization has a distinct goal of promoting reuse of business assets. The challenge of reuse is truly a cultural shift in the way that developers and architects have traditionally delivered projects. Given this long-standing, cultural barrier, the quality metrics need to tell a very distinct story for the internal constituents. The data should give the organization the confidence and trust that the business asset is robust enough for the application. The negative case is also true: the architect should also be able to determine that the service asset is not robust enough for a specific application.

Externally, visible quality metrics provide "credit reports" for specific services. This type of report assists in eliminating the finger-pointing when it comes to "on-boarding" customers within an SOA project (which may still be perceived as more of an integration activity). Although SOA introduces more stable, interoperable standards, integration with business partners still requires dedicated time and effort. While finger-pointing may still occur, having the asset credit report available to external partners can alleviate this situation.

Promoting trust for an asset must begin early, as soon as it is created. Visibility into asset quality helps drive the development cycle and increases trust for later reuse. In order to promote trust early, the business must define policies that govern the different aspects of the services life cycle (runtime and design-time). For example, policies that govern the development of services implemented as Web services will need to address standards compliance such as schema validity, semantic validity, WS-I compliance and a definition of adopted WS-\* standards. Such policies are critical to achieving consistency and ensuring reuse and interoperability.

Policies also include best practices, both general to many industries (and related to the technical aspects of the underlying SOA meta-data and artifacts), and specific to the organization's goals. Examples of these domain-general policies include security, maintainability, reusability, and any other policies that are tailored to domain-specific requirements.

Once the policies are defined, it is critical that they are executable and that their application can be verified, tested and measured. Typically, policy delivery and management will fall into the hands of architecture groups that ensure that policies are applied to SOA artifacts by development, testing, and quality assurance groups, as illustrated in Figure 2.

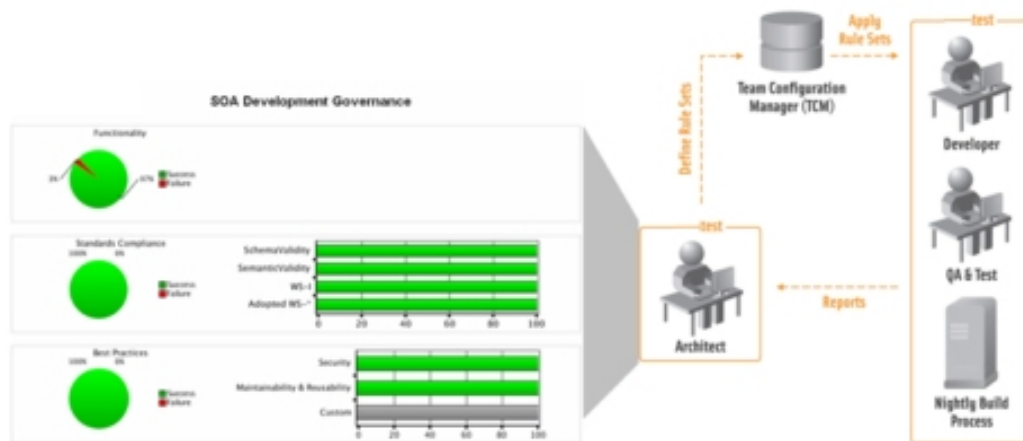


Figure 2: Modern quality assurance as it relates to a service delivery lifecycle.

[click here to view larger image](#)

The lack of visibility and the absence of an objectively measurable process can lead to the organization and its partners to simply not reuse business services. A significant reduction in trust can therefore significantly deteriorate the potential of an SOA project.

## 2. Supply an Infrastructure for Reuse

Reuse of service assets is an inherent aspect of SOA and one of its primary value propositions. As a result, it is critical for existing quality processes to evolve in order to fit the service reusability paradigm by adopting testing processes capable of reusing underlying test assets.

A robust automation infrastructure needs to be in place in order to facilitate the reuse of such test assets. This type of infrastructure can be comprised of collaboration and/or sharing-enabling products. For example, you would not want people redoing, recreating, and rerunning the same tests over and over again; instead, it is preferable to ensure that

test assets are created, shared, leveraged, and extended properly among team members.

In order to accomplish this, you need an environment that allows you to store these test assets. This could be something as simple as a source control system, or a more specialized test repository solution designed specifically to manage test cases and their execution details. Of course, such an environment can be used with any software development process, but it is especially applicable to service-oriented solutions because they are inherently based on shared and reused services.

### 3. Promote Bottom-up Quality

Most businesses worry about the externally-visible functionality of an application. But because services can be reused in unpredictable fashions, ensuring quality at the message layer only is not sufficient. Besides, there are other considerations. For example, to what extent is the service maintainable and to what extent can it evolve with business needs?

Therefore, you must also ensure that the underlying application is robust. To achieve quality at the application level, certain activities (such as coding standards, static analysis, unit testing, etc.) need to be constantly carried out in an efficient manner. Once quality metrics and assumptions become clearly defined and visible in the underlying assets, they can be better trusted and reused.

### 4. Leverage the Infrastructure for Top-down Quality

Top-down quality workflows are best embodied at the messaging layer of an SOA system. This is where you should start, but do not limit your processes here.

Top-down quality ensures that the service at runtime adheres to the quality requirements that are imposed upon it. Top-down quality workflows are best embodied at the messaging layer of an SOA system where organizations must ensure that the service processes deliver the desired business requirements. This includes activities such as verifying and enforcing standardized WSDL and WS-BPEL definitions, as well as design and development policies (Figure 3). Additionally, security should be addressed as part of the software development lifecycle rather than an after-the-fact activity.

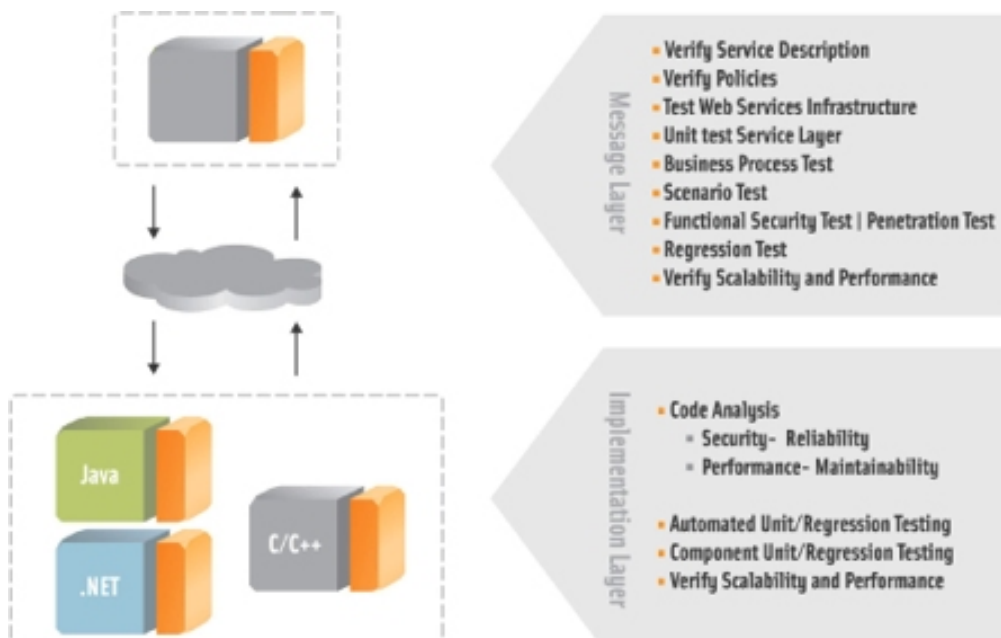


Figure 3: Common practices needed for quality at both the message layer and the application layer.

### 5. Assist the Management of Design Complexity

In order to meet the demands of developing, testing, and maintaining a complex SOA, you need to have an infrastructure that allows you to stub out and emulate individual services. Because certain service endpoints may be out of your control, you must be able to emulate service consumers and clients while also being able to create a “test harness” that can simulate more complex processes (such as asynchronous messaging and long-running WS-BPEL

compositions).

Even more critical is the case of B2B partners where providers and consumers are not available. In this situation, you need to provide a “runnable” test driven development (TDD) environment to permit both endpoints to work in parallel and to optimize the development process. The key here is to provide development and test environments that are as closely similar to the production environment as possible.

#### *6. Concentrate on Quality Process Improvement*

The optimal process is one that continues to improve. Because SOA introduces architectural and structural differences, it causes a separation between QA and development with architecture-heavy processes.

The quality of the overall process is most vital at architectural and business enablement levels, as well as the service, application, and team levels. Therefore, it is critical to have a unified dashboard that spans and ties these different levels in order to measure and track the overall process quality.

#### **Conclusion**

The steps discussed in this article will assist in bridging traditional gaps when applying quality assurance processes to service-oriented solutions. Promoting these tasks in an overall quality strategy will help evolve reusable business services and furthermore drive a visible process that begins when a requirement is defined and delivers the process cadence necessary to produce quality services that become true IT assets.

Once an efficient quality process is established, the key is to maintain that process and continually evolve and optimize it. After this process becomes increasingly visible, the quality assurance in general becomes naturally more predictable. The end result is that project teams benefit from being able to make better decisions, set more obtainable goals, and achieve more desirable outcomes.

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL



[home](#) [past issues](#) [contributors](#) [official book site](#) [legal](#) [rss](#)

Copyright © 2006-2007 SOA Systems Inc. All Rights Reserved