

The SOA Magazine

Feature Article



Service Development Lifecycle Controls for Creating a Service Factory

by Clive Gee and Robert Laird

Published: February 23, 2009 (SOA Magazine Issue XXVI: February 2009)

[Download PDF](#)

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

Abstract: The concept of a software factory describes a practical work-product approach to governing an efficient service factory - a software engineering-based approach to defining, developing, testing, deploying, and operating functional services and automated business processes. All services follow a similar lifecycle of analysis, followed by design, development, deployment, and ongoing management.

Because the service creation process is repetitive, a production engineering approach to automating software development can be used. The Production Engineering method required a significant effort up front, creating a specialized production or assembly line that can then mass-produce the product efficiently and in quantity. In effect, we are building a Services Factory: much of the purpose of SOA governance is to define how that factory can operate most effectively.

In the following excerpt from the book "SOA Governance: Achieving and Sustaining Business and IT Agility" [REF-1] we will take a look specifically at service development lifecycle control points.

Introduction

While most organizations have some form of a system development lifecycle (SDLC), the nature of creating shared services is best guided by an SDLC with sufficient governance control points to ensure quality of service. This article discusses and explains the key concepts of a governance control point, as applied specifically to the service development lifecycle.

Service Development Lifecycle Control Points

Most organizations already have some type of system development lifecycle (SDLC) and a methodology that is used to perform development, although we often see in practice a lack of enforcement of that approach across different business units, and even if a set of best practices, standards, policies, and patterns has been defined, they are not always enforced.

Effectively enforcing best practices and a consistent SDLC provides a reasonable entry point for real governance, while not being a huge stretch from what is already being performed via the SDLC. At the same time, if the governance maturity level of the organization can be increased to the degree that it is able to govern the SDLC, the organization is then in a much better position to proceed to the next phase of the SOA governance cycle and create program and organization governance.

The danger here for even initial attempts at SOA governance is that often some key individuals view the imposition of any process or governance as being something that might apply to other people but not to them personally. For them, it's an over-engineered, useless exercise that just gets in the way of meeting their own deadlines. So, many governance processes are simply bypassed, or they're followed in a less than an enthusiastic manner. The main reason for this is that governance is imposed from the outside and the

execution is onerous.

What would happen if governance were mostly automated, easy, and added value to the development process and actually helped with project deadlines? Would the skeptics be more willing to take the medicine if it genuinely eased their pain?

To adequately govern the SDLC, there is a need to establish measurements, policy, standards, and control mechanisms to enable people to carry out their governance roles and responsibilities as efficiently as possible, without introducing overly bureaucratic procedures.

Governance of the SDLC may be characterized by the sorts of decisions that need to be made at certain "control points" within the process of services development. A control point is a decision checkpoint that provides an opportunity to measure adherence to the established processes, whether you are on track to meet the targets and goals you have established, and then decide whether the way the processes are executed or managed needs adjusting. Knowing what decisions involved in the process are critical, when to make them, and understanding what measurements are needed to monitor those processes are all essential aspects of governance.

Certain activities within a process may be associated with a control point. At the end of each identified activity, there is a control point at which the governance function decides whether the program is ready to move to the next activity. Each of these milestones is a control point.

At its essence, the governance of the SDLC provides a way to identify control points and to define the governance rules. At each control point, it is necessary to identify the following:

- The roles for who does what at the control point
- The policies to be applied at the control point
- Measurements at each control point that should be applied and collected for later governance vitality actions
- The proof of compliance records to be created and archived

A control point will be created where there is a demonstrated advantage weighing the standardization and efficiency provided versus the time, effort, and possible project delay. The control point enables SOA governance the opportunity to ascertain progress, to communicate this progress, to forecast efforts for subsequent phases of the SDLC based on scope and issues found, to review and report compliance, and to facilitate the injection of expertise and qualified review of the artifacts, process, or decisions made by the development team.

Control points don't have to consist of huge formal meetings. Services and most automated business processes are smaller entities than projects, and there are many more of them. Therefore, the existing governance approach has to be streamlined or it might grind to a halt. We've found in practice that effective control point reviews can be made during regular - typically weekly - sessions of a subset of the SOA enablement. A real productivity aid in performing these control point reviews is the use of previously completed checklists, signed off by one or more senior professionals as certification that one or more tasks has been completed successfully, and that the service, process, or other work product is fit for purpose and ready in all respects for the next task in the development process. These checklists should be viewed as contracts between different experts in the service development process. The most important part of the checklist is the signature block to show who exercised approval authority; people tend to be careful about the quality of anything that carries their personal reputation with it.

Another productivity aid is the use of automated tooling. As much of the governance control point as possible should be automated. This aids in better near real-time feedback to the developers and provides an easy method to recheck work that has been updated. In addition, human beings are busy and will tend to apply governance in an inconsistent manner. Machines are consistent but not usually as flexible as needed. The combination of the two provides an optimal governance mix.

Let's look at the control points needed to govern a generic development lifecycle, at least at a high level. Figure 1 represents a "governance dashboard" monitoring a typical SDLC with an eye toward the key concepts and the points where they must be addressed by SOA governance.

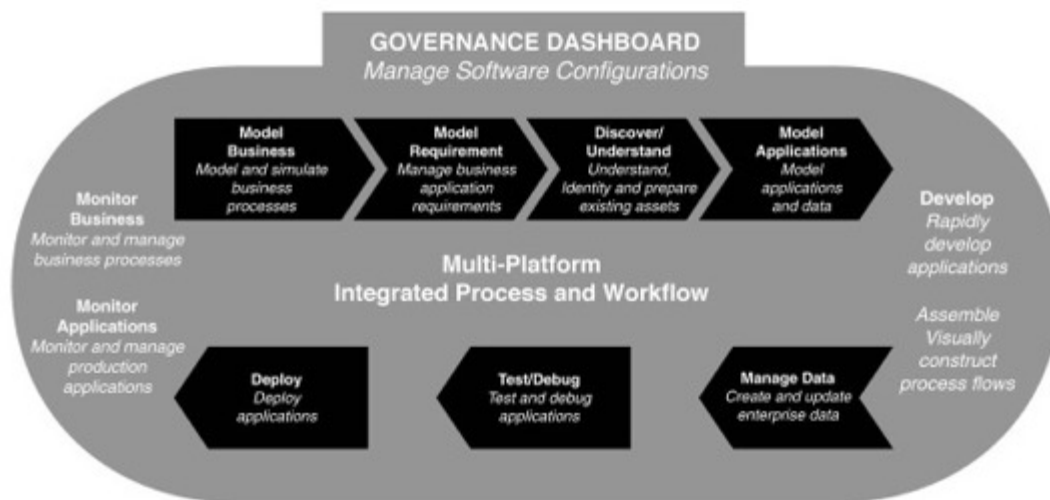


Figure 1:
Software
Development
Lifecycle
Governance
Dashboard

As mentioned previously, we need a streamlined process that can handle the large number of services and automated processes that we need to implement to have real impact on business agility and flexibility. However, that streamlined process must not sacrifice the quality of governance just because of the need for extra speed. That would be an unacceptable trade-off.

Some organizations deal with highly regulated processes that have mission-critical or life-critical products and need to apply highly formal, auditable governance to manage the risks involved. Other organizations have processes with lower associated risks that can be more lightly regulated. We have found in practice that the same governance process can handle both these extremes perfectly well. If there is a need for stricter governance, it can be met with tighter policies at the control points together with more stringent policy enforcement and compliance measurement. If less-strict governance is more appropriate, the same process can be used with less restrictive policies, fewer audits, and lower levels of checklist signoff required.

Even within a single organization, different processes may require different styles of governance. Some processes, such as service certification, require stricter governance than other processes, such as solution architecture. Different organizational cultures require different levels of autonomy in decision making. Good governance requires good judgment.

First, let's update our Figure 1 with the location of these control points so that you have a visual representation in mind as you read their descriptions. Figure 2 shows where the control points occur in that development cycle.

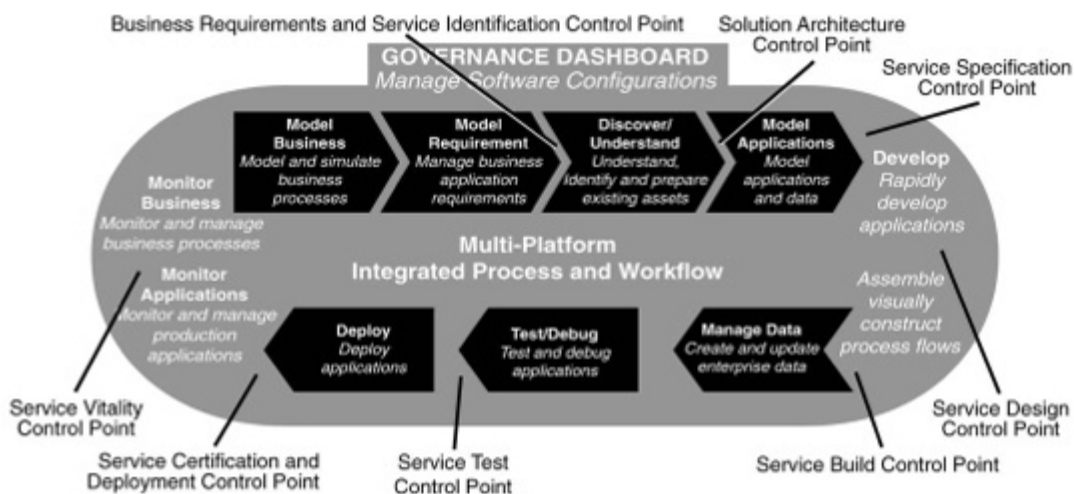


Figure 2:
Software
Development
Lifecycle with
SOA Control
Points

Here are descriptions of these control points.

Business Requirements and Service Identification Control Point

For an SOA approach, there is an emphasis on creating services that provide agility and reuse for the business. This first business requirements and service identification control point consists of a high-level

review to determine that services are being identified in accordance with services selection and prioritization policies.

This first business requirements and service identification control point should address the following types of questions:

- Business goals. What are the business goals that the business seeks to attain and how do we measure the benefits or progress toward the business goals via key performance indicators (KPIs)?
- Do the requirements as we currently understand them clearly support those goals, and do they align with an existing "business heat map"?
- Are those requirements sufficiently understood and agreed to? Are they presented in a form such as use cases, business process models, sequence diagrams, or class diagrams that are consistent with the SOA development approach?
- How do we provide traceability of the requirements so that we can ascertain that those requirements have been met during the development process? Have those requirements been entered into an enterprise-wide requirements and business rules catalog? Is there any conflict with existing entries in that catalog?
- Which of those requirements could be translated into good candidate services, either because they represent functionality that may be needed by multiple consumers or that might be needed for process automation? Which requirements could be better supported by deploying applications, automated processes, or manual processes?
- Where we have identified candidate services, have we identified potential consumers, and determined whether any of them have specific requirements that should be considered?
- Given finite IT resources, what development priority should we assign? Is ownership of any new candidate IT asset defined, and is outline funding available for its development?

Solution Architecture Control Point

Different IT developers and groups, if left to make all design decisions on their own, would invariably use completely different platforms, coding languages, tools, styles, methods, and techniques. This variation adds cost and complexity to the ability of the business to make future changes, and makes future maintenance very hard and costly. Further, it reduces the reliability, stability, and interoperability of the organization's IT assets. We have seen this problem at many organizations that we have visited. Simply put, the purpose of the solution architecture control point is to prevent that expensive multiplicity of approaches from occurring ever again.

Essentially, any proposed IT artifact that makes it past this control point is part of the IT build plan. For the area of solution architecture, the governance should control for a series of criteria the following:

- Do the proposed standards, policies, and reference architectures - the solution architecture - identify the standards, policies, and design patterns to be followed in the service implementation? This will include reference architectures, platform standards for hardware, and software-usage standards.
- Have any reusable assets been identified and assessed for suitability? Has the service sourcing policy been followed?
- Have the nonfunctional requirements been identified and assessed? This includes the number of transactions per time unit, a busy hour analysis, the service performance required, presentation access to the service functionality, data managed by the service, space required for the installation of the service, and any dependency and configuration requirements. Governance must validate that all these are considered and addressed.
- Governance must validate that all security policies are being considered and addressed.
- Governance must validate that all legal and regulatory policies are considered and addressed.
- By this stage in the development of IT assets such as services or automated processes, the technical IT staff involved should have a pretty good idea about the complexity of the tasks involved, and the probable level of resources required to complete development. Should development of the asset be

confirmed, the scope reduced, or the asset abandoned?

Service Specification Control Point

A service specification should be created for each service whose development has been approved. Best practices for service design must integrate both an IT and business perspective for the design of the interface and the responsibilities of each service. Because the service specification is, in effect, the organization's face to business partners, customers, and other stakeholders, the service externals - those details of a service that are to be made public - become an important part of the overall business design. The design should take into account the requirements of all potential service consumers (within reason), and be created at a granularity that maximizes business value. For the area of service identification and specification, the governance should control for a series of criteria the following:

- Does the service identified make sense, is at the right granularity, and is not duplicating an existing service?
- Does the service specification follow all SOA standards and policies?
- Does the service specification follow the messaging model? If not, should an exception be granted?

Service Design Control Point

After the service solution architecture has been turned over to the design team, a number of design elaboration decisions must be made. Collectively, these form the service internals - a set of design models, notes, and advice that will guide the service developers as they create and test the service code. For the area of service design, the governance should control for a series of criteria the following:

- Has a service architect confirmed that the design should be able to meet the nonfunctional and functional requirements for this service?
- Have the service designer and data architect agreed that the service can be made to conform to the signature (that is, inputs and outputs) described in the service externals?
- If a service is wrapping an existing or planned application, are the necessary interfaces to that application well defined and stable (that is, won't change if a new version of that application is installed)?
- Have the monitoring metrics (for example, usage, quality of service [QoS] levels) been established?
- In the case of automated processes, have the monitoring requirements been defined and planned?
- In the case of long-running automated processes, have all the necessary actions to handle recovery from process errors or technical failures been addressed?
- Is the overall quality and level of completeness of the service specification package good enough that the service developers or process developers can complete development without further input?

Service Build Control Point

After the service design has been turned over to the service build team, a number of implementation decisions need to be made before development of the code or executable model. In the interests of consistency and quality, we strongly recommend the use of code walkthrough reviews, where peers (that is, other service developers or process developers) review the work in progress and offer constructive criticism.

The service build control point is effectively the last of these code walkthroughs, and should be performed with slightly more formality than the others. Questions that should be addressed include the following:

- Was the asset coded in accordance with the design?
- Does the code follow the accepted coding standards?
- Have all the associated artifacts (for example, load libraries, metadata files, resources) been defined to create a transportable build? Have the versions of each of those artifacts been checked to see that there are no version conflicts with services already in production?

Service Test Control Point

Service testing is different from testing complete IT solutions or applications. Because services and automated processes do not have their own user interface, it is not possible to perform user acceptance testing directly on services or automated processes. Code frameworks or specialized tools are needed to exhaustively test services and automated processes thoroughly to avoid uncovering problems during later formal user acceptance testing when the rest of the IT solution that uses those services or processes has been completed.

SOA governance must ascertain that the services test is being performed in a manner conducive to a services approach, and that exhaustive functional and nonfunctional tests have been passed before releasing any SOA asset to production. The service test team must create and use the right service test environment with tools and data to affect a comprehensive test. This should include the following:

- Using the optimum set of service test tools and frameworks.
- The use of an automated build and test environment that can enable fast changes of the tested software and regression testing. This environment must closely resemble the production environment.
- A load/stress test tool to test nonfunctional requirements, specification, creation, and loading of realistic but artificial test data.
- A test management reporting tool to keep management apprised of the testing status.
- Trace the test case to the original user requirements.

Service Certification and Deployment Control Point

The objectives of the deployment are to migrate the services to the production environment while minimizing client downtime and impact on the business. This process is subject to many errors if performed manually. It is vital that the correct version of the services be deployed and that any deployment binding with other services and applications be performed quickly and correctly. Areas for governance to validate include the following:

- The use of a tool that automates the deployment and back-out process.
- Final certification checks have been made against the services to verify compliance with all policies and standards and being able to demonstrate that what was tested matches not only the requirements but what was delivered, and that no corrective changes made during testing have invalidated other test results.
- IT operations have completed acceptance testing and have formally accepted the asset, signifying their confidence in being able to operate it within the terms of the QoS specified for it.
- The service registrar and business service champion have reviewed the service description in the service registry and approved it.

Certification of a service or automated process is a formal "passing out" ceremony, and granting of certification should signify that the SOA enablement team is happy for their reputation to be associated with the performance of the new asset.

Service Vitality Control Point

Service vitality takes place periodically as part of SOA governance to check up on and update the governance processes, procedures, policies, and standards in reaction to the results of the real world. This involves examining any and all lessons learned in any of the SOA planning, program control, development, or operations activities. It also includes such things as comments and feedback from all stakeholders and an examination of any common patterns (for example, common exemption requests or common reasons for failure to pass one or more control points) that need remedial action. Metrics in the efforts required for each stage of the development process can show trends that indicate improvements or declines in their vitality.

A formal service vitality control point review should be conducted every three to six months to determine whether the SOA transition remains on track, and whether the level and style of governance is optimal.

Individual service or automated processes should be reviewed every 6 to 12 months. Usage data of all

versions of each service can determine any "stale" versions that can be deprecated or deleted, and whether the deployment options taken and decisions on who should own and who should access each service are optimal.

Conclusion

We have focused in this extract on SOA Governance service development control points as a method to create a software engineering capability of a service factory. The factory is a production line for services. All services pass through a common, repeatable series of development, deployment and management steps. Quality and governance is built-in throughout the entire process.

References

[REF-1] "SOA Governance: Achieving and Sustaining Business and IT Agility" by William A. Brown, Robert G. Laird, Clive Gee, Tilak Mitra (IBM Press, ISBN 0137147465, Copyright 2009 by International Business Machines Corporation. All rights reserved.)

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 © 2006-2009
SOA Systems Inc.